JAERI-M
9283

20MVタンデム加速器データ収集システムの拡張

１９８１年２月

冨田　芳明

日本原子力研究所
Japan Atomic Energy Research Institute

JAERI－M 9283

２０ＭＶタンデム加速器データ収集システムの拡張

日本原子力研究所東海研究所物理部
冨田　芳明

（１９８０年１２月２５日受理）

２０ＭＶタンデム加速器データ収集システムのプログラムの拡張を行ったので報告する。今回の拡張によって標準的でないデータの取り込みや，標準でないＣＡＭＡＣモジュールの使用がプログラムの一部をきめられた処方にしたがって変更するだけで容易に行えるようになった。変更はＭＢＤのマイクロプログラムを書くことによって，あるいはリストモードにおけるリデュースドスペクトルのサブルーチンを書くことによっての二通りのやり方で行われる。標準のプログラムを使用する場合でもリストモードでは３２台までのＡＤＣが使用できるようになった。今回の拡張は主としてデータの取り込み方に柔軟性は持たせるためのものであるが，装置のコントロール等の目的のためにも応用できる。この報告は実際にプログラムを変更しようとする人のための手引として書かれている。

JAERI-M 9283

Expansion of the Data Acquisition System for the 20 MV Tandem Accelerator

Yoshiaki TOMITA

Division of Physics, Tokai Research Establishment, JAERI

(Received December 25, 1980)

This report describes an expansion of the program of the data acquisition system for the 20 MV tandem accelerator. By the present expansion it became possible to change the accuisition mode or to use non-standard CAMAC modules with partial modification of the program according to well defined prescriptions. The modification can be made by writing microprograms for the MBD or appending subroutines for the reduced spectra in the LIST mode data acquisition.

The new program can handle upto 32 ADC's in the standard LIST mode data acquisition.

The present expansion aimed to increase the flexibility in data acquisition. It can also be applied to control experimental devices.

Keywords; Data Acquisition, Program, Computer, CAMAC, Tandem Accelerator

目　　次

Contents

1. はじめに

２０ＭＶタンデム加速器のデータ収集システムにおいては，当初の目的であった標準的なＡＤＣを使用したりストモードおよびＰＨＡモードのデータ収集が可能になっており，その概要の報告も行われている。[1)]

しかしタンデムで行われる多様な実験に対応するためには当初のシステムでは不十分であり，拡張が要求されてきた。システムがCAMACを採用しているため，ハードウェア的には拡張は容易であるが，ソウトウェアに関してはプログラムのサイズがかなり大きく，いくつかのタスタが互いに連絡をとりながら動くこともあって，変更のためにはプログラム全体に対する知識とかなりの労力が必要とされていた。

今回行ったプログラムの書きかえはこの点を改善するためのものであり，データの取込み方に対する変更はプログラムのごく一部を変更するだけで可能になった。しかもこの変更に際してはプログラムの他の部分に関する知識はほとんど要求されないようになった。今回の変更は主として核反応実験からの要請にもとづいたものであり，形式的にはリストモードおよびＰＨＡモードというわく内のものである。しかしこの形式にはそれほど大きな制約があるわけではなく，実験装置のコントロールも含めてかなり広い範囲に応用でき，システムの柔軟性はかなり増したと考えられる。

この解説は実際にプログラムを変更する際のガイドとして書かれているが，自分でプログラムを変更しない使用者のためにもどのような拡張が可能であるかを知る上で役立つであろう。またオペレーティングシステムＲＳＸ１１Ｍを理解している使用者はこの解説（特に第３章）を読むことによって，操作手順の背景にあるロジックが理解でき，操作手順を覚える上でも，またシステムのエラー時の処置を可能にする上でも役立つものと思われる。読者の予備知識としては，文献１）に書かれている程度のシステム全体に関する知識と，システムでのある程度の使用経験を予想している。実際にプログラミングを行うためには，ＭＢＤのインストラクションに関する知識[2)]（リデュースドスペクトル作成のためには不要），ＰＤＰのインストラクションに関する知識[3)]，およびアセンブラーに関する知識[4)]（ごく初歩的な知識でよい）が要求される。

なおこの解説でのべる範囲内でどのような変更を行っても，オペコンとグラフィックディスプレイを使用したシステムのコントロールと収集モードの設定，収集データのディスプレイや磁気テープへのダンプ等はすべて標準的なサービスを受けることができる。

第２章において今回の変更の概要がのべられ，第３章では関連するプログラムの流れとイニシャライズ時の入力についてのべている。第４章はＭＢＤのプログラムを書きかえることによるデータの取込みの変更の仕方の解説であり，第５章はリデュースドスペクトルの作成法の解説である。第６章では今回の変更のラインにそった今後の拡張についてのべている。第７と８章はプログラミング上必要とされるサブルーチンやデータの構成についての解説である。附録１－４にはプログラムの参考リストをあげる。附録５以降は今回の書きかえによって変更された事項のうち今後のプログラムのメインテナンスに必要と思われるものを補ったものである。

2. 変更の概要

今回変更されたプログラムは“ＹＴ－４”とラベルされたディスクに入っている。システムのCOMMONであるCOMTBLのサイズを大きくし，実際に利用していないDEC－NETのためのパーティションを取除いた。したがってシステムディスクも従来のものと互換性がない。主な変更は次の通りである。

(1) データをＣＡＭＡＣからメモリーに取込むＭＢＤのマイクロプログラムにプログラム番号を与え，［ＩＮＩＴ］の際に番号を指定することによって対応するプログラムがＭＢＤにロードされるようにした。このことによってデータの取込み方の変更や標準的でないCAMACモジュールの使用が，マイクロプログラムのファイルを用意しておくだけでいつでも行えるようようになった。各マイクロプログラム（例えばリストモードのプログラムＬＩＳＴ）毎に１０種類までのプログラムをディスクに入れておいて切換え使用が可能である。

(2) リストモードで共有メモリー中のリストバッファーに取込まれるデータから作成するリデュースドスペクトルの作成に関して“ＴＹＰＥ”と呼ばれる番号を設けた。PDP１１／０４（以下１１／０４と省略）のプログラムATOMICに“ＴＹＰＥ”に対応するサービスルーチンを追加することにより，［ＩＮＩＴ］時にその番号を指定して，異った種類のスペクトルを作成することができる。この変更を容易にするためにATOMICのリデュースドスペクトルに関連した部分を別のファイルのRDSP０４として独立させた。

(3) １１／０４は乗除算命令や浮動小数演算命令を持っていないので複雑な処理には向いていない。このためにタスクRDSP５５を作成して１１／５５でもリデュースドスペクトルが作成できるようにした。このプログラムはスケジューラーの部分を除いてRDSP０４とほとんど同じであり，RDSP０４の書きかえの知識があれば容易に書きかえることができる。

(4) (2)にのべたリデュースドスペクトルの追加等のために内容の異ったいくつかのATOMICを切換えて使用したい場合がある。このため［ＩＮＩＴ］時にATOMICのファイル名のイクステンションフィルドを指定できるようにした。

(5) 標準のプログラムを使用する場合でも，リストモードでは従来の８名のＡＤＣインターフェースの他に２４台までのＡＤＣが使用できるようになった。リデュースドスペクトルの作成の際にはメモリーを節約するためスペクトルの一部だけを必要な分解能でメモリーに入れることができるようにした。

(6) 以上の変更を容易にするため［ＩＮＩＴ］の画面にいくつかの新しい入力を追加した。これらの多くはプログラムの変更に際して自由に使用でき，ユーザーとインターフェースの部分を変更することなしに収集モードの拡張が可能になった。

(7) これまでの使用の結果判明したプログラムの誤りを修正した。また従来［ＩＮＩＴ］における収集モードの変更の後処理にかなりの待時間があったのを短縮したり，しばしば必要とされた１１／０４の再起動を不要にしたり，積分においてバックグラウンドの差引きを可能にする等，いくつかの改良を行った。

3. プログラムの動作

ここでデータ取込みの拡張に関連したプログラムの流れを簡単にのべる。また新しい [INIT] 時の入力についてものべる。当初のプログラムに関しては作成者によって書かれた内部仕様書[5]があるが，今回の変更によってプログラムの流れが変った部分についてもここでのべる。

3.1 [INIT] におけるパラメーター設定

データ収集のためのパラメーターの設定はタスクＩＮＩＴとＩＮＩＴによって起動されりタスクEDICMC, EDIMOD, EDIMAP, EDIDMPによって行われる。EDICMCによって作成されるCAMACの構成の画面にはFig. 1に示すように３つの新たな追加が行われている。

(1) Ａ００－Ｃ０７ の２４個の新たなCAMACモジュールの指定が行える。これらのデータはCOMTBLにEXTRACとして入れられ [INIT] [END] によって１１／０４に送られる。標準的なリストモードのデータ収集ではADC1～8 以外のＡＤＣを使用する場合にはＡ００からはじめて必要な個数だけＣＮを入力する。またＸ１の欄には各ＡＤＣのビット長が入れられる。この場合にはＸ２は使われない。標準でないプログラムではこれらをどのように使うかはプログラム作成者の自由である。

(2) “EXTRA PARAMETER”として１２個のフリーパラメーターが指定できる。これらはCOMTBLのEXTRAPに入れられ１１／０４にも送られる。標準的なプログラムでは使用していない。便宜上半分は１０進で半分は８進数で入力できるようにしてある。

(3) １１／０４のプログラムの指定

１１／０４の標準的なプログラムは

DK1：〔１０，１４〕ATOMIC. STD

として登録されている，イクステンションのＳＴＤの代りに別の指定を行えば指定されたプログラムがタスクLODPRGによって１１／０４にロードされる。

CAMACの構成に変更があった時は，データ収集モードに対する再チェックが必要なので [END] を押しても [NEXT] と解釈される。

タスクEDIMODによって作成されるデータ収集モードの設定の画面をFig. 2に示す。ここでも３つの新しい入力が追加されている。

(1) マイクロプログラムのプログラム番号

標準でないマイクロプログラムを使用するためのもので，第４章にくわしくのべる。標準のプログラムは番号０である。

* CAMAC CONFIGURATION *

#	MODULE	C	N	GL	
$$ 00	LAM G-1	1	23	13	
01	LAM G-2	2	22	12	
02	C.LAM	1	08	24	CHNL-7
03	ADCI-1	1	20	24	CHNL-7
04	ADCI-2	1	19	23	CHNL-6
05	ADCI-3	1	18	22	CHNL-5
06	ADCI-4	1	17	21	CHNL-4
07	ADCI-5	1	16	20	CHNL-3
08	ADCI-6	1	15	19	CHNL-2
09	ADCI-7	1	14	18	CHNL-1
10	ADCI-8	1	13	17	CHNL-0
11	GATE	1	06	00	
12	CLOCK	2	21	02	
13	SHAPER	2	20	00	
14	PS-1	2	19	16	
15	PS-2	2	18	15	
16	LT-1,2	2	17	11	
17	LT-3,4	2	16	10	
18	LT-5,6	2	15	09	
19	LT-7,8	2	14	08	
20	SCALER	2	13	07	
21	INTRPT	2	12	14	

#	C	N	(X1	X2)
A00	1	01	(10	0)
A01	1	02	(11	0)
A02	1	03	(10	0)
A03	1	04	(11	0)
A04	1	05	(10	0)
A05	1	09	(11	0)
A06	1	10	(10	0)
A07	1	11	(10	0)
B00	2	01	(11	0)
B01	2	02	(10	0)
B02	2	03	(11	0)
B03	2	04	(10	0)
B04	2	05	(11	0)
B05	0	00	(00	0)
B06	0	00	(00	0)
B07	0	00	(00	0)
C00	0	00	(00	0)
C01	0	00	(00	0)
C02	0	00	(00	0)
C03	0	00	(00	0)
C04	0	00	(00	0)
C05	0	00	(00	0)
C06	0	00	(00	0)
C07	0	00	(00	0)

* GATE INFORMATION *

#	SIGNAL	LOGIC	MODE	OUTPUT
0	TTL	PLUS	PARA	
1	TTL	PLUS	PARA	
2	CMC	MINS	PARA	
3	CMC	MINS	PARA	
4	CMC	PLUS	PARA	
5	TTL	PLUS	PARA	
6	TTL	PLUS	SNGL	PULS
7	TTL	PLUS	SNGL	PULS

** EXTRA PARAMETERS **

	DECIMAL	OCTAL
0	0	000000
1	0	000000
2	0	000000
3	0	000000
4	0	000000
5	0	000000

** 11/04 PROGRAM **

ATOMIC.STD

Fig. 1 "CAMAC CONFIGURATION" の画面

＊ DATA ACQUISITION ＊

```
   MICRO-PROG#:  MBDINT=0  STTSTP=0  SCLTIM=0
    ( LIST PRM = 9  FOR  2-PARA      )
    ( LIST PRM = 0  FOR  1-PARA ONLY )
$$ # OF LIST PRM = 4 (+ 013)     PROG#=0

      PARAM    ADC    BIT

        1       5      11
        2       6      11
        3       7      12
        4       1      12

       LIST BUFFER SIZE (BLOCK) : 4


  1-PARAMETER PHA NUMBER OF SPECTRA : 4

      PARAM    ADC    BIT    PROG

        5       2      10      0
        6       3      10      0
        7       4      10      0
        8       8      10      0


  NUMBER OF REDUCED SPECTRA 1-PR:6   2-PR:0

     PRM     GATE          TYP THRSH TRL TRU

  #01 06 P-03 0010,3000   00  0200  00  01
         P-00 0000,0000  (00  0000  00  00)
         P-17 0500,0800  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)

  #02 03 P-00 0000,0000   00  0000  02  00
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)

  #03 03 P-01 0100,1500   01  0004  02  00
         P-03 0100,4095  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)

  #04 01 P-00 0000,0000   02  0002  00  01
         P-03 0200,3000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)

  #05 07 P-00 0000,0000   04  0013  00  00
         P-03 0100,3000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)

  #06 01 P-00 0000,0000   03  0001  01  00
         P-00 0103,0500  (04  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
         P-00 0000,0000  (00  0000  00  00)
```

Fig. 2　“DATA ACQUISITION”の画面

(2) リストモードにおいてADCI1～8（以下では標準ＡＤＣと呼ぶ。）以外からとり込むデータの個数が “＃ OF LIST PRM”のラインの（ ）内に入力されEXTLSTとしてCOMTBLに入れられ，１１／０４に送られる。標準的なプログラムではここに指定した個数だけＡ００から始まるモジュールからデータを入力する。標準でないマイクロプログラムではデータをどこから読むかは作成者の自由である。

(3) リデュースドスペクトルのＧＡＴＥの指定の欄に TYPE，THRESHOLD，TRL，ＴＲＵの４つの入力を追加した。ＴＹＰＥがリデュースドスペクトルの種類を指定する。５０以上のＴＹＰＥは１１／５５のRDSP55によって処理される。くわしくは第６章でのべる。なおＧＡＴＥに対するパラメーター番号は当初のプログラムでは入力できなかったが，変更によって必要なものだけ入力するようになった。標準的なプログラムではこの番号が０のラインのＧＡＴＥは使用されない。したがって“FULL-GATE”は入力する必要はない。これらのデータはAUXPARとしてCOMTBLに入れられ，［ＥＮＤ］で１１／０４に送られる。

なおCAMACの構成を変更したり，ＡＤＣのビット長を変更したりした場合はリデュースドスペクトルに関する入力の全ラインに対してカーソルを通過させてエラーチェックをやり直す必要がある。この操作が終るまでプログラムは［NEXT］［ALT］*）［END］を禁止する。

3.2 ［ＩＮＩＴ］［ＥＮＤ］の動作

［ＩＮＩＴ］［ＥＮＤ］の操作により，ＩＮＩＴは入力されたパラメーターをチェックし，エラーがなければRDSP55を起動し，更に“ＳＥＮＤ−ＤＡＴＡ”ディレクティブによりリデュースドスペクトルの作成の際使用されるテーブルREDINF作成を指示する。１１／５５で作成するリデュースドスペクトルがない場合にはRDSP55は必要なフラグをセットしてＥＸＩＴする。そうでない場合はRDSP55はデータ収集が開始され，DATACQから“SEND-DATA”により，“BUFFER-FULL”が通知されるのを待つ状態になる。

続いてタスクCOMM04がＩＮＩＴにより起動される。COMM04は共有メモリーの先頭の８KWを他のメモリーへ退避させ（この領域が１１／０４にデータやプログラムを送るために使用される。），［０４−ＡＣＴＩＶＥ］の状態をチェックし，オフであれば，タスクLODPRG を起動する。LODPRGは“ＣＡＭＡＣ ＣＯＮＦＩＧＵＲＡＴＩＯＮ“の画面で指定された１１／０４のプログラムをディスクから読み共有メモリーの先頭（１１／０４から見て４００００番地。以下共有メモリーの番地はすべて１１／０４から見た値を示すことにする。また番地はすべて８進法で記述する。）にロード，グラフィックディスプレィに４００００からスタートされるよう表示を行う。ロードされたプログラムの先頭には共有メモリーのプログラムを１１／０４のローカルメモリー（０−３７７７７番地）に移すプログラムがつけられており，１１／０４のプログラムATOMICが１０００番地以後にロードされ，プログラムの実行はATOMICに引つがれる。ATOMICはＴＴＹのインターラプトベクター のセット，ＭＢＤのリセット，CAMAC へのＢＺ

---

* オペコンの［ＡＬＴ］キーは当初のプログラムではオペコンの使用をＩＮＩＴからOPECON に移すために使用されていたが，今回の変更で画面を前にもどすようになっている。（［ＮＥＸＴ］の逆）

の送出等を行い，ＭＢＤの７４００番地以後に１個のCAMAC コマンドを実行するためのマイクロプログラムをロードしＰＤＰから起動されるチャンネル０にリンクする。このプログラムはCAMACのイニシャライズのために使用される。（第4.2節参照） ATOMICの起動は４００００番地をクリアーすることでLODPRGに知らされ，LODPRGはＥＸＩＴして，COMM０４とATOMICの間で交信が始められる。交信はDR１１－C と共有メモリーを使用して行われ，５７７７４と５７７７６番地の２ワードがそれぞれ１１／０４，１１／５５からの通信の内容を示すインターラプトステータスワードとして使用される。インターラプトステータスワードは附録５にまとめてある。交信はまずCOMM０４がATOMICにデータを送りATOMICがそれに答える形で行われ，データ収集時のようにATOMIC側から交信が始められることはない。ATOMICに送られる情報はあとでのべるように［ＩＮＩＴ］での変更の度合に応じて必ずしもすべてが送り直されるとは限らないが，まずすべてが送られる場合についてのべる。

(1) COMM0.4は［ＩＮＩＴ］で作成された，CAMACの構成,データ収集モード，ＡＤＣのビット長，リデュースドスペクトルに関するデータ，プリセット値，スペクトルを入れる共有メモリーの番地等のデータ（第８章にくわしくのべてある。）を共有メモリーにロードする。

(2) ATOMICは共有メモリーからデータを取込み，ＭＢＤをリセット，CAMAC にＢＺを送出し，送られたデータにもとづいて使用されるＧＬに対してＭＢＤおよび各クレートのＬＡＭグレーダーにセットするマスクワードを作成する。続いて各モジュールに対して典型的なＣＡＭＡＣコマンドをＭＢＤのチャンネル０を起動することによって実行し，ＸレスポンスがなければエラーテーブルにＣＮをセットする。（このチェックはコインシデンスユニットとＡＤＣインターフェースに対しては行われず，マイクロプログラムのEDITORによってセットされるイニシャライズコマンドを実行する際に行われる。他のモジュールに対しても同じ方法をとるようEDITORを書きかえる予定である。）次にサブルーチンRDTBLを呼びリデュースドスペクトルを作成する際使用するテーブルREDINFを作成する。Ｘーチェックでエラーがあった場合はエラーテーブルを共有メモリーにロードし，COMM０４はタスクPRINTを通じてＣＮをコンソールに出力する。

(3) COMM０４は，マイクロプログラムをＭＢＤのメモリーにロードするためのプログラムLOADERを共有メモリーにロードし，ATOMICはこれを７０００　番地以降に入れる。

(4) 次に必要とされるマイクロプログラムが１つずつ送られる。（マイクロプログラムの種類については第4.２節参照） まずマイクロプログラムとペアーになっているEDITORが共有メモリーにロードされ，ATOMICによって６００００番地以降に移される。次にCOMM０４はマイクロプログラムを４００００番地からロードする。ここでロードされるEDITORとマイクロプログラムは“DATA ACQUISITION”の画面で指定されたプログラム番号のものである。ATOMICはEDITORを起動し，EDITORはマイクロプログラム中のCAMAC コマンドに必要なＣＮをセットしたり，ＭＢＤのファイルレジスターにロードするデータのテーブルＦＲＥＧにデータを入れたりする。EDITOR中にはCAMACをイニシャライズするためのコマンドがセットされているが，ATOMICはこれを取出して，５００００番地以降に１つのテーブルとしてまとめる。続いてLOADERが起動され，マイクロプログラムをＭＢＤにロードする。この際のロード番地は COMM０４によって各マイクロプログラムの前につけられてい

る。

(5) COMM04は全情報の終了を送信し，ATOMICはこれを受けてデータ収集開始のための準備を行う。

まずLAMグレーダー，クロック，プリセットスケーラー，インターラプトレジスター等からのインターラプトおよびリストモードのマイクロプログラムの“BUFFER-FULL”のインターラプトに対するベクターがセットされ，次にCAMAC に対するイニシャライズが次のように行われる。

- MBDリセット，CAMACにBZ送出。
- MBDのマスクレジスターに(2)で作成されたマスクをロード。
- 各クレートコントローラーのINHIBITを解除，BDをイネィブル。
- LAMグレーダに(2)で作成したマスクをロード。
- スケーラー等のイニシャライズ（(4)でのべたようにこれらはEDITOR中にテーブルをセットするよう書きかえる予定）。
- (4)でロードされたマイクロプログラムMBDINTを起動，すべてのチャンネルのファイルレジスターにFREGに入っている値をロードし，各チャンネルがCAMACから対応するGLによって起動されるようにする。
- (4)で50000番地以降に作られたイニシャライズのためのCAMACコマンドをチャンネル0を起動することによって次々と実行する。Xレスポンスがなければ TTYにCNを出力する。

以上の終了後COMM04に“RUN ENABLE”を送信する。

(6) COMM04は“RUN ENABLE”を受けとると，イベントフラグ55により終了をINITに通知しEXITする。この間Xチェック以外でエラーがあれば更にイベントフラグ33をセットしてエラーのあったことをINITに知らせる。INITはエラーがなければ RUN ENABLE と 11/04-ACTIVE のランプをオンにし，タスクDATACQを起動してEXITする。エラーがあった場合は“CAMAC CONFIGURATION”の画面の表示にもどる。

なお INIT END の際DATACQがRUN中の時は，INIT（リストダンプの磁気テープを使用している時）またはCOMM04は磁気テープとDR11-Cの解放（DETACH）をDATCQに要求する。DATACQはこれを受けてEXITする。この際それまで使用していた磁気テープが使用中止となっていればE-O-Vが磁気テープに書かれる。

COMM04によるATOMICとの交信の際すべての情報が送り出されるのは次の場合である。

- INITの開始時のTAG名入力要究に対して，“C. .”以外を入力した時。
- CAMACの構成，使用するADC番号，ADCのビット長，マイクロプログラムの番号，リストバッファーサイズのどれかに変更があった時（使用をやめるADCがあるだけの時は除く）。
- 11/04-ACTIVE のランプがオフの時。

- プリセット値が変更された時 （ PRESET による変更は無関係。）
- 0でないマイクロプログラム番号がある時。

下記の変更しかなかった時は必要な情報だけが送られる。

- リストモード，1パラメーターPHA，2パラメーターPHAのどれかがとりやめになった時。
- リデュースドスペクトルに変更があった時。
- MEMORY MAPに変更があった時。

3.3 データ収集の動作

オペコンの RUN ENABLE ランプがオンになった後，オペコンあるいはCAMACからの要求によってデータ収集がスタートする。データ収集はCAMACからのＬＡＭ（GL=17−24）によってマイクロプログラムが起動され，CAMACからのデータを共有メモリーに書き込むことによって行われる。“STOP”中も含めてデータ収集中のATOMICからの通信はすべてDATACQが受ける。例外的に MEMORY ADVANCE の場合にだけタスクMEDORYが直接ATOMICからの応答を受ける。この間DATACQはDR11−CをDETACHする。ＡＴＯＭＩＣに対する送信はOPECONによっても行われる。

リストモードの収集を行わない時はATOMICもDATACQ も定期的にスケーラーのデータを送受信するだけである。

リストモードの場合はリストバッファーが一杯になるとリストモードのマイクロプログラムからATOMICに対してインターラプトが出される。この際マイクロプログラムのそのバッファーに対応する“BUFFER−FULL”のフラグをセットし，次のバッファーに切換える。11／55で次のバッファーの処理が終了していない時は，マイクロプログラムはコインシデンスユニットのＬＡＭを禁止してリストモードの収集を中断する。インターラプトによってATOMICはDATACQに“BUFFER−FULL”を送信し，リデュースドスペクトルの作成が指定されていれば作成を開始する。この時前のバッファーに対するリデュースドスペクトルの作成は中止され，処理残りのイベント数がスペクトル毎の“LOST DATA”に加算される。コンソールに出力される“LIST−OVER RUN”，“REDUCED−OVER RUN”で示される数字は上記のリストモード中断，リデュースドスペクトル作成中止の回数である。DATACQは“BUFFER−FULL”の送信を受け,一杯になったバッファーの内容を別のバッファーエリア（リストバッファーに対応して２個ある）にコピーし磁気テープに対するダンプを開始する。この際同じ側のバッファーに対する前回のダンプが終了していなければ終了を待ってからコピーが行われる。コピーが終了するとATOMICに対して“BUFFER−UNLOAD”を送信する。（磁気テープの指定がなければコピーは行わず，直ちに“UNLOAD”を返す。）ここで11／55で作成するリデュースドスペクトルが指定されている場合にはDATACQは“SEND−DATA”ディレクティブによってRDSP55に対して“BUFFER−FULL“を送信する。RDSP55はこれを受けてリデュースドスペクトルの作成を開始する。この時ATOMICの場合と同様に前のバッファーの処理は中断され“LOST DATA”に処理残りのイベント数

が加算される。

なおRDSP55はディスプレィのためのタスクDSUP，DSCONと同じくシステム中最低のプライオリティで動く。したがってプライオティは高いタスクが頻繁に動く場合（例えばリストモードの入力レードが高くて，磁気テープの指定がない場合はほとんどのCPU時間がATOMICとの交信に使われる）にはRDSP55に割当てられるCPU時間は少くなり，場合によっては全くなくなることもある。DATACQは，送った“SEND-DATA”がRDSP55に受けとられずある回数以上たまった時には，それ以上“SEND-DATA”を行わず自分で“LOST DATA”の加算を行う。またRDSP55において“SEND-DATAによるAST*)”が禁止されている場合も同様にする。

ATOMICはDATACQから“BUFFER-UNLOAD”を受けるとそのバッファーに対する“BUFFER-FULL”のフラグをリセットする。ここでもしコインシデンスユニットのLAMが禁止されていれば，START／STOPのマイクロプログラムを起動してコインシデンスユニットのLAMをイネィブルしリストモードの収集を再開する。

START中はクロックからのLAMによって，ATOMICはスケーラーを読むマイクロプログラムを起動し定期的に（現在は1秒毎）スケーラーの値を11／55に転送する。この転送はプリセットスケーラーがプリセット値に達して収集がSTOPする時も行われる。

---

*) アシンクロナス・システムトラップ　ソウトウェア的なタスクに対するインターラプト。

4. マイクロプログラムの作成

この章ではマイクロプログラムを作成することによるデータの取込み方に対する変更についてのべる。すでにのべたように，この変更はリストモードおよびＰＨＡモードの形式内のものであるが，この形式を守りさえすればデータ取込みの際に何らかの条件を判定して，実験装置に対して何らかのコントロールのコマンドを出したりすることは可能であり，極端な場合には，データ収集のためのマイクロプログラムで全くデータ収集を行わず，コントロールだけを行うような変則的な使用をすることもプログラム作成者の自由である。

4.1 データ収集モードの定義

ここでおのおののデータ収集モードに対する形式上の制約をのべる。マイクロプログラムの作り方に対する制約については第4.2－5節でのべる。

(1) リストモード

コインシデンスユニットからのＬＡＭによって起動されるマイクロプログラムＬＩＳＴによって，リストバッファーにデータを取込む。ここでコインシデンスユニットは現在用意されている標準的なモジュールである必要はなく，ＬＡＭを発生する機能が最低限必要なだけである。（この点は以下にのべるＡＤＣインターフェースについても同様である。）一般には１つのＬＡＭによって代表されるイベント毎に，“DATA ACQUISITION”の画面で入力されたワード数のデータをリストバッファーに書き込む。一方のバッファーが一杯になった時１１／０４に対してインターラプトを発生し，バッファーを切換える。このインターラプトによって必要に応じて磁気テープのバッファーのダンプとリデュースドスペクトル作成のサービスが受けられる。どのようなデータをどのような形で取込むかは任意である。例えば２個のデータの和をとるとか，２個のデータを１ワードにパックするとか，上位ビットを何らかのタグに使用する等が可能であるし，何らかの条件によってデータを捨ててもよい。

［ＩＮＩＴ］時の制約としてはＡＤＣインターフェース＃１－８中の少くとも２個を使用するよう指定する必要があるが，必要がなければこれらからデータを読まなくてもよい。

(2) ２パラメーターＰＨＡ

コインシデンスユニットからのＬＡＭによりマイクロプログラムＰＨＡ２が起動され，共有メモリー内の“MEMORY MAP”の画面で示される領域内のデータに必要な変更を加える。領域の大きさは［ＩＮＩＴ］時に指定される２台のＡＤＣのビット数とトランケートビット数できまり，一般にはこの領域外のデータに変更を加えてはならない。これらのビット数と実際に取込むデータとの関連にはこれ以上の制約はない。

(3) １パラメーターＰＨＡ

指定されたＡＤＣインターフェースからのＬＡＭによりマイクロプログラムＰＨＡ１が起動されデータの取込みを行う。スペクトル領域の大きさがＡＤＣのビット長できまる以外は２パラメーターＰＨＡと基本的な差はない。

4.2 マイクロプログラムの種類

INIT の段階でCOMM04により11／04に送られてくるマイクロプログラムには次のものがある。

(1) MBDINT

起　動　　：　ATOMIC
チャンネル　：　0－7
プログラムコード：　0

ここでプログラムコードは各マイクロプログラムとEDITORの先頭につけられているプログラムの種類を区別する番号である。

2つの異った目的で起動される。第1の場合は INIT 時および MEMORY ADVANCE 時であり，MBDのファイルレジスターに初期値を入れる。この際各チャンネルのCEL（チャンネルイネィブルラッチ）がセットされCAMACからのGLによって対応するチャンネルのプログラムが起動されるようになる。第2の場合はリストモードのデータ収集をやめる場合で，リストバッファーのどこまでデータが入っているかを知るために起動される。この際リストバッファーの残りの部分には－1が入れられる。この2つの場合はPDRがポイントする番地に入れられているコードによって区別される。マイクロプログラムの終了は11／04にインターラプトを行うことでATOMICに知らされる。

プログラムはMBDのメモリーのはじめからロードされ，最初の部分は次のような構成になっている。

番　地	命　令
0	JVC
1	JP　3（データ収集時はBK1）
2	JVC
3	CON　BK0

0番地の“JVC”はCAMACからのGLにより対応するチャンネルのプログラムへジャンプするためのものであり，3番地以後がMBDINTの実験のプログラムになっている。1番地の“JP　3”命令は　MBDINTが終了した後“BK1”命令に書きかえられる。この変更と2番地の“JVC”命令によって，以後ATOMICによってチャンネルが起動された時そのチャンネルのバンク1のCTRに入っている番地からプログラムが実行されるようになる。

(2) STTSTP

起　動　　：　ATOMIC
チャンネル　：　7
プログラムコード：　1

データ収集のスタート・ストップを行うプログラムである。ATOMICによってセットされるPDRの値にしたがって下のような処理を行う。プログラムの終了はATOMIC中のステー

タスワードSTATUSを1にセットすることによってATOMICに通知される。

PDR＝1　オペコンまたはCAMACからスタート入力があった場合であり，次のようなスタートの動作を行う。ゲートモジュールの出力端子0－5の出力をロジックレベル“1”にし，出力端子6からパルス出力を行う。

PDR＝2　第3.3節でのべたように，リストモードの収集の際，両方のバッファーが一杯になると，コインシデンスユニットのLAMは禁止される。その後DATACQ から“BUFFER−UNLOAD”の通信を受けた時STTSTPが起動されLAMの禁止を解除する。

PDR＝3　CAMACへのSTOP入力，またはプリセットスケーラーがプリセット値に達した時発生するGLによるインターラプトを受けて起動され，次のようなストップ動作を行う。ゲートモジュールの出力端子0－5の出力をロジックレベル“0”にし，端子7からパルス出力を行う。

PDR＝4　オペコンから START オフの入力があった時に起動される。動作はPDR=3と同じである。

標準的でないモジュールを使用してデータ収集を行う場合はこのプログラムを変更する必要が生ずる場合がある。新しいモジュールを設計する時はゲート入力がない場合にはLAMが禁止されるようにしておくと変更が不要の場合が多い。

(3)　SCLTIM

起　動　：　ATOMIC

チャンネル　：　1

プログラムコード：　2

スケーラー（プリセットスケーラー，ライブタイムスケーラー，12チャンネルスケーラー）に関するプログラムである。PDRに処理のためのパラメーターブロックの番地が入れられており，その第1ワードが処理の種類を示すコードになっている。コードにより次のように処理が異なる。プログラムの終了はSTATUSを1にセットすることで示される。

コード　1　クロックからのGLによるインターラプトが生じた時，またはプリセットに達してデータ収集ストップの動作が行われた後起動され，スケーラーのカウントを読みATOMIC中のPS11以下に書き込む。PS11 の番地はパラメーターブロックの第2ワードに入っている。

コード　2　オペコンから SCALER CLEAR の入力があった時起動される。プリセットスケーラーに対してはパラメーターブロックの第2ワード以後に入っているプリセット値をダウンカウンターにロードし，他のスケーラーはクリアーする。

コード　3　PRESET によってプリセット値を変更した場合に起動され，パラメーターブロックの第2ワード以下に入れられているプリセット値の変化分を，ダウンカウンターに加える。

(4) ＧＬＣＬＲ

起　動　　：　ＡＴＯＭＩＣ

チャンネル　：　６

プログラムコード：　６

クロック，プリセットスケーラー，ＬＡＭグレーダー，インターラプトレジスター等からＧＬによるインターラプトが発生した時ＬＡＭをクリアーするプログラムである。

(5) ＰＨＡ１

起　動　　：　ＣＡＭＡＣからのＧＬ

チャンネル　：　０－７

プログラムコード：　３

ＡＤＣインターフェースからのＧＬにより起動され，１パラメーターＰＨＡのデータ収集を行うプログラムである。１パラメーターＰＨＡの個数に対応して，一般には複数個のプログラムがＭＢＤにロードされる。

(6) ＰＨＡ２

起　動　　：　ＣＡＭＡＣからのＧＬ

チャンネル　：　０－７

プログラムコード：　４

コインシデンスユニットからのＧＬにより起動され，２パラメーターＰＨＡのデータ収集を行うプログラムである。

(7) ＬＩＳＴ

起　動　　：　ＣＡＭＡＣからのＧＬ

チャンネル　：　０－７

プログラムコード：　５

コインシデンスユニットからのＧＬにより起動され，リストモードのデータ収集を行うプログラムである。現在のＡＴＯＭＩＣではデータレートが高い時，クロックのインターラプトとかち合ってハングアップする可能性があるので，最低のプライオリティーのチャンネル０で動かした方がよい。

以上のプログラムは上にあげた順で１１／５５で送られてくる。ただしデータ収集のプログラムはその収集モードが指定された時だけ送られる。各プログラムの名前はファイル名でもある。

(8) ＣＡＭＡＣコマンド実行のためのプログラム

起　動　：　ＡＴＯＭＩＣ

チャンネル：　０

このプログラムは先にあげた７個のプログラムと異なり単独のファイルとしては存在せず，

ATOMIC中に入れられており，第3.2節でのべたように，ATOMICの起動時にMBDにロードされる。CAMACイニシャライズのコマンドや，TTY入力によるCAMACコマンド(第7.3節参照)の実行のために使われる。プログラムの終了はSTATUSを1にセットすることによって示される。

CAMACコマンドを実行するためにはマクロPFCNAを使用して

PFCNA　F,C,N,A,〔DH,DL〕

でコマンドのパラメーターブロックを作り，その番地をPDXに入れてチャンネル0を起動すればよい。ここでFCNAは数字で入力する時は10進表示で入れる。DH，DLはデータの上位8ビットおよび下位16ビットであり，データを必要としないコマンドでは入れなくてもよい。(この場合は0にイニシャライズされる。)

PFCNAで作られるパラメーターブロックは4ワードで次の構造になっている。

FCNA	
BC0／1	DH
DL	
Q	X

ここでFCNAはBARのビット構成で入れられ，除かれたF8のビットは次のBC0／1によって表される。READコマンドの場合にはデータがDH，DLに入れられる。コマンド実行時のQ，Xが4ワード目に入る。

4.3　マイクロプログラムとEDITORの構造

すでにのべたようにマイクロプログラムにはEDITORがペアーになっている。この節では附録1，2に示すリストモードのプログラムLISTを例にとって構造と役割りを説明する。

4.3.1　EDITOR

EDITORのソースファイルは

LIST.MAC

であり，ロードされるバイナリーファイルは

LIST.EDn

である。ここでnは INIT 時に指定するプログラム番号(0－9)であり，標準のプログラムでは0である。

EDITORの主な役割りは，INIT 時に設定されたデータをもとにしてマイクロプログラム中データの可変部分やファイルレジスターにロードする値のテーブルにデータを書き込むこと，およびCAMACイニシャライズのためのコマンドのテーブルをセットすることである。必要に応じてTTYにメッセージを出力したり，INIT で指定できないデータをTTYから入力することも可能である。(TTYを使用するためのサブルーチンに関しては第7章に

のべてある。）

附録１の①（以下１－①と書く）に示されているように“. ENABL ABS”を指定して絶対形式でアンブルされる。（RSX−11Mの標準の形式ではない。）マクロライブラリーINSTBD. MLBはＭＢＤの命令をアセンブルするためのものであり，ATOMIC. MLBはEDITORがプログラムATOMICのサブルーチンとして動作するために必要なリンクを行うためである。 ＩＮＩＴ 時に設定されCOMM０４によってATOMICに送られたデータもATOMIC. MLBによってEDITOR中で使用できるようになっている。

プログラムの最初の部分（１－②）でまずロケーションカウターが６００００にセットされる。第１ワードにはEDITORのワードサイズ（EDITOR の後半に入れられているマイクロプログラムのダミーは含まない。），第２ワードにはロード番地が，第３ワードには第4.２節でのべたプログラムコードが入れられる。実際には第３ワード以後が６０００４番地以降にロードされる。第４ワードにCAMACイニシャライズのためのコマンドブロックの先頭番地を入れる。イニシャライズコマンドがない場合には－１を入れればよい。コマンドは１－⑧のようにPFCNA を使用してセットすればよい。コマンドブロックの最後にデリミッターとして－１を入れる。ここではCNが入れられておらずEDITORによって１－⑦で使用するモジュールのＣＮが入れられる。しかしいつも固定のＣＮで使用するモジュールの場合にはＰＦＣＮＡで直接ＣＮを入れてよい。

一般にEDITORの後半には１－⑨以下に見られるようにマイクロプログラムのダミーコードが入れられている。マイクロプログラムはこのダミーコードと同じ形で４００００番地からロードされる。このダミーコードはEDITORがマイクロプログラムにデータを書き込む際のアドレスのリンクのために入れられており，マイクロプログラムを変更しない場合は入れなくてもよい。ダミーマイクロプログラムの先頭には３ワードの情報が入っている。（１－⑨）これらはCOMM０４によって共有メモリーに入れられる情報である。第１ワードがマイクロプログムのワードサイズ，第２ワードがＭＢＤのコントロールメモリーのロード番地，第３ワードの下位バイトにはプログラムコード，上位バイトには ＩＮＩＴ 時に指定したプログラム番号が入れられる。ＰＨＡ１の場合にはいくつものプログラムがロードされる可能性があるので対応するパラメーター番号が第３ワードの最上位４ビットに入れられる。

なお現在COMM０４がEDITORをロードするやり方では “. BLKW”，“. BLKB”，“RSWD”は正しくロードされないので使用してはならない。

4.3.2 マイクロプログラム

ソースファイルは

LIST. MPR

バイナリーファイルは

LIST. MBn

でｎはプログラム番号であり，ペアーになっているEDITORと同じでなければならない。

マイクロプログラムはアセンブルする段階ではＭＢＤのロード番地がきまらないので，０番地からロードされる形でアセンブルし，ロード番地がきまった段階でプログラム中で使用され

ている番地を修正する。この手続きはCOMM04がマイクロプログラムを 共有メモリーにロードする際サブルーチンMBDLNKによって行われる。マクロライブラリーBDINST. MLBはこの操作が可能な形にマイクロプログラムをアセンブルするために使用される,EDITORとちい". ENABL ABS"を指定してはならない。

マイクロプログラムのロード番地を決定する際，プログラムが２５６ワード単位のページの境界を越える時は次のページの先頭からロードするようにしている。これはブランチ命令がページ内でのみ有効であることを考慮したものである。プログラムのサイズが２５６ワードをこえる時はブランチがページ内に入っているように注意する必要がある。

4.4 アセンブルの手続き

標準的なマイクロプログラムとEDITORのソースは１１／７０システムディスクのU. I. C. 〔１０，５４〕に入っている。（コマンド@〔１０，５４〕LSTMPR によってすべてのリストがとれる。）

ここでもリストモードのプログラムを例にとり，プログラム番号３のプログラムを作成することにする。最終的には１１／５５のDK1に

〔10，14〕LIST. ED3，LIST. MB3

を作成すればよい。

変更は一般に11／70で行う。まず〔１０，５４〕に入っているソースを自分のU. I. C. へ例えばＬＩＳＴＳＰの名前でコピーする。これに変更を加え（コピーせずに全く新しいものを作成することは自由である。）

```
MAC  LIST. ED3=〔1，1〕INSTBD／ML，ATOMIC／ML，〔uic〕LISTSP
MAC  LIST. MB3=〔1，1〕BDINST／ML，ATOMIC／ML，〔uic〕
                    LISTSP. MPR
```

によってアセンブルする。作成したバイナリーファイルを１１／５５のDK1に入れるディスクの自分のU. I. C.に移し（あるいは直接作成する），ディスクを１１／５５にロードしＰＩＰで〔10，14〕へ移す。（１１／７０ではファイル保護機能のため直接〔１０，１４〕へは入れられない。）

当然のことであるが自分が管理していないディスクへの書き込みは管理している人の了解を得て慎重に行う必要がある。特に同じプログラム番号で異ったプログラムを登録すると混乱するので注意を要する。

4.5 マイクロプログラムの例

ここで実際のマイクロプログラムの作成あるいは変更を容易にするために，例として標準のいくつかのマイクロプログラムについてよりくわしくのべる。またデータ収集のマイクロプログラムではファイルレジスター中の特定のものをきまった目的に使用しなければならないものがあるので，それについてもふれる。

4.5.1　ＬＩＳＴ

リストモードのEDITORのリストを附録１に，マイクロプログラムを附録２に示す。

ファイルレジスターは次のように使われている。

ＤＡＲ：　バッファー０の先頭番地

ＣＣＲ：　バッファー１の先頭番地

ＩＬＲ：　バッファー０の実際にデータが入る最後の番地

ＷＣＲ：　バッファー１の実際にデータが入る最後の番地

ＧＰ２：　次にデータを入れるバッファーの番地

ＧＰ１：　１

以上のうちＧＰ２以外は異った使い方をしてもよい。なおここで使用される番地はすべてワード単位の番地である。附録１の④（以下１－④と書く）でこれらに対する初期値がテーブルＦＲＥＧにセットされている。

ＡＤＣからのデータは“DATA ACQUISITION”の画面で指定した順（パラメーター番号順）に読まれ，イベント毎（ｎパラメーターのリストモードならｎ個のデータで１イベント）にデリミッターを１ワード入れ，ｎ＋１語ずつバッファーに入れられる。

LIST BUFFER CLEAR によってバッファー全体に－１が入れられるのでマイクロプログラムでは１ワードスキップすることによって－１のデリミッターをつけている。１つのイベントがバッファーに入りきらない時は途中まで入れることはせず次のバッファーに切換える。

マイクロプログラムにおいてはＬＩＳＴ０からはじまる部分とＬＩＳＴ１以下の部分がそれぞれバッファー０と１に対するプログラムであり，処理スピードを多少上げるためにほとんど同じコードをくり返してある。２－①でCAMACからデータを読みＡＤＣのビット長に対応するマスクとＡＮＤをとりリストバッファーに書き込んでいる。このマスクはリデュースドスペクトルを作成する際に割当てられたメモリー領域以外に書き込まらなように入れられたものである。現在では一般にリデュースドスペクトル作成の際にチェックを行っているのでこのマスク必ずしも必要ではない。データを読むCAMACコマンドとマスクは２－⑤に入れられており，EDITORによって１－⑤⑥でセットされる。なお２－⑤には３２個までのＡＤＣのＲＥＡＤコマンドが入れられるようになっている。ｎパラメーターの時はｎ＋１番目のコマンドを０にしてデリミッターとして使用している。２－②ではバッファーが一杯になったかどうか判定し一杯でなければ単にコインシデンスユニットのＬＡＭをクリアー（Ｆ１０）してＥＸＩＴする。一杯になった時は２－③でBFLG0＝1と“BUFFER－0－FULL”のフラグをセットする。次にBFLG1をチェックして次のバッファーであるバッファー１が一杯（まだ11／55で処理されていない）かどうかをチェックし，一杯であればコインシデンスユニットのＬＡＭを禁止（Ｆ２４）して，禁止したことをフラグCLDSBLを１にセットしてATOMICに知らせる。続いて２－④でＬＡＭをクリアーしＧＰ２に入っているバッファーの番地を次のバッファーの先頭番地に書きかえて１１／０４に対してインターラプトを出してＥＸＩＴする。この際前のインターラプトがまだ受けつけられていず，なおかつ他のチャンネルからのデマンドがあればＥＸ１によってプログラムを一旦中断しより高いプライオリティーのチャンネルの実行を優先

させる。

イニシャライズ時には1－⑧に入っているコマンドが実行される。ここではコインシデンスユニットに対してはＬＡＭをイネィブルし(Ｆ26)し，ＬＡＭをクリアー(Ｆ10)している。またＡＤＣインターフェースに対してはＬＡＭを禁止(Ｆ24Ａ0)し，クリアー信号の送出をイネィブル(Ｆ26Ａ1)している。このようにイニシャライズしてもＧＡＴＥモジュールがリセットされており，コインシデンスユニットもＡＤＣインターフェースも START オンまでデータを受けつけることはない。

4.5.2　PHA1，PHA2

1パラメーターＰＨＡのためのＥＤＩＴＯＲのリストを附録3に示す。(マイクロプログラムはＥＤＩＴＯＲ中にコピーがあるので別に示さない。)

ＰＨＡ1ではファイルレジスターは次のようにイニシャライズされる。

ＤＡＲ：　スペクトル領域の先頭番地

ＩＬＲ：　オーバーフロー領域の先頭番地

ＣＣＲ：　ADC−READのCAMACコマンド

ＷＣＲ：　3

これらのうちＤＡＲとＩＬＲは MEMORY ADVANCE の機能と両立させるために上のように使用しなければならないが，他のファイルレジスターの使い方はプログラム作成者の自由である。

附録3の④(以下④と書く)の上位バイトの上位4ビットにはCOMM04によってパラメーター番号が入れられている。①ではこの番号から使用するＡＤＣインターフェースの番号を求め，以後ファイルレジスターの初期値テーブルＦＲＥＧや，ＡＤＣインターフェースのＣＮ等を参照する際に使用している。②では共有メモリーの割当てられたスペクトル領域の範囲外にデータを書き込まないように，ＩＮＩＴ で指定したＡＤＣのビット長に相当するマスクをマイクロプログラム中のＭＡＳＫにセットしている。イニシャライズのためにＬＡＭとＡＤＣのクリアーをイネィブルするコマンドＦ26Ａ0とＦ26Ａ1が③に入れられている。

⑤以下がマイクロプログラムである。⑤ではＡＤＣインターフェースからデータを読み，ＭＡＳＫとＡＮＤをとり，データーに対応するスペクトル領域をインクリメントしてＥＸＩＴする。ＡＤＣインターフェースはREAD はコマンドＦ0でＬＡＭのクリアー，ＡＤＣのクリアー，ＡＤＣインターフェースのデータレジスターのクリアーのすべて行われるように設計されているのでここではＦ0を実行するだけでよい。ただしインクリメントの結果オーバーフローが生じた場合には⑥以下で対応するオーバーフロー領域をイメクリメントする。オーバーフロー領域がインターラプトステータスワードと重っている場合には11／04のローカルメモー中のOVFLOW以下の2ワードを使用する。

2パラメーターＰＨＡのためのプログラムＰＨＡ2はＰＨＡ1と同様のプログラムであり，ＬＡＭがコインシデンスユニットから発生するのでコインシデンスユニットのＬＡＭをクリアーすることと，2台のＡＤＣからのデータ N_x, N_y から

$$N = [N_y \cdot 2^{-TRY}] \cdot 2^{B_x - TRX} + N_x \cdot 2^{-TRX}$$

によって実効チャンネル数を計算して対応する領域をインクリメントすることが異っている。

ファイルレジスターの使用上の制限は

ＤＡＲ：　スペクトル領域の先頭番地

ＷＣＲ：　オーバーフロー領域の先頭番地

だけである。

5. リデュースドスペクトルの作成

ここではリストバッファーが一杯になった時に開始されるリデュースドスペクトル（R. S. と省略する）の作成についてのべる。R. S. は当初リストモードのデータ収集のモニターのために設けられたが，データレートがあまり高くない場合には最終的なデータ収集の手段として使用でき単純なＰＨＡモードのデータ収集にくらべて柔軟性に富むデータ収集が可能になることに注目してよい。プログラミングは第８章にのべるデータの構成さえ理解していればきわめて容易で実験に合せてその都度プログラムを書きかえて使用できる。したがって実験によってはリストモードであっても磁気テープへリストバッファーをダンプすることは不要になる。

すでにのべたようにR. S. は主として１１／０４で作成されるが，１１／５５で作成することもできる。１１／０４のプログラムRDSP０４と１１／５５のタスクRDSP５５は，RDSP０４がATOMICのデータを作用するのに対して，RDSP５５は１１／５５のシステムのCOMMONであるCOMTBLを参照する点で異なるが，多くの場合同じ名前を使用しておりプログラムのステートメントもほとんど同じである。したがってここでは主としてRDSP０４についてくわしくのべる。

１１／０４のプログラムATOMICはメインプログラムATMMANとR. S. に関する部分RDSP０４から成っており，それぞれ別々のファイルになっている。RDSP０４のリストを附録４に示す。RDSP０４はRDTBLとREDSPCの２つのサブルーチンから成っており，RDTBLは INIT 時にセットされたデータからテーブルREDINFを作成し，REDSPCはバッファーが一杯になった時起動され，REDINFを使用してR. S. を作成する。REDINFは各R. S. 毎に作成され，REDSPCがR. S. を能率よく作成できるように構成される。

附録４の①（以下①と省略）に見られるようにRDSP０４もマクロライブラリーATOMIC. MLB によってATMMANとリンクされる。②⑭ にあるようにRDTBLとREDSPCはそれぞれ番地ARDTBLとARDSPCからはじまる。現在ARDTBL＝３３０００，ARDSPC＝３５４００ととってある。（ATMMAN およびATOMIC. MLB中で定義，必要に応じて変えられる。）１１／０４のプログラムエリアはローカルメモリーだけであり３７７７７番地までなのでこのはんい内でプログラムを作成しなければならない。必要に応じて使用しないR. S. の部分を削除すればよい。

R. S. は INIT 時に指定されるTYPE によって種類が識別される。RDTBLとREDSPC にはそれぞれのＴＹＰＥ に対する処理ルーチンがある。ＴＹＰＥ は０－４９がRDSP０４に５０－９９がRDSP５５に割当てられている。０および５０が標準的なＴＹＰＥである。RDTBLでは⑧で，REDSPCでは⑮でＴＹＰＥにしたがって対応する処理ルーチンへブランチしている。新しいR. S. を追加するためにはこのあとに処理ルーチンを追加すればよい。

REDINFは次のように第１ワードにR. S. の個数が入れられ，以下各R. S. に対するテーブルがデミリッター177777_8（＝－１）でくぎられて並べられている。各R. S. に対するテーブルのサイズは一定ではない。

REDINF：

R. S. の個数
R. S. #1に対するテーブル
－1
R. S. #2に対するテーブル
－1

1パラメーターのTYPE＝0のR. S. のREDINFは次の通りである。

オフセット	内　　容
0	0（＝TYPE＊4）
2	次のR. S. に対するREDINFの番地
4	“LOST DATA”の番地
6	バッファー中の処理すべきイベント数
8	次に処理するイベントの番地
10	R. S. を作成するパラメーターの番号＊2（0, 2……）
12	THRESHOLD
14	TRL
16	最大チャンネル数
18	REDCHNテーブルの番地
20	スペクトル領域の先頭番地
22	GATEのパラメーター番号＊2（0, 2……）
24	OFFSET（GATEの下限）
26	LIMIT（GATEの上限）

以下指定されたGATEの個数だけ3ワードのブロックを入れる。

これらのうちオフセット0－8の5ワードは2パラメーターのR. S. も含めてすべての場合同じ形をとる。最後のR. S. に対してはオフセット2は0となる。オフセット6，8はバッファーが一杯になった時にセットされる。

RDTBLでは⑧まででオフセット10までがセットされているので，⑨以下でオフセット12以降を作成すればよい。オフセット10の“パラメーター番号＊2”は必要に応じて変更してもよい。なおプログラム中で使用するパラメーター番号は INIT の画面のパラメーター番号より1小さく0から始まる番号である。

“最大チャンネル数”はADCのビット数をBとした時

$$B_{RS} = B - TRL - TRU$$

によって計算される有効ビット長に対応して

$2**B_{RS}-1$

によって与えられる。

TRL，TRU，THRESHOLDは INIT の画面の最初のＧＡＴＥのラインのものが使用される。１６ワードのテーブルMAXCHNには

$$MAXCHN(n)=2^{n}-1 \qquad ;n=1-16$$

が入っているので利用できる。

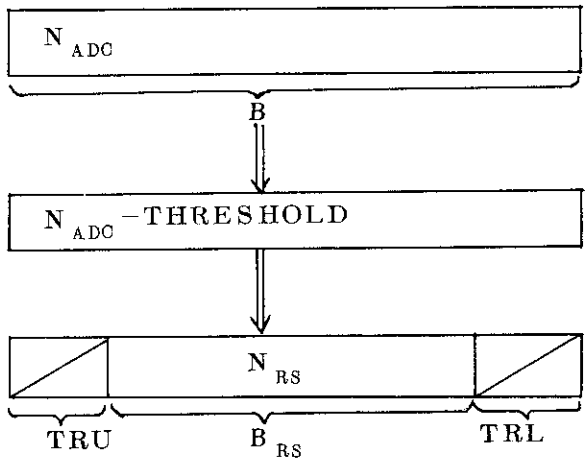
すべてのR. S.に対してスペクトル領域はB_{RS}できまるサイズが割当てられているので，一般にはこの範囲内でスペクトルを作成しなければならない。上の場合の最大チャンネル数はこの目的で使用される。

REDCHNはR. S.の個数のワード数を持つテーブルで，各R. S.毎に計算されたR. S.のチャンネル数N_{RS}の２倍を入れる。第5.1.3節でのべるTYPE＝３のR. S.ではREDCHNのテーブルをリストバッファーと同様にあつかってR. S.同志の間でＧＡＴＥをかけることが可能になっている。オフセット１８にはテーブル中のそのR. S.に割当てられた番地が入れられる。

なおREDCHNはイベント毎に書き直されるので，R. S.を作る際ＧＡＴＥからはずれたイベントに対しては対応するREDCHNをクリアーしておき，TYPE＝３のR. S.では０チャンネルをはずしてＧＡＴＥをセットするようにする。

ＧＡＴＥは使用するものだけがREDINFにセットされる。“DATA ACQUISITION”の画面でＧＡＴＥのパラメーター番号を０（内部データは177_8）としたラインおよび全チャンネルを含むＧＡＴＥは⑥でチェックされテーブルから除かれる。

TYPE＝０の１パラメーターR. S.ではN_{RS}はＡＤＣからのチャンネル数N_{ADC}から次のように作られる。



ＴＲＬのビットは切捨てられ，上位のＴＲＵビットがすべて０でなければスペクトルには加えられない。したがってＡＤＣに入るスペクトルのTHRESHOLD 以上の必要な部分だけを必要な分解能でモニターできる。（リストダンプの磁気テープにはＡＤＣからの生のデータが入る。）

2パラメーターのTYPE=0のREDINFは次のようになっている。

オフセット	内　　容
0	2（TYPE*4+2）
2	次のR. S.のREDINFの番地
4	“LOST DATA”の番地
6	処理すべきイベント数
8	次に処理するイベントの番地
10	パラメーター・Xの番号*2（0，2……）
12	パラメーター・Yの番号*2
14	TRX
16	TRY
18	BX−TRX（BX：パラメーター・Xのビット数）
20	スペクトル領域の先頭番地
22	GATEのパラメーター番号*2
24	OFFSET
26	LIMIT
	以下セットされているGATEの個数だけくり返し。

RDTBLでは⑧においてオフセット18までがセットされている。守らなければならない形式はオフセット0−8までなので必要に応じてオフセット10以後は変えてよい。

5.1 リデュースドスペクトルの例

ここではいくつかのR. S.の例について説明を行い，実際にプログラムを書く時の参考にする。なお先にのべたようにR. S.は必要に応じて書き直されたり削除されたりするので，ここにあげる例もいつもこの形でディスクに入っているとは限らない。したがって実際の使用にあたってはその時点での使用マニュアルを参照する必要がある。

5.1.1 荷電分割法による位置スペクトル

1パラメーターのTYPE=1のR. S.として入れられており，パラメーター番号P_1，P_2の2台のADCからのデータN_1，N_2から

$$N_{RS}=(N_2\cdot 2^{B_{RS}})/(N_1+N_2)$$

$$B_{RS}=B_1-TRL-TRU$$

によってスペクトルを作成する。ここでB_1はN_1のビット長である。P_1は“DATA ACQUISITION”の画面でR. S.のパラメーター番号PRMの欄で指定され，P_2は最初のラインのTHRESHOLD で指定される。（P_2は最初のGATEのラインのGATEのパ

ラメーター番号を使用した方が使いやすかったかも知れない。）ＧＡＴＥの使用法はＴＹＰＥ＝０の場合と同じである。

ＲＥＤＩＮＦは次のように作られる。

オフセット	内　　容
0	4（TYPE＊4）
2－8	TYPE＝0と同じ
10	$(P_1-1)*2$
12	$(P_2-P_1)*2$
14	B_{RS}
16	REDCHNの番地
18	スペクトル領域の先頭番地
20	スペクトル領域の先頭番地
22	以下ＧＡＴＥのデータ

⑨のPOSTBLにはじまる部分がＴＹＰＥ＝１に対するＲＥＤＩＮＦ作成のルーチンである。この段階でのレジスターの内容は⑧の上のコメントに書かれている。

処理ルーチンの先頭で⑨のように

```
CALL    CALBIT
```

を行えばＲ５にB_{RS}が入る。（ＣＡＬＢＩＴを呼ぶための条件はＲ３がテーブルＡＵＸＰＡＲ中の第１番目のＴＨＲＥＳＨＯＬＤをポイントしていることである。Ｒ５以外のすべてのレジスターは保存される。）次の⑩ではＴＨＲＥＳＨＯＬＤに入っているP_2がリストモードのパラメーター数ＴＴＬＬＳＴのはんい内に入っているかどうかチェックし，入っていない時にはリストバッファー中のデリミッターをデータとして参照するようにしている。一般にＴＹＰＥ≠０に対しては

INIT 時に入力データに対するチェックはＲ．Ｓ．のパラメーター番号，ＴＲＬ，ＴＲＵを除いて行われない。（データがどのような目的で使用されるか不明のため。）したがってまちがった入力によって計算機がエラーを起さないように，ＲＤＴＢＬの段階でチェックを行う必要がある。ここでは大きすぎる入力をした場合にリストバッファーより上の存在しない番地やデバイスの番地をアクセスする可能性をなくすための処置を行っている。ＴＴＹにエラーメッセージを出力すればもっと親切になる。⑪では$N_{RS}*2$を入れるためのＲＥＤＣＨＮの番地をセットしている。

最後に

```
JMP     STSTAD
```

を行うことによってオフセット２０の“スペクトル領域の先頭番地”以下のデータがＲＥＤＩＮＦに入れられる。ここではＲ１がオフセット２０をポイントしていることだけが必要である。

Ｒ．Ｓ．の作成はＲＥＤＳＰＣの⑰－㉒で行われる。この時点でのレジスターの内容は⑮の上のコメントに書かれている通りである。Ｒ０とＲ５は保存されなければならないが他のレジスターは自由に使ってよい。⑰ではサブルーチンＣＨＫＧＡＴを呼んでイベントがＧＡＴＥの条件を満足しているかどうかのチェックを行っている。ＣＨＫＧＡＴの呼び方は

CALL　　CHKGAT
BR　　ＧＡＴＥからはずれた時のブランチ先
ＧＡＴＥを満した時の処理命令

であり，ＣＡＬＬの際Ｒ１が最初のＧＡＴＥのパラメーター番号をポイントしていることが必要である。ここの場合は㉒へブランチしてREDCHNのテーブルをクリアーしている。⑱ではリストバッファーからN_1，N_2をとり，N_{RS}＊２を計算している。１１／０４は割算の命令を持つていないので引算とシフトによって行っている。⑲ではN_{RS}＊２をREDCHNに入れ，⑳で対応するスペクトル領域をインクリメントしている。オーバーフローが生じた時にはサブルーチンRDOVFを呼んで対応するオーバーフロー領域をインクリメントしている。この部分はいつもこの形で使えばよい。（カウントを１６ビットで打切ってよければRDOVFを呼ばなくてもよい。）インクリメントの前後でＰＳＷ（プロセッサーステータスワード）を操作しているのは，この間ＭＢＤからの“BUFFER−FULL”のインターラプトを禁止することによって，オーバーフローが生じてからオーバーフロー領域をインクリメントする前にREDSPCルーチンが打切られ　ことを防いでいる。（RDSP５５ではこのような処置をしていないので原理的にはオーバーフローカウントの算え落しが起りうる。）

最後に

JMP　　ENDSP

を行うことによって次のR. S.の処理へ移る。

5.1.2　２個の入力の和のスペクトル

１パラメーターのTYPE＝４はパラメーター番号P_1，P_2の２台のＡＤＣからのデータN_1，N_2から

$$N_{RS}=(N_1+N_2)*2^{-TRL}$$

によってスペクトルを作成する。P_1，P_2はTYPE＝１と同様にR. S.のパラメーター番号と第１行のTHRESHOLD　で指定される。有効ビット長は

$$B_{RS}=B_{P_1}-TRL-TRU$$

であり，N_{RS}がこのビット長を越えた時はデータは捨てられる。ＧＡＴＥの使用法はTYPE＝０と同じである。

REDINFは⑬で作成され次のような構成になっている。

オフセット	内　　容
0	４＊４
2−8	（TYPE＝０と同じ形式）
１０	$(P_1-1)*2$　（０，２，４……）
１２	$(P_2-P_1)*2$
１４	最大チャンネル数　$2**B_{RS}-1$
１６	REDCHNテーブルの番地

１８	スペクトル領域の番地
２０以下	ＧＡＴＥのデータ

⑬で実際にセットしているのはオフセット１２－１６の３ワードだけである。R. S.の作成は㉗で行われる。

5.1.3　作成されたリデュースドスペクトル間でのＧＡＴＥ

すでにのべたように，TYPE＝３はREDCHNテーブルをリストバッファーと同様に使用して，R. S.間でＧＡＴＥをかけてスペクトルを作成する。

スペクトルを作成するR. S.の番号を第１行のTHRESHOLD で指定し，ＧＡＴＥとして使用するR. S.の番号を第２行以下のＴＹＰＥの欄で指定する。ＧＡＴＥの区間はTYPE＝０の場合と同様に指定されるが，第１行が使用されないことと，ＧＡＴＥのパラメーター番号P－＊＊ の欄が無視されることが異なる。第２行以下でＴＹＰＥの欄が０の行はＧＡＴＥとして使用されない。なおR. S.のパラメーター番号ＰＲＭは単に有効ビット長を決定するために使用され，作成されるR. S.との関係は必ずしもなくてもよい。

REDINFは⑫で作成され，次の構成となっている。

オフセット	内　　容
０	３＊４
２－８	（TYPE＝０と同じ）
１０	スペクトルを作成するN_{RS}が入っているREDCHNの番地
１２	最大チャンネル数＊２
１４	スペクトル領域の先頭番地
１６	GATEとして使用するN_{RS}が入っているREDCHNテーブルの番地
１８	ＧＡＴＥの下限のチャンネル数
２０	ＧＡＴＥの上限のチャンネル数
２２以下	セットされているＧＡＴＥの個数だけ１６－２０をくり返す。

ここでＧＡＴＥの使用法が通常の場合と異なるのでオフセット１６以下も⑫で直接セットしている。最後に

```
JMP      SETDEL
```

によって，次のR. S.のためのREDINF作成に移る。この時Ｒ０がCMRED中の次のR. S.の情報の先頭をポイントしていなければならない。（最後のR. S.の場合はデリミッターの－１をポイントする。）

R. S.の作成は㉓，㉔で行われる。ＧＡＴＥのチェックはサブルーチンCHKGATをCALLせずに，㉓で直接行っている。

5.2 変更の手続き

変更したRDSP04を組み込んで最終的にATOMIC. AAAを作成することにする。ここで第3.1節でのべたように，イクテンションAAAは“CAMAC CONFIGURATION”の画面でロードするプログラムを指定するための入力に対応する。

変更は11／70で行う。まず〔10，54〕RDSP04. MAC を自分のU. I. C. にコピーし，変更を行って

RD4AAA. MAC

を作成する。

次に

```
MAC   RDSP04. AAA=〔1，1〕INSTBD／ML. ATOMIC／ML.
      〔uic〕RD4AAA
```

によってアセンブルし，オブジェクトを11／55のDK1に入れるディスクに入れる。次にディスクを11／55にロードし

```
PIP   DK1：〔10，14〕ATOMIC. AAA=DK1：〔10，14〕
      ATMMAN. BIN，〔uic〕RDSP04. AAA／ME
```

によって11／04にロードするプログラムを作成する。タスクLODPRGは上のようにマージしたファイルをメモリー上に次々とオーバーライトして行くので，何個ものファイルをマージしてもよい。

11／55のタスクRDSP55の場合には

DK1：〔5，6〕RDSP55. TSK

に代るタスクを作成すればよい。識別のためにイクステンションをTSKの代りに他の名前を使用することにする。

例えばRDSP55. BBBを作成することにする。11／70で〔10，54〕RDSP55. MAC を自分のU. I. C. にコピーし，変更を行ってRD5BBB. MACを作成し

```
MAC   RD5BBB=RD5BBB
```

によってオブジェクトファイルを作成する。

次にコマンド入力

ⓐ〔10，54〕RDSP55BLD

を行うと，イクステンションの名前のキーインを要求されるのでBBBを入力すると，タスクRDSP55. BBBが自分のU. I. C. に作成される。これを11／55のDK1に移し

```
REM   RDSP55
INS   DK1：〔uic〕RDSP55. BBB
```

によってインストールすればよい。

このインストールの操作はコマンド

ⓐ〔1，2〕TANDEM

が行われた後で行う必要がある。

6. データ収集以外への応用および今後の拡張について

今回の拡張はデータのとり込み方の変更を容易にするためのものであるが，第4章でのべたように他の目的のために使用することは自由である。例えば実験装置に対して何らかのイニシャライズを行うことはどれかのマイクログラムのEDITOR中で容易に行える。（第3.2節でのべたように INIT END の際一般にCAMACに対してBZが出されるが，実験装置のコントロールのためには単にBDを禁止するだけに変更した方がよいかもしれない。）また本来はデータ収集のためのPHA1等のマイクロプログラムの，どれかの番号のプログラムを実験装置のコントロールのために使用すれば，CAMACからのLAMによって装置のコントロールを行うことが可能である。またスペクトル領域はどのような目的で使用してもよく必ずしもヒストグラムを作成しなくてもよい。

より一般的な拡張のためにはすでに設置されているインターラプトモジュールを使用すればよい。現在はこのモジュールからのLAMによってインターラプトサービスルーチンINTMDVが起動され，続いてマイクロプログラムSTTSTPが起動され，データ収集のストップを行っているだけである。INTMDVとSTTSTPを書きかえることによってかなり自由な拡張が可能となる。この場合変更の便宜を考えるとINTMDVも含めて，CAMACからのLAMに対するインターラプトサービスルーチンをRDSP04のように独立したファイルにした方がよいと思われる。独立させるためには単にリンクのために必要な番地をATOMIC，MLBに追加すればよい。必要に応じて何らかの出力モジュールを追加することは今回の拡張の範囲内で可能である。

現在11／04のメモリーが不足気味である。将来はメモリーマネージメントをつけられる上位機種にグレードアップする必要があるがさしあたり次のような解決策がある。ATMMAN中のイニシャライズに関連した部分の大部分はメモリー中に常駐している必要はない。現在マイクロプログをロードするためのLOADERが第3.2節でのべたように，イニシャライズ中だけ共有メモリーにロードされ動くが，上にのべた部分をLOADERにつけてATMMANからとり除けば1kW程度のスペースが確保できる。この変更はATMMANを多少整理するだけで容易に行えるはずである。

また特殊なコントロールのために条件を入力する場合には“CAMAC CONFIGURATION”のEXTRA PARAMETERを使用するか，11／04でTTYから入力するやり方が可能であるが，より使いやすくするためにはINITのサブタスクを1つ追加してフリーパラメーターだけの画面を作ればよい。各パラメーターには名前の欄を設けて，例えば第1行で名前のフィールドを入力するかどうか指定し，指定すれば名前のフィールドにパラメーターの内容を示す任意の入力ができるようにすればよい。さらにこの画面の第1画面の“CAMAC CONFIGURATION”の画面の前に置き，必要な時だけ ALT キーによって表示するようにすれば，この画面を必要としない使用者にとってはこれまでと全く同じになる。これらのパラメーターは収集モードと直接関係ないので，特別のチェックは INIT 段階では必要なく，サブタスクの作成は容易である。あとはこれらのパラメーターをCOMM04を通じてATOMICに送ればよいだけである。これらのパラメーターをCOMTBLに入れるためには，今回11／55のメモリーが8kW増えたので，COMTBLを2kWに拡張すればよいであろう。

7. ＴＴＹサービスルーチンおよびキーボードコマンド

イニシャライズ時等に必要に応じて１１／０４側でＴＴＹに対する入出力が使用できる。またデータ収集ストップ時にはキーボードコマンドによってメモリーの内容をダンプしたり，ＣＡＭＡＣコマンドを実行することが可能である。１１／０４はＯＳの下で動いていず，ＴＴＹもきちんとしたドライバープログラムによって管理されていない。したがってＴＴＹの使用には多少の制約がある。ＴＴＹに対するインターラプトサービスルーチンはレベル０で動き，
START 時には入力を無視する。

7.1 ＴＴＹからの入力

プログラム中でＴＴＹの入力は

CALL READ 〔@ADREAD〕

を実行することによって，バッファーTTYBUFに入力文字が入れられる。ここで〔　〕はEDITORのようにATOMIC. MLBによってATMMANとリンクされているプログラム中でＲＥＡＤを使用する時は

CALL @ADREAD

によって行うことを示している。

入力は＜ＣＲ＞または＜ESC＞キーが入力されるか，８０文字入力されることによって終了する。サブルーチンＲＥＡＤからのRETURN時には入力のステータスがＩＯＳＴに入れられる。ＩＯＳＴの上位バイトには入力文字数（＜CR＞と＜ESC＞はバッファーには入れられるが文字数には数えられない。）が，下位バイトには次の完了コードが入る。

０：　＜ＣＲ＞または８０文字入力

２：　＜ESC＞入力

ＴＴＹからの入力はＲＥＡＤによる他，＜CTRL－C＞で始まるキーボードコマンド（第7.3節参照）によっても使用される。両者はフラグTTYSTSによって

$$\text{TTYSTS}=\begin{cases} 100000 & ;\ \text{READ} \\ 1 & ;\ \text{コマンド入力} \end{cases}$$

と区別されている。入力終了後TTYSTSはクリアーされる。

ＲＥＡＤの下ではいくつかの特殊キーが次のように使用される。

＜CTRL－U＞：入力キャンセル，バッファーのはじめから入れ直す。

＜DELETE＞：一字消去

＜CTRL－O＞：無視される。

＜CTRL－C＞：無視される。

なおＲＥＡＤにおいてはすべてのレジスターが保存される。

入力文字のデコードのためにサブルーチンDECBINとOCTBINが用意されている。

DECBINは１０進数，OCTBINは８進数のデコードのために使用され，それぞれ

CALL DECBIN 〔@ADDTOB〕

CALL OCTBIN 〔@ADOTOB〕

によってＣＡＬＬされる。ＣＡＬＬの際Ｒ０がデコードする文字列の先頭をポイントしていなければならない。デコードされたバイナリー数はＲ５に入れられ，Ｒ０はバッファー中のデリミッターの次の文字をポイントしている。デコードは下に示すようなデリミッターに出会うまで行われ，ステータスがDCDSTSに入れられる。DCDSTSの上位バイトにはデコードした文字数が，下位バイトにはデリミッターを区別する完了コードが次のように入れられる。

完了コード	デリミッター
0	<CR>
2	,
4	<LF>
6	$
10	<ESC>
377	その他の数字でない文字（エラー）

先頭のスペースは無視される。負符号の入力は許される。オーバーフローのチェックは行わないので必要ならＣＡＬＬ側で文字数をチェックすればよい。

Ｒ０，Ｒ５以外のレジスターは保存される。

7.2 ＴＴＹへの出力

一連の文字列をＴＴＹに出力するためには文字列の先頭番地をＲ５に入れ

CALL TALFW 〔@ADMSGO〕

文字列の最後にはデリミッターとして０のバイトを入れなければならない。(.ASCIZを使えばよい。)出力後Ｒ５はデリミッターの０をポイントしており，他のレジスターは保存される。

内部データを６桁の８進数として出力するためにはＲ４にデータを入れて

CALL BOATYP 〔@ADOTYP〕

を行えばよい。この場合すべてのレジスターが保存される。

内部データの１０進表示のためにBTOD，BTOSDの変換サブルーチンが用意されている。ＢＴＯＤは符号なし，BTOSDは符号付の変換である。使用法は

CALL BTOD 〔@ADBTOD〕

CALL BTOSD 〔@ADBTSD〕

である。必要な入力は次の通りである。

Ｒ０ ：文字列を入れるバッファーの先頭番地

Ｒ１ ：文字数

Ｒ２ ：バイナリーデータ

変換終了後R0はバッファー中の次の番地をポイントしており，他のレジスターは保存される。なおR1で指定した文字数が変換した文字数より多い場合には先頭にスペースが入れられ，不足の時は全体に＊が入れられる。

8進変換のためにはサブルーチンBTOOがあり

CALL BTOO 〔@ADBTOO〕

によって，R2に入れられたバイナリー数が，R0によってポイントされるバッファーに6文字の8進数に変換されて入れられる。変換後R0はバッファー中の次の番地をポイントし，他のレジスターは保存される。

7.3 TTYキーボードコマンド

イニシャライズ完了後，データ収集ストップ時にはTTYからのキーボードコマントが使用できる。キーボードコマンドは主としてプログラムのデバッグのために用意されている。

キーボードコマンドは<CTRL−C>の入力によって開始され（>がエコーされる。）<CR>入力によって，コマンド入力が終了する。コマンド処理中はフラグINCMNDがセットされ，START ONは禁止される。

(1) メモリーダンプ

D， 開始番地， 終了番地

によって11／04のメモリーが8進表示でTTYから出力される。番地は8進数で入力する。

(2) MBDメモリーダンプ

M， 開始番地， 終了番地

によってMBDのコントロールメモリーの内容がDコマンドと同様にダンプされる。

(3) CAMACコマンド

C， F， C， N， A， DL， DH

によってCAMACコマンドが実行される。FCNAは10進数で入力され，データの下位16ビットDLと上位8ビットDHは8進数で入力される。データを必要としないコマンドではDL，DHは入力しなくてよい。デリミッターとして$<LF><ESC>を使って，80文字内で何個かのコマンドを同時に入力することができる。コマンド実行後データとQ，Xのレスポンスが TTYに出力される。

キーボードコマンドにおいても特殊キー<CTRL−U>と<DELETE>の意味はサブルーチンREADの場合と同様であるが，<CTRL−O>はダンプ出力を打切るために使用される。

8. プログラミング上必要なデータの構造

第４，５章にのべたプログラミングを行うためには内部のデータの構造に関する知識が必要となる。同じデータが１１／０４側と１１／５５側の両方にある場合が多いので両方のデータについて同時にのべる。どちらのデータかを区別するため，１１／０４のデータの場合は１１／０４，１１／５５のデータの場合はCOMTBLまたはRDSP５５（COMTBL に入っていず，タスクRDSP５５内で定義されているデータ）と［ ］内に書くことにする。なお１１／０４のデータはすべてATOMIC, MLBによってリンクされるのでEDITORやRDSP０４中で以下にあげる名前で使用できる。

この他，共有メモリーのデータ構成についての知識が必要なので，メモリーマップをFig.3に示す。

CAMAC ［COMTBL］CMCMC［１１／０４］ ２３ワード

標準的なCAMACモジュールの構成に関するデータである。

0	ＬＡＭグレーダー１
2	〃 2
4	コインシデンスユニット
6	ＡＤＣインターフェース１
8	〃 2
!	〃 3－7
２０	〃 8
２２	ＧＡＴＥモジュール（COMMAND OUT）
２４	クロック
２６	パルスシェーパー
２８	プリセットスケーラー１
３０	〃 2
３２	１２チャンネルスケーラー
３４	＊
３６	インターラプトモジュール

各々のデータは次のようになっている。

15 13	12 8	7 3	2 0
C	N	GL	CH#

ここでＣＨ#はＭＢＤのＤＭＡチャンネルの番号でありコインシデンスユニットとＡＤＣインターフェースに対してだけ使用される。オフセット３４はスケーラーのカスケードに関する情

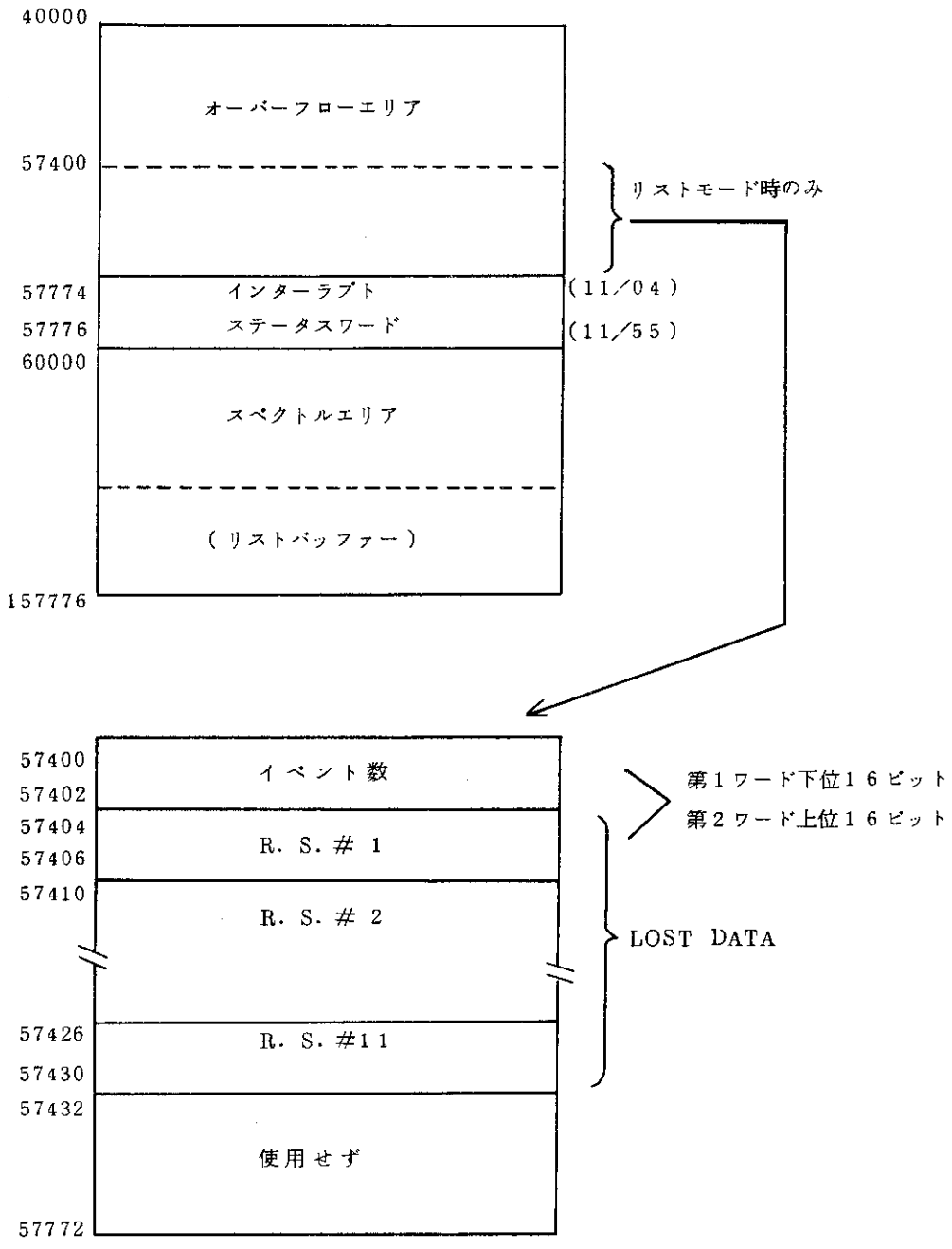


Fig. 3　共有メモリーのMAP

報であるが使用されていない。

なお第４章等でのべたように，“標準的”ということはマイクロプログラム等との関連において標準的であるだけであり（コインシデンスユニットからのＬＡＭによりマイクロプログラムＬＩＳＴが起動される等）ハードウェアとして標準的なモジュールである必要はない。

EXTRAC〔COMTBL，11／04〕 ２４ワード

標準的でないCAMACモジュールの構成に関するテーブルでありA00－C07の順に各々１ワードずつ割当てられ，次のビット構成となっている。

15 13	12 8	7 3	2 0
C	N	X1	X2

Ｘ１，Ｘ２の意味はマイクロプログラムによって異なる。プログラム番号０のリストモードのプログラムではＸ１によってＡＤＣのビット長が指定され，Ｘ２は使用されない。

EXTRAP〔COMTBL，11／04〕 １２ワード

“CAMAC CONFIGURATION”の画面で入力される“EXTRA PARAMETER”のテーブルで最初の６ワードが１０進入力のデータ，次の６ワードが８進入力のデータである。使用法は自由である。

ACQMOD〔COMTBL〕 １０ワード

データ収集モードに関するテーブルで２パラメーターＰＨＡを行うかどうかで構成が異なる。

(1) ２パラメーターＰＨＡなしの場合

0	N_{1P}	MODE＝0	
2	N_{LIST}	バッファーサイズ	
4	ビット長	〃	パラメーター１
6	〃	〃	〃 ２
｜	〃	〃	｜
18	〃	〃	パラメーター８

ここでN_{1P}，N_{LIST} は１パラメーターＰＨＡの個数とリストモードで使用される標準的なＡＤＣの個数である。リストモードを行わない時はN_{LIST}は０となる。MODE＝０によって２パラメーターＰＨＡがないことを示す。バッファーサイズは２５６ワード単位のブロック数で示される一方のリストバッファーの大きさである。

(2) ２パラメーターＰＨＡありの場合

0	N_{1P}	MODE＝1	
2	ADC－Y番号	ADC－X番号	
4	Yビット長	Xビット長	
6	TRY	TRX	トランケート・ビット数
8	ビット長	ＡＤＣ番号	パラメーター３

10	ビット長	ADC番号	パラメーター4
⋮	〃	〃	⋮
18	〃	〃	パラメーター8

どちらの場合も使用されないパラメーターの欄は0になる。例えば2パラメーターPHAを行う場合でN_{1P}が1ならオフセット10以下は0となる。

CSABAT〔COMTBL〕 可変長，最大20ワード

カレント・スペクトル・エリア・ブロック・アドレス・テーブルであり，データ収集中のスペクトル領域の先頭ブロック番号（0−63）とブロックサイズを示す。

0	サイズ	先頭番号	パラメーター1
2	〃	〃	〃 2
4	〃	〃	〃 3
⋮	〃	〃	⋮
14	〃	〃	〃 8
16	〃	〃	R. S. がある時は
	〃	〃	その個数だけ
	−1（デリミッター）		

使用していないパラメーターおよびリストモードのパラメーターに対しては0が入る。2パラメーターPHAに対してはパラメーター1が使用されパラメーター2の欄は0となる。 リデュースド・スペクトルのデータはオフセット16から始まる。

CMACQ〔11／04〕 6ワード

ACQMODとCSABATから作られるテーブルである。

(1) 2パラメーターPHAなし

0	N_{1P}	MODE=0	
2	N_{LIST}	リストバッファーサイズ	
4	パラメーター2	パラメーター1	ADC番号
6	〃 4	〃 3	
8	〃 6	〃 5	
10	〃 8	〃 7	

使用していないパラメーターに対してはADC番号は0となる。

(2) 2パラメーターPHAあり

0	N_{1P}	MODE=1	
2	Xビット長	TRY	TRX
4	ADC−Y番号	ADC−X番号	
6	パラメーター4	パラメーター3	ADC番号
8	〃 6	〃 5	
10	〃 8	〃 7	

トランケートするビット数TRX, TRYはオフセット2の下位バイトの4ビットずつを使用する。

CMCSA〔11／04〕可変長，最大20ワード

ACQMODとCSABATから作られる。

0	ビット長	先頭ブロック番号	パラメーター1
2	〃	〃	〃 2
⋮	〃	〃	⋮
14	〃	〃	〃 8
16	ブロックサイズ	〃	R. S. の個数
	〃	〃	だけ
	−1（デリミッター）		

使用していないパラメーターに対しては0が入る。PHAのスペクトル（オフセット14まで）とリデュースドスペクトルの場合で上位バイトの意味が異なることに注意を要する。

SSTADR.〔11／04〕可変長，最大19ワード

各スペクトル領域の先頭番地のテーブルであり，サブルーチンRDTBLで作成される。

0	パラメーター1	PHAモードでない
2	〃 2	パラメーターの場合は0となる。
⋮	⋮	
14	〃 8	
16	R. S. 1	R. S. の個数だけ
	R. S. 2	

ACQSPL〔COMTBL，11／04〕 9ワード

0	EXTLST
2	TTLLST
4	NGATE
6 − 16	6ワード未使用

EXTLST：リストモードで使用される標準ADC以外のパラメーター数

TTLLST：リストモードのパラメーター数

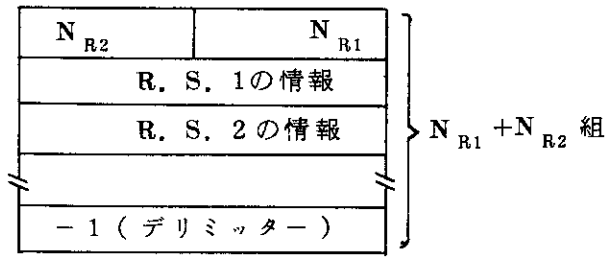
NGATE ：1パラメーターR. S. に対するGATEの行数（最大7）

これらの3ワードは上の名前で参照できる。

REDSPC〔COMTBL〕 可変長，最大300ワード

CMRED〔11／04〕 可変長，最大200ワード

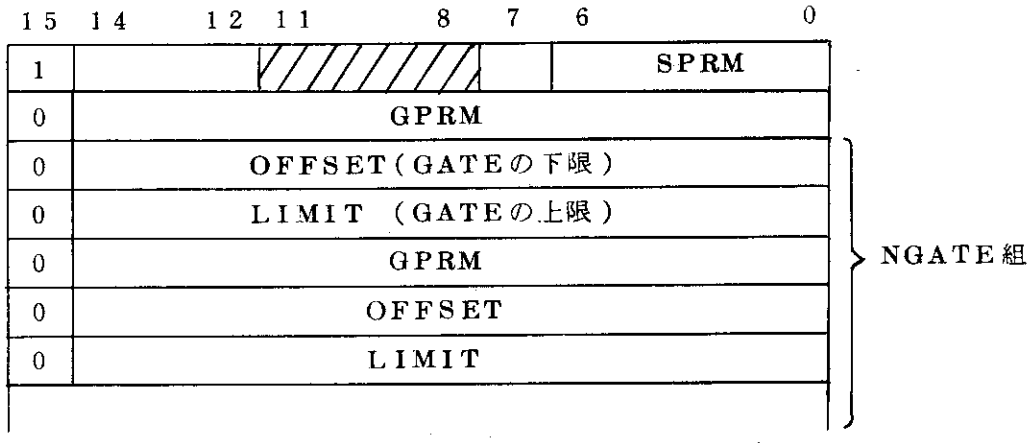
リデュースドスペクトルに関する情報である。



N_{R1}，N_{R2}はそれぞれ1パラメーター，2パラメーターのリデュースドスペクトルの個数である。リデュースドスペクトルのない時は第1ワードが－1になる。

各リデュースドスペクトルの情報は1パラメーターと2パラメーターの場合で形が少し異なる。

(1) 1パラメーターR. S.の場合



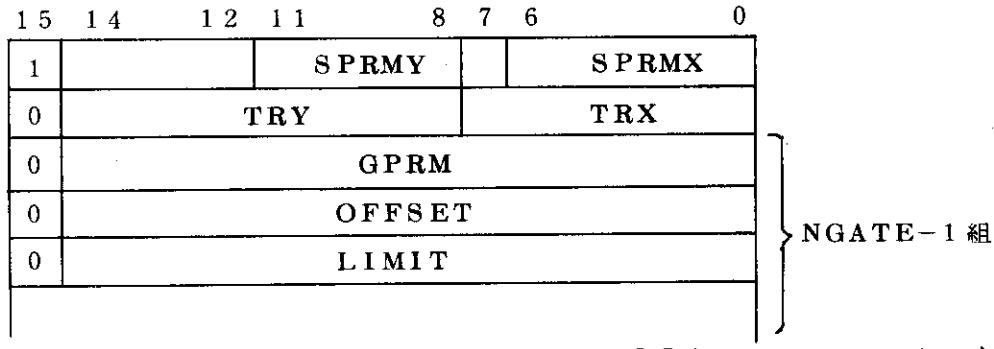
SPRM：R. S.を作るパラメーター番号（0，1，2……）

GPRM：GATEのパラメーター番号。使用しないGATEに対しては177_8

第1ワードのビット7，12−14は INIT 時におけるリデュースドスペクトルの変更に関連して次のように使われている。

ビット	名　前	1の時の意味
7	SW0	テーブルをはじめて作る時
12	SW3	追加したR. S.
13	SW1	変更あり
14	SW2	とりやめにしたR. S.

(2) 2パラメーターR. S.の場合

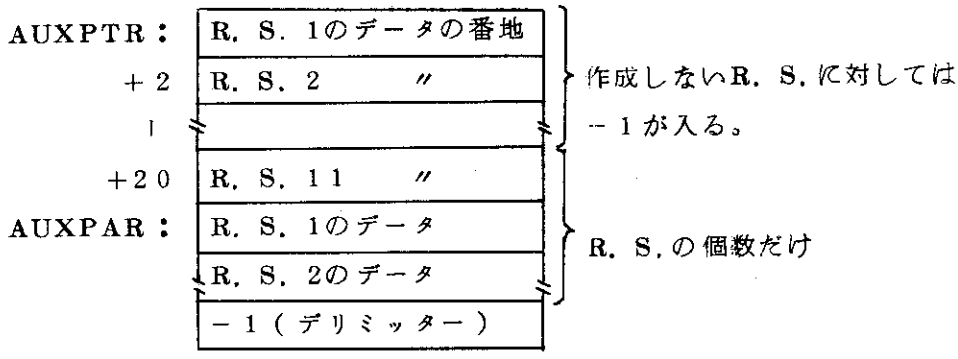


SPRMX，SPRMYはR. S.を作成するパラメーターの番号（0，1，2……）であり，TRX，TRYはトランケートするビット数である。GATEのデータはリストモードのパラメーター数が2の時はない。SW0−3は1パラメーターR. S.の場合と同じである。

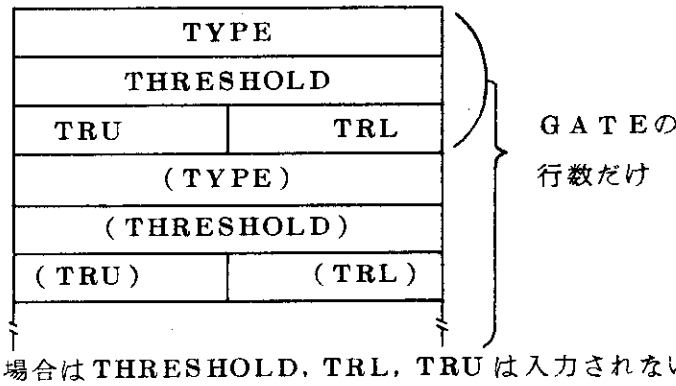
ＳＷ２のビットはあるR. S.の作成をとりやめにする時，他のR. S.の領域を変更しないために使用されている。しかしこれまでの使用経験ではこの処置はそれほど重要でなく（必要なら“MEMORY MAP”の画面で領域を指定できる），またこのためにタスクEDIMODが複雑になっていることもあり，将来はSW0－3をすべてやめてパラメーター番号のために使用できるビット数を増した方がよいと思われる。

REDAUX〔COMTBL，１１／０４〕可変長，最大８５ワード

各R. S.のTYPE，THRESHOLD，TRL，TRUを入れるテーブルAUXPARと，各々のR. S.に対するAUXPARの先頭番地のテーブルAUXPTRから成っている。AUXPTRは11／０４では絶対番地であるがCOMTBL内ではAUXPARの先頭からのオフセットである。



各々のR. S.のデータは次の通りである。



２パラメーターのR. S.の場合はTHRESHOLD，TRL，TRUは入力されないがテーブルは同じ形をとる。

REDINF〔COMTBL，１１／０４〕 可変長

COMTBL：最大２７３ワード，１１／０４：最大４００ワード

データの形式は第５章にのべた通りである。COMTBLではRDSP５５で処理される分のデータが入っているが，１１／０４ではRDSP５５で処理される分もダミーとして入っている。

REDCHN〔RDSP５５，１１／０４〕 １１ワード

各R. S.で計算されたN_{RS}の２倍を入れるテーブルである。（第5.1.3節参照）

MAXCHN〔RDSP５５，１１／０４〕 １６ワード

MAXCHN（Ｉ）＝２＊＊Ｉ－１ Ｉ＝１，１６

FREG〔11／04〕　7＊2＊8ワード

ＭＢＤのファイルレジスターの初期値のテーブルである。主としてEDITORによってセットされる。FORTRAN的に書けばFREG(I, J, K)に対してJ＝1，2がバンク0，バンク1のレジスターセットに対応し，K＝1－8がチャンネル0－7に対応し，Iは次のようにレジスターに対応する。

I	レジスター
1	DAR
2	ILR
3	CCR
4	WCR
5	CTR
6	GP1
7	GP2

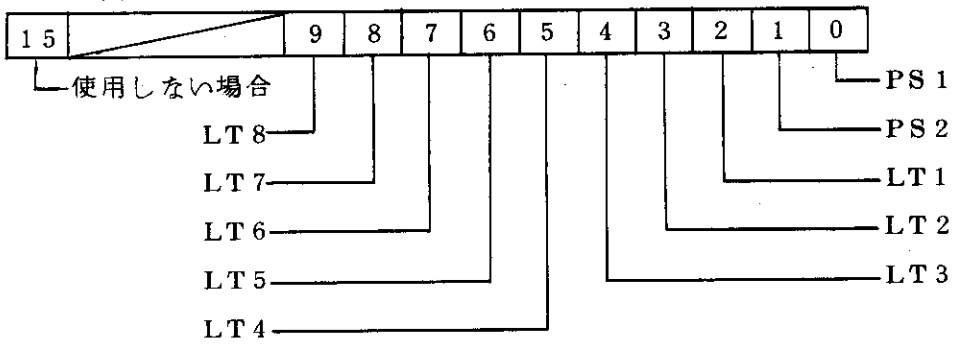
PRSTVL〔COMTBL〕　15ワード

CMPRE 〔11／04〕　14ワード

スケーラーの使用状態とプリセット値に関するテーブルである。

0	タイマー	スケーラーの割当て
2	電流積算計	
3	パラメーター1のライブタイム	
｜	｜	
18.	〃　8　〃	
20.	PS1プリセット値（下位16ビット）	
22.	〃　（上位16ビット）	
24.	PS2プリセット値（下位16ビット）	
26.	〃　（上位16ビット）	
28.	クロック周波数	11／04にはない。

オフセット0－18. 使用するスケーラーの割当てを示すテーブルで各々次のようなビット構成を持ち，“PRESET”の画面で割当てられたスケーラーに対応するビットがセットされる。これらのデータは11／04では使用されない。



オフセット28.はCOMTBL中にだけあり，ビット0－6が10^0－10^6の周波数に対応する。

CURTS〔COMTBL〕PS11〔11／04〕 44ワード

スケーラーの読みとり値が入るテーブルである。各々に2ワードずつ，下位16ビット，上位16ビットの順で割当てられる。

オフセット	スケーラー
0	PS1
4	PS2
8.－36.	LT1－8
40.－84.	12チャンネルスケーラー・0－11

OVRFLW〔COMTBL〕OVFLOW〔11／04〕 2ワード

インターラプトステータスワードの領域に対応するオーバーフローデータ

EVTCT〔COMTBL〕 2ワード

共有メモリー中の，リストモードのイベント数のコピー。第1ワードが下位16ビット，第2ワードが上位16ビット。DATACQによってセットされる。

LOSTCT〔COMTBL〕 2＊11ワード

共有メモリー中の各R. S.の“LOST DATA”のコピー。各R. S. 毎に下位16ビット，上位16ビットの順で2ワードずつ割当てられる。DATACQによってセットされる。

FBUF, SBUF, PBUF〔11／04〕 各1ワード

それぞれバッファー0，バッファー1，一杯になったバッファーの先頭番地である。

SIZE, LIST〔11／04〕 各1ワード

リストバッファーの一方の大きさと，1イベントの長さであり，どちらもバイト単位である。

MASK〔11／04〕 1ワード

MBDのマスクレジスターにロードされるマスク。

CMSK1, CMSKC1〔11／04〕 各1ワード

クレート1のLAMグレーダーに送られるLAMのマスクで，CMSK1がGL＝1－16，CMSKC1がGL＝17－24 のマスクである。

CMSK2, CMSKC2〔11／04〕 各1ワード

クレート2に対する上と同様のマスク。

TASK55〔11／04〕 1ワード

DR11Cの11／55側での使用状態を示す。DATACQがDR11Cを使用している時ビッ

ト０がセットされる。

CLDSBL〔１１／０４〕　１ワード

両方のバッファーが一杯になってコインシデンスユニットのＬＡＭが禁止されている時に１にセットされる。

STATUS〔１１／０４〕　１ワード

１１／０４がマイクロプログラムの終了を知るためのステータスワード。必要に応じてマイクロプログラムの最後で１にセットされる。

STOP〔１１／０４〕　１ワード

データ収集状態を示すフラグ。“ＳＴＯＰ”（ START オフ）でビット０が１にセットされる。プリセット値によって“ＳＴＯＰ”になった時は更にビット１５がセットされる。ビット15は SCALER CLEAR によってクリアーされる。実験装置のコントロールが終了するまで“ＳＴＡＲＴ”したくない時は，このワードの他のビットを使用すればよい。

NEXTF〔１１／０４〕　１ワード

リストバッファーが一杯になった時，次に使用するバッファーが０であるか１であるかにしたがって０または１にセットされる。

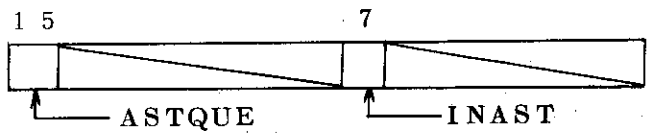
BFLG0，BFLG1〔１１／０４〕　各１ワード

リストバッファーが一杯になった時マイクロプログラムＬＩＳＴによって１にセットされ，DATACQから“ＢＵＦＦＥＲ－ＵＮＬＯＡＤ”の通信を受けた時ＡＴＯＭＩＣによってクリアーされる。各々がバッファー０および１に対応する。

WCD〔１１／０４〕　１ワード

一方のリストバッファーに入るイベント数

REDFLG〔COMTBL〕　１ワード



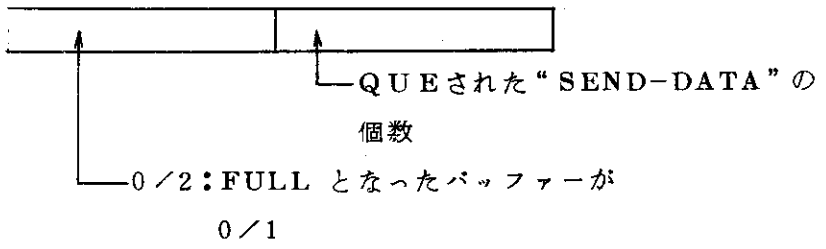
DATACQとRDSP５５間の同期のために使用されるフラグビットである。

ＡＳＴＱＵＥ：DATACQがRDSP５５に対して“ＢＵＦＦＥＲ－ＦＵＬＬ”の“ＳＥＮＤ－ＤＡＴＡ”ディレクティブを出す際に，ＲＤＳＰ５５がＡＳＴサービスルーチン中にいなければセットされる。

ＩＮＡＳＴ　：ＲＤＳＰ５５がＡＳＴサービスルーチン中にいる時にセットされる。

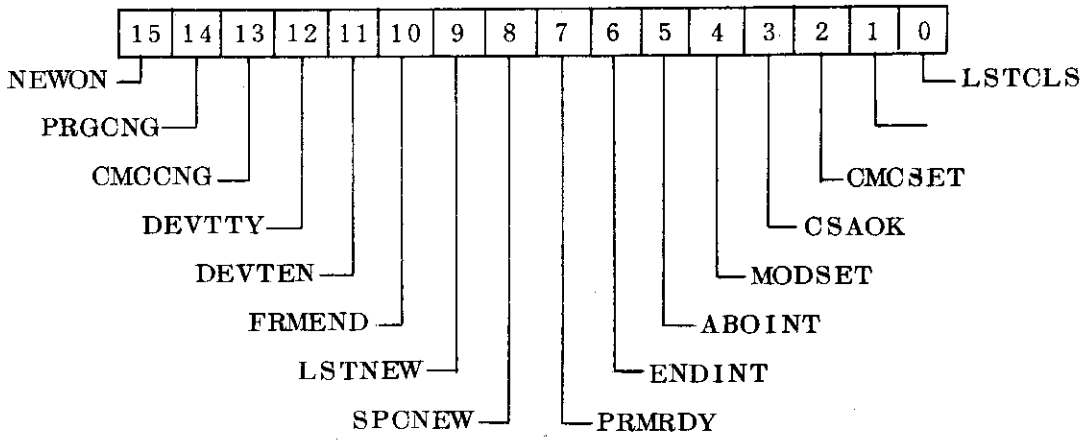
SNDCNT〔COMTBL〕　1ワード

DATACQからRDSP55への“SEND-DATA”に関連して使用される。



ACQFLG+4〔COMTBL〕

INIT 時に使用されるフラグワードである。この解説にのべた範囲のプログラムの変更を行う場合には直接関係ないが，今回のプログラムの書きかえによって文献5) にのべられている内容と異った部分があるので今後のためにここにあげておく。



NEWON ： NEW オンでセット。
PRGCNG：ATOMICのイクステンション名変更。
CMCCNG：CAMAC構成に変更あり。
DEVTTY：INIT がコンソールをATTACHしている。
DEVTEN：INIT がオペコンをATTACHしている。
FRMEND：現在の画面で，全パラメーターがセットされた。
LSTNEW：リストモードの収集モードに変更あり。
SPCNEW：スペクトルダンプファイルCREATE。
PRMRDY：全画面のパラメーターがセットされた。
ENDINT： INIT END 処理中。
ABOINT：サブタスクABORT中。
MODSET：EDIMODに入ったらセット。
CMCSET：EDICMCに入ったらセット。
LSTCLS：リストダンプファイルをクローズすべき時セット。

なおこれらのビット中には使用されていないものがかなりある。

附録1　リストモードEDITORのプログラムリスト

```
;  LIST OF LIST.MAC
        .TITLE LIST MODE
;*********************************************************
;     ASSEMBLE COMMAND
;        LIST,ED*=[1,1]INSTBD/ML,ATOMIC/ML,[U,I,C]LIST
;*********************************************************
        ;  MODIFIED 25-SEP-80
        .MCALL  INSTBD,ATOMIC
        INSTBD
        ATOMIC
        .ENABL  AMA,ABS
        .NLIST  ME
        .LIST   MEB
;
;       EDITOR (LIST MODE)
;
        .=60000
SIE:    .WORD   <END1-ST1>/2    ;SIZE OF EDITOR
ST1:    .WORD   .+2     ;LOAD ADR(EDITOR)
        .WORD   5       ;KIND OF PROGRAM=LIST MODE
        .WORD   CMDBLK  ;COMMAND BLOCK POINTER
;
;
        CMP     TTLLST,#32.     ;TOO LARGE # OF PRM?
        BLOS    OK              ;NO, OK
        MOV     #ERRMSG,R5
        CALL    @ADMSGO
        RETURN
ERRMSG: .ASCIZ  <15><12>/# OF PRM IS TOO LARGE/<15><12>
        .EVEN
OK:     MOV     CMCMC+4,R0      ;COINC. MODULE
        BIC     #177770,R0      ;GET C-L CH#
        ASL     R0
        ASL     R0
        MOV     R0,R1
        ASL     R0
        ADD     R0,R1
        ASL     R0
        ADD     R0,R1
        ADD     #FREG,R1        ;FREG POINTER
        MOV     FBUF,@R1        ;DAR=BASE ADR1
        CLC
        ROR     (R1)+
        MOV     #CD,R2  ;# OF EVENTS IN A BUF
        CLR     R3
        MOV     TTLLST,R4       ;# OF PRM
        INC     R4
1$:     TST     R4
        BEQ     2$
        ADD     R2,R3
        DEC     R4
        BR      1$
```

```
2$:     ASL     R3
        TST     -(R3)
        MOV     R3,R2
        ADD     FBUF,R3          ;LAST ADR OF BUF0
        CLC
        RUR     R3
        ADD     SBUF,R2          ;BUF1
        CLC
        ROR     R2
        MOV     R3,(R1)+         ;ILR=LAST0
        MOV     SBUF,@R1         ;CCR=BASE ADR2
        CLC
        ROR     (R1)+
        MOV     R2,(R1)+         ;WCR=LAST1
        MOV     @#40002,(R1)+    ;CTR=HEAD OF LIST MODE
        MOV     #1,(R1)+         ;GP1=1
        MOV     FBUF,@R1         ;GP2=DAR
        CLC
        ROR     (R1)+
        MOV     TTLLST,-(SP)     ;# OF FRM
        MOV     #X3,R4
        MOV     #CMACQ+4,R1
        CLR     R0
        MOVB    CMACQ+3,R2       ;# OF NORMAL ADC
        ASL     R2
        MOV     R2,OFFEXT
        CLR     R5               ;PRM# (0,2,...)
3$:     CMP     R5,OFFEXT        ;NORMAL ADC?
        BLO     31$              ;YES
        MOV     R5,R2
        SUB     OFFEXT,R2
        CMP     R2,#24.*2        ;IN EXTRA TABLE?
        BHIS    322$             ;NO, SET DUMMY CN
        MOV     EXTRAC(R2),R3
        BR      32$
31$:    MOVB    (R1)+,R2         ;GET ADC#
        DEC     R2
        ASL     R2
        MOV     CMCMC+6(R2),R3   ;GET C N
32$:    ASR     R3
        ASR     R3
        ASR     R3
        ASR     R3
        BIC     #170017,R3
        BNE     321$
322$:   MOV     #7000,R3 ;ERROR...DISCRIMINATE FROM DELIMITER
321$:   BIS     R3,ADC(R0)       ;SET CN
        BIS     R3,ADC+8.(R0)
        ADD     #16.,R0          ;NEXT ADC
        BIS     R3,(R4)+         ;READ ADC F C N A
        CMP     R5,OFFEXT        ;NORMAL ADC?
        BLO     33$
        MOV     EXTRAC(R2),R2
        ASR     R2
```

```
        ASR     R2
        ASR     R2
        BR      34$
33$:    MOVB    CMCSA+1(R5),R2
34$:    BIC     #^C37,R2          ;BIT SIZE
        CMP     R2,#16.
        BLE     35$
        MOV     #16.,R2
35$:    ASL     R2
        MOV     MAXCHN-2(R2),(R4)+  ;MASK FOR DATA
        TST     (R5)+
        DEC     (SP)
        BNE     3$
        MOV     #-1,ADC(R0)       ;DELIMITER
        TST     (SP)+
        CLR     @R4               ;DELIMITER
        MOV     CMCMC+4,R0        ;C-L C N
        ASR     R0
        ASR     R0
        ASR     R0
        ASR     R0
        BIC     #170017,R0
        BIS     R0,CMDBLK
        BIS     R0,CMDBLK+8.
        BIS     R0,X0             ;CLR C-L
        BIS     R0,X1             ;DSBL C-L
        RTS     PC
OFFEXT: 0
CMDBLK: PFCNA   26,0,0,0          ;ENABL LAM
        PFCNA   10,0,0,0          ;CLR LAM
        .NLIST  MEB
ADC:    .REPT   32.
        PFCNA   24,0,0,0          ;DSABL LAM
        PFCNA   26,0,0,1          ;ENABL CLEAR
        .ENDR
        .LIST   MEB
        .WORD   -1                ;DELIMITER
END1:   .WORD   0
;
;*************************************************************
;  FOLLOWING CODES ARE DUMMY MICRO-PROGRAM.
;     THEY ARE NOT USED.
;*************************************************************
;       LIST MODE MICRO
;
        .=40000
STL:    .WORD   <END0-ST0>/2
ST0:    .WORD   .-.       ;MICRO-PROGRAM LOAD ADDRESS (MBD MEMORY).
                          ;  SET BY COMM04.
        .WORD   5         ;KIND OF MICRO-PROGRAM
```

```
$=0
LIST0=$
        MV      GP2,MAR
        LCI     CLIST
        MV      MEM,BAR,BC0
        INM     CTR,CTR
RD0=$
        MV      MEM,ATR         ;MASK
        BCT     $,BRB
        :
        :
        :
*****  SAME AS LIST.MPR  *****
        :
        :
        :
CLIST=$
        .NLIST  MEB
X3:     .REPT   32.
        FCNA    0,0,0,0         ;READ ADC
        DATA    0               ;MASK
        .ENDR
        .LIST   MEB
        DATA    0               ;DELIMITER
CLAM=$
X0:     FCNA    10,0,0,0        ;CLR LAM
ADBFL=$
        DATA    BFLG0/2,BFLG1/2 ;ADDR OF BUF-FULL-FLAG
CLMSK=$
X1:     FCNA    24,0,0,0        ;DSABL LAM
DSBLCL=$
        DATA    CLDSBL/2        ;ADDR OF LAM-DSABL-FLAG
END0:   .WORD   0
        .END
```

附録2　リストモードマイクロプログラムのリスト

```
;  LIST OF LIST.MPR
;*****************************************************
; ASSEMBLE BY
;     MAC LIST.MB*=[1,1]BDINST/ML,ATOMIC/ML,[0,1,C]LIST.MPR
          .TITLE  LIST MODE MICRO-PRO
;           MODIFIED 25-SEP-80
          .MCALL  BDINST,ATOMIC
          BDINST
          ATOMIC
          .NLIST  ME
          .LIST   MEB
          .ENABL  AMA
LIST0=$
          MV      GP2,MAR
          LCI     CLIST
          MV      MEM,BAR,BC0
          INM     CTR,CTR
RD0=$
          MV      MEM,ATR           ;MASK
          BCT     $,BRB
          AND     BDL,MDR,WTR
          INM     CTR,CTR
          MV      MEM,BAR
          BCT     CL0,ZF
          INM     CTR,CTR,BC0
          BCT     $,DCB
          INM     MAR,MAR
          JP      RD0
CL0=$
          BCT     $,DCB
          INM     MAR,MAR
          SB      ILR,0
          BCT     FOL0,ZF
          LCI     CLAM
          MV      MEM,BAR,BC1
          LCI     LIST0
          INM     MAR,GP2,EX2
FOL0=$
          LD      ADBFL
          DEP     MAR
          MV      GP1,MDR,WTH
          BCT     $,DCB
          INM     MAR,MAR,RDR
          BCT     $,DCB
          MV      MDR,0
          BCT     CLRL0,ZF
          LD      CLMSK
          DEP     BAR,BC1
          LD      DSBLCL
          DEP     MAR,WTR
```

①

②

③

```
CLRL0=$
        LCI     CLAM
        BCT     $,BRB
        MV      MEM,BAR,BC1
        BCT     $,DCB
INT0=$
        BCF     FIN0,INB
        BCF     INT0,BDF
        LCI     INT0
        CON     EX1
FIN0=$
        LCI     LIST1
        MV      CCR,GP2,INT
        CON     EX2
;
LIST1=$
        MV      GP2,MAR
        LCI     CLIST
        MV      MEM,BAR,BC0
        INM     CTR,CTR
RD1=$
        MV      MEM,ATR
        BCT     $,BRB
        AND     BDL,MDR,WTR     ;MASK
        INM     CTR,CTR
        MV      MEM,BAR
        BCT     CL1,ZF
        INM     CTR,CTR,BC0
        BCT     $,DCB
        INM     MAR,MAR
        JP      RD1
CL1=$
        BCT     $,DCB
        INM     MAR,MAR
        SB      WCR,0
        BCT     FUL1,ZF
        LCI     CLAM
        MV      MEM,BAR,BC1
        LCI     LIST1
        INM     MAR,GP2,EX2
FUL1=$
        LD      ADBFL+1
        DEP     MAR
        MV      GP1,MDR,WTH
        BCT     $,DCB
        DEM     MAR,MAR,RDR
        BCT     $,DCB
        MV      MDR,0
        BCT     CLRL1,ZF
        LD      CLMSK
        DEP     BAR,BC1
        LD      DSBLCL
        DEP     MAR,WTR
CLRL1=$
        LCI     CLAM
        BCT     $,BRB
```

(4)

```
          MV        MEM,BAR,BC1
          BCT       S,DCB
INT1=$
          BCF       FIN1,INB
          BCF       INT1,BDF
          LCI       INT1
          CON       EX1
FIN1=$
          LCI       LISTO
          MV        DAR,GP2,INT
          CON       EX2
CLIST=$
          .NLIST    MEB
X3:       .REPT     32.
          FCNA      0,0,0,0             ;READ ADC
          DATA      0                   ;MASK
          .ENDR
          .LIST     MEB
          DATA      0                   ;DELIMITER
CLAM=$
X0:       FCNA      10,0,0,0            ;CLEAR LAM
ADBFL=$
          DATA      BFLG0/2,BFLG1/2
CLMSK=$
X1:       FCNA      24,0,0,0            ;DISABLE LAM
DSBLCL=$
          DATA      CLDSBL/2
          .END
```

(5)

附録３　１パラメーターPHAのEDITORのプログラムリスト

```
;  LIST OF PHA1.MAC
        .TITLE  1-PARAMETER PHA (EDITOR)
        ;MODIFIED 26-JUN-80
        .MCALL  INSTBD
        INSTBD
        .NLIST  ME
        .LIST   MEB
        .ENABL  AMA,ABS
        .MCALL  ATOMIC
        ATOMIC
        .=60000
P11EDS: <P11EDE-P11EDS-2>/2     ;SIZE
        .WORD   .+2             ;EDITOR LOAD ADDR
        3               ;1PARA PHA
        .WORD   CMDBLK                                  ┐
        MOVB    40005,R3                                │
        ASR     R3                                      │
        ASR     R3                                      │
        ASR     R3                                      │
        ASR     R3                                      │
        BIC     #177760,R3      ;PRM#                   │
        MOV     #FREG,R5                                │
        MOVB    CMACQ+3(R3),R0          ;GET ADC#       │
        BIC     #177400,R0                              │
        DEC     R0                                      │
        ASL     R0                                      │
        MOV     CMCMC+6(R0),R1          ;ADC CH#        │
        MOV     R1,R2                                   │
        BIC     #177770,R1                              │
2$:     TST     R1                                      │
        BEQ     1$                      ;FREG POINTER   │
        ADD     #28.,R5                                 │
        DEC     R1                                      │
        BR      2$                                      │
1$:     ASL     R3              ;PRM# *2                │
        MOV     SSTADR-2(R3),@R5        ;DAR--BASE      │
        CLC                                             │
        ROR     @R5                                     │
        MOV     (R5)+,R0                                │
        ASL     R0                                      │
        SUB     #60000,R0                               │
        ASR     R0                                      │
        ASR     R0                                      │
        ADD     #40000,R0                               │
        CLC                                             │
        ROR     R0                                      │
        MOV     R0,(R5)+        ;ILR--OVFLW             │
        ASR     R2                                      │
        ASR     R2                                      │
        ASR     R2                                      │
        ASR     R2                                      │
        BIC     #170017,R2              ;C N            ┘
```

①

```
        BIS     R2,CMDBLK
        BIS     R2,CMDBLK+8.
        MOV     R2,(R5)+                ;CCR--READ ADC
        MOV     #3,(R5)+                ;WCR
        MOV     40002,(R5)+             ;CTR
        MOVB    CMCSA-1(R3),R0
        BIC     #177400,R0      ;BIT NO.
        MOV     #2,R1
4$:     DEC     R0
        BEQ     3$
        ASL     R1
        BR      4$
3$:     DEC     R1
        MOV     R1,MASK0        ;SET MASK
        RTS     PC
CMDBLK: PFCNA   26,0,0,0        ;ENABL LAM
        PFCNA   26,0,0,1        ;ENABL CLEAR
        .WORD   -1
P11EDE: 0
;
;       1 PARA PHA
;       DAR--BASE
;       ILR--OVF
;       CCR--READ
;       WCR--3
;
        .=40000
P11ST:  <P11ED-P11ST-2>/2       ;SIZE
        .WORD   .-.     ;LOAD ADDR (SET BY MBDLNK)
        .BYTE   3,1*20+0        ;3: 1PARA PHA
                ;1: PRM#   0: PROG # (FOR SPECIAL M1CPU-PRO)
                ;.... SET BY COMM04
SNGL=S
        MV      CCR,BAR,BC0
        LD      MASK
        BCT     S,BRB
        AND     BDL,GP2
        AD      DAR,MAR,RDH
        BCT     S,DCB
        INM     MDR,MDR,WTR
        BCT     S,DCB
        BCT     OVFF,CF
        CON     EX2
OVFF=S
        MV      GP2,ATR
        SRL     2
        AD      ILR,GP1
        LD      OVF     ;57774/2
        SB      GP1,ATR
        BCF     CK6,ZF
        LCI     OVFLW   ;OVFLOW
        JP      GETA-1
```

```
CK6=$
        LD      OVF+1    ;57776/2
        SB      GP1,ATR
        BCF     GETA,ZF
        LCI     OVFLW+1  ;OVFLOW+2
        MV      MEM,GP1
GETA=$
        MV      GP1,MAR,RDR
        MV      GP2,ATR
        AND     WCR,GP2
        LCI     OVFDT
        AD      GP2,ATR
        AD      GP2,CTR
        MV      MEM,GP1
        INM     CTR,CTR
        BCT     $,DCB
        MV      MDR,ATR
        AND     MEM,ATR
        AD      GP1,ATR
        AND     MEM,GP1
        MV      MDR,ATR
        IOR     MEM,ATR
        EOR     MEM,ATR
        AD      GP1,MDR,WTR
        LCI     SNGL
        BCT     $,DCB
        CON     EX2
OVFDT=$
        DATA    1,17,20,360,400,7400,10000,170000
OVF=$
        DATA    57774/2,57776/2
OVFLW=$
OVFLW0: DATA    OVFLOW/2,OVFLOW/2+1
MASK=$
MASK0:  DATA    0
P11ED:  0
        .END
```

附録4　　リデュースドスペクトルのプログラムRDSP04のリスト

```
;  LIST OF RDSP04.MAC
        .TITLE  REDUCED SPECTRA (11/04)
        .IDENT  /MPR006/
;          MODIFIED 28-NOV-80
        .SBTTL  RDTBL AND SSTADR
        .ENABL  AMA,ABS
        .MCALL  ATOMIC,INSTBD
        INSTBD
        ATOMIC
;
;       TABLE OF START ADDRESS
;                AND
;       TABLE FOR REDUCED SPECTRUM
;
        PSW=177776
        LOST=57404
.=ARDTBL                                    (2)
RDTBL:  MOV     #CMCSA,R0       ;FOR GET START ADR
        MOV     #SSTADR,R1
1$:     MOV     (R0)+,R2        ;GET BLOCK
7$:     BIC     #177400,R2
        MOV     #60000,R3       ;START OF CO-USED MEM
        SWAB    R2
        ASL     R2              ;BLOCK SIZE *512.
        ADD     R2,R3
3$:     MOV     R3,(R1)+        ;SET START ADR
        CMP     #-1,@R0         ;END?
        BNE     1$
        TST     GSTADR          ;ONLY GET STR ADR?
        BEQ     4$              ;NO
        CLR     GSTADR          ;Y
        RTS     PC
4$:     TSTB    CMACQ           ;LIST MODE?
        BEQ     5$
6$:     RTS     PC              ;NO
5$:     TSTB    CMACQ+3         ;ONLY 1PARA?
        BEQ     6$              ;Y
GSIZE:  MOVB    CMACQ+2,R1      ;GET LIST BUF SIZE
        BIC     #177400,R1
        SWAB    R1
        ASL     R1
2$:     MOV     R1,SIZE         ;GET SIZE
        MOV     #160000,R2      ;FOR START ADR
        SUB     R1,R2           ;2'ND BUF START ADR
        MOV     R2,SBUF
        SUB     R1,R2           ;1'ST BUF START ADR
        MOV     R2,FBUF
        MOV     TTLLST,R0       ;GET P-NO. FOR LST MD
        INC     R0
        CLC
```

(1)

(3)

(4)

```
        ROR     R1
        CLR     R2
4$:     SUB     R0,R1           ;GET EVENT BLOCK NO.
        BMI     3$
        INC     R2
        BR      4$
3$:     MOV     R2,wCD          ;# OF EVENTS IN A BUFFER
        ASL     R0              ;DISTANCE APAR EVENT
        MOV     R0,LIST         ;TO BPARA EVENT
        MOV     CMRED,R0
        BNE     .+4.            ;NO RED SPEC
        RTS     PC
        CMP     #-1,R0
        BNE     .+4
        RTS     PC              ;NO RED SPEC
        MOV     R0,R1
        SWAB    R1
        BIC     #177400,R0
        BIC     #177400,R1
        MOV     R0,PRMCR1       ;# OF 1-PAR RED SPEC
        ADD     R1,R0
        MOV     R0,REDINF       ;RED SPEC NO.
        CLR     PCTR
        MOV     #CMRED+2,R0     ;POINT
        MOV     #REDINF+2,R1    ;POINT
GTABLE: CMP     @R0,#-1         ;ALL END?
        BEQ     ALENDO          ;Y
        BIT     #40000,@R0      ;ERASE?  ;SW2 BIT ON?
        BEQ     NEW0     ;NO
        TST     (R0)+
1$:     TST     (R0)+
        BPL     1$              ;NEXT POINTER
        TST     -(R0)
        BR      GTABLE
GOO:    INC     PCTR            ;# OF RED SPEC
        BR      GTABLE
;
NEW0:   JSR     PC,GOTBLO       ;MAKE 1 BLOCK TABLE
        MOV     R1,R3    ;SAVE ADDR OF REDINF BLOCK
        MOV     #WORKA,R4
1$:     MOV     (R4)+,(R1)+     ;STORE
        CMP     @R4,#-1
        BNE     1$
        MOV     @R4,(R1)+
        BR      GOO
ALENDO: CLR     2(R3)           ;SET NO NEXT POINTER
        MOV     TENDA,ENDA      ;SET POINT END ADR
        CLR     FIRST
        RTS     PC
;
```

④

```
;
;       MAKE REDUCED SPEC TABLE
;         FOR A SPECTRUM INTO WORKA
;       R0--CMRED
;       RESULT--WORKA
;       TEMP END ADR--TENDA
;
GOTBL0: MOV     R1,-(SP)                ;ADDR OF REDINF BLOCK
        MOV     #WORKA,R1
        TST     PRMCR1          ;1PARA END OR NOTHING?
        BEQ     1$              ;Y
        CLR     (R1)+   ;1PARA FLAG
        BR      2$
1$:     MOV     #2,(R1)+        ;2PARA FLAG
2$:     MOV     R1,-(SP) ;ADDR OF "NEXT REDINF PNTR" IN WORKA
        TST     (R1)+           ;SET NEXT POINT AFTER END
        MOV     PCTR,R5         ;GET LOST POINT
        ASL     R5
        ASL     R5
        ADD     #LOST,R5
        MOV     R5,(R1)+        ;SET LOST POINTER
        MOV     WCD,(R1)+       ;INIT # OF EVENT TO BE HANDLED
        TST     (R1)+           ;CURNT BUF ADR SET AT END OF CH.
        TST     PRMCR1          ;2PARA?
        BEQ     PA2RA           ;Y
        MOV     (R0)+,@R1       ;1PARA
        BIC     #^C177,@R1      ;PRM# A
        ASL     (R1)+           ;PAR# *2
STRTAD: MOV     PCTR,R2
        ASL     R2              ;SPC# (0,2,...)
        MOV     AUXPTR(R2),R3   ;AUXPAR PNTR
        MOV     (R3)+,R4                ;TYPE
        TST     PRMCR1                  ;1PARA?
        BEQ     10$                     ;NO
        MOV     R4,-12.(R1)
        ASL     -12.(R1)
        ASL     -12.(R1)                ;SET TYPE*4
        BR      20$
10$:    CMP     TTLLST,#2
        BNE     12$
        CLR     R4      ;NO TYPE FOR 2-ADC 2-PARA
12$:    MOV     R4,-20.(R1)
        ASL     -20.(R1)
        ASL     -20.(R1)
        ADD     #2,-20.(R1)      ;SET TYPE*4+2 FOR 2-PARA
20$:    TST     R4
        BNE     SPCIAL                  ;SPECIAL REDSPC
        TST     PRMCR1                  ;1PARA?
        BEQ     STSTAD                  ;NO
        CALL    CALBIT          ;R5=BIT SIZE-TRL-TRU
        MOV     (R3)+,(R1)+     ;THRESHOLD
        MOVB    (R3)+,R4        ;TRL
        MOV     R4,(R1)+        ;SET TRL
        ASL     R5
        MOV     MAXCHN-2(R5),(R1)+      ;MAX CH#
```

(5)

```
        MOV     #REDCHN,(R1)
        ADD     R2,(R1)+        ;REDCHN PNTR
STSTAD: MOV     SSTADR+16.(R2),(R1)+
SETGAT:
PRMBO:  MOV     @R0,@R1         ;GATE PAR#
        BMI     SETDEL          ;END
        CMP     @R1,#177        ;UNUSED GATE?
        BNE     10$             ;NO
        ADD     #6,R0           ;SKIP UNUSED GATE
        BR      PRMBO
10$:    TST     (R0)+
        ASL     (R1)+           ;SET PRM#*2
        MOV     (R0)+,R4        ;GET OFFSET
        MOV     (R0)+,R5        ;GET LIMIT
        TST     R4
        BNE     30$             ;NOT FULL GATE
        MOV     -2(R1),-(SP)    ;PRM#*2
        CALL    ADCBIT
        MOV     (SP)+,R3        ;BIT SIZE
        ASL     R3
        CMP     R5,MAXCHN-2(R3) ;LIMIT=MAX?
        BLT     30$             ;NOT FULL GATE
        TST     -(R1)           ;REMOVE GATE PAR#*2
        BR      PRMBO
30$:    MOV     R4,(R1)+        ;SET OFFSET
        MOV     R5,(R1)+        ;SET LIMIT
        BR      PRMBO
SETDEL: MOV     #-1,(R1)+       ;Y  SET DELIMITTER
        SUB     #WORKA,R1
        MOV     R1,R4           ;SIZE OF TABLE
        MOV     (SP)+,R3 ;RESTORE ADDR OF "NXT REDINF PNTR"
        ADD     (SP),R4 ;POP & ADD ADDR OF THIS REDINF BLOCK
        MOV     R4,@R3  ;POINTER FOR NEXT BLOCK IN REDINF
        CMP     #-1,@R0         ;RED SPEC END?
        BNE     .+4
        CLR     @R3             ;NO NEXT POINTER
        MOV     R4,TENDA        ;TEMP ADR SAVE
        TST     PRMCR1          ;1 2 PARA?
        BEQ     10$             ;2
        DEC     PRMCR1
10$:    MOV     (SP)+,R1
        RTS     PC
PA2RA:  MOV     (R0)+,R4
        MOV     R4,R5           ;GET PRM X Y
        BIC     #^C177,R4
        ASL     R4              ;PRM X
        MOV     R4,(R1)+
        SWAB    R5
        BIC     #^C17,R5
        ASL     R5
        MOV     R5,(R1)+        ;SET PRM Y
        MOV     @R0,R4          ;GET TRNCX
```

```
        BIC     #177760,R4
        MOV     R4,(R1)+
        MOV     @R0,R4          ;GET TRNCY
        ASR     R4
        ASR     R4
        ASR     R4
        ASR     R4
        BIC     #177760,R4
        MOV     R4,(R1)+        ;SET TRNCY
        MOV     (R0)+,R4        ;GET SHIFT LEFT CTR
        SWAB    R4
        BIC     #177600,R4      ;BIT X -TRNCX
        MOV     R4,(R1)+        ;SET COUNT
        BR      STRTAD
;
;
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;  SPECIAL RED-SPEC (1-PARA)
;    FOLLOWING DATA HAVE BEEN SET IN REDINF
;       TYPE*4
;       SPACE FOR NEXT REDINF ADDR
;       LOST DATA POINTER
;       # OF EVENTS
;       SPACE FOR LIST BUF ADDR
;       PAR#*2 (0,2,...)
;
;  R0=> GATE PAR#*2 IN CMRED
;  R1=> REDINF (NEXT TO PAR#*2)
;  R2=  SPC# (0,2,...)
;  R3=> THRESHOLD IN AUXPAR
;  R4=  TYPE
;
;   R0 MUST POINT NEXT REDINF ON RETURN TO SETOEL
;
SPCIAL: TST     PRMCR1
        BLE     SPCL2P
        CMP     R4,#1           ;TYPE=1?
        BNE     10$
        JMP     POSTBL
10$:    CMP     R4,#2
        BNE     20$
        JMP     POSTBL
20$:    CMP     #3,R4
        BNE     30$
        JMP     SPGTBL
30$:    CMP     R4,#4           ;SUM ADC?
        BNE     40$
        JMP     SUMTBL
40$:    CMP     R4,#5
        BNE     99$
        JMP     TXYPOS
99$:    JMP     OTHRED
;***********************************************
;  SPECIAL RED-SPEC (2-PARA)
;       FOLLOWING DATA HAVE BEEN SET IN REDINF
```

```
;       TYPE*4+2
;       SPACE FOR NEXT REDINF ADDR
;       LOST DATA POINTER
;       # OF EVENTS
;       SPACE FOR LIST BUF ADDR
;       PRMX*2 (0,2,...)
;       PRMY*2 (")
;       TRX
;       TRY
;       BX-TRX
;
;  R0==>1ST GATE PAR# IN CMRED
;  R1==>REDINF (NEXT TO "BX-TRX")
;  R3==>THRESHOLD IN AUXPAR
;  R4=TYPE
;
SPCL2P: CMP     R4,#1                                    ) (8)
        BNE     99$                                      )
        JMP     TPOS2P
99$:    JMP     OTHRED
;*************************
;
;       REDINF FOR POSITION DATA
POSTBL:
        CALL    CALBIT          ;R5=B-TRL-TRU            ) (9)
        MOV     (R3),R3         ;THRESHOLD=2ND PAR#
        BEQ     10$             ;ERROR                   ) (10)
        DEC     R3
        CMP     R3,TTLLST
        BLO     20$             ;OK
10$:    MOV     TTLLST,R3       ;USE DELIMITER AS DATA !!!!!!!!!!
20$:    ASL     R3
        SUB     -2(R1),R3       ;2*(PAR#2-PAR#1)
        MOV     R3,(R1)+
        MOV     R5,(R1)+
        MOV     R2,@R1  ;SPC# (0,2,...)                  ) (11)
        ADD     #REDCHN,(R1)+ ;REDCHN POINTER(FOR CALCAL)
        JMP     STSTAD          ;SET START ADDR AND GATES
;
;*************************
;
;  CALC'D CH# FOR BOTH SPEC. & GATE
;
;       REDINF
;       3*4  (RED TYPE)
;       NEXT REDINF POINTER
;       LOST DATA POINTER
;       # OF EVENTS
;       LIST BUF POINTER
;       REDCHN POINTER FOR SPEC
;       MAX CH#*2
;       START ADDR
```

```
;       REDCHN POINTER FOR GATE
;       OFFSET
;       LIMIT
;          ******
;           *****
;       -1
;
SPGTBL:
        CALL    CALBIT          ;R5=BIT SIZE-TRL-TRU
        MOV     (R3)+,R4        ;THRESOLD=RED SPC# TO USE
        ASL     R4
        ADD     #REDCHN-2,R4
        MOV     R4,-2(R1)       ;REPLACE PAR#*2
        ASL     R5
        MOV     MAXCHN-2(R5),(R1)
        ASL     (R1)+           ;MAX CH#*2
        MOV     SSTADR+16.(R2),(R1)+    ;START ADDR
        TST     (R3)+           ;==>NEXT TYPE (GATE SPC#)
        ADD     #6,R0           ;=>GATE PAR# IN CMRED
10$:    TST     (R0)+           ;NEXT SPEC?
        BMI     50$             ;YES
        MOV     (R3)+,R5        ;GATE SPC# (1,2,...)
        BEQ     20$             ;SKIP
        ASL     R5
        ADD     #REDCHN-2,R5
        MOV     R5,(R1)+        ;REDCHN POINTER FOR GATE
        MOV     (R0)+,(R1)
        ASL     (R1)+           ;OFFSET*2
        MOV     (R0)+,(R1)
        ASL     (R1)+           ;LIMIT*2
        ADD     #4,R3
        BR      10$
20$:    ADD     #4,R3           ;=>NEXT TYPE IN AUXPAR
        ADD     #4,R0           ;SKIP GATE
        BR      10$
;
50$:
        TST     -(R0)
        JMP     SETDEL
;
;**************************************
;  REDINF FOR SUM OF 2-ADC
SUMTBL:
        CALL    CALBIT
        MOV     (R3),(R1)       ;THRESHOLD=2ND PAR#
        BEQ     10$
        DEC     (R1)
        CMP     (R1),TTLLST
        BLO     20$             ;OK
10$:    MOV     TTLLST,(R1)     ;USE DELIMITER FOR DATA !!!!!!!
20$:    ASL     (R1)            ;2ND PAR#*2 (0,2,...)
        SUB     -2(R1),(R1)+    ;2*(2ND PAR# - 1ST PAR#)
        MOVB    1(R3),R3        ;TRL
        MOV     R3,(R1)+
        ASL     R5              ;2*(B-TRL-TRU)
```

(12)

(13)

```
        MOV     MAXCHN-2(R5),(R1)+      ;MAX CH#
        MOV     R2,(R1)                 ;SPC# (0,2,...)
        ADD     #REDCHN,(R1)+           ;REDCHN ADDR
        JMP     STSTAD
;
;
;*************************************************
;  REDINF FOR 2-DIM POSITION DETECTOR(1-PARA)
;       0   5*4
;       .........
;       10.  PAR#*2  (P1)
;       12.  (P2-P1)*2
;       14.  (P3-P2)*2
;       16.  (P4-P3)*2
;       18.  B-TRL
;       20.  THRESHOLD*2
;       22.  (2**(B-TRL-TRU)-1)*2 ...MAX CH# *2
;       24.  REDCHN POINTER
;       26.  START ADDR
;       28.  GATES.......
;
;    N=((P3+P4)/(P1+P2+P3+P4))*2**(B-TRL)
;    NRS=N-THRESHOLD
;        TOP TRU BITS ARE TRUNCATED.
;       FOR GP1=1, NEXT PAR# IS P2.
;       FOR GP1=2, NEXT TO THE NEXT IS P2.
;       FOR GP1=0, NEXT PAR# IS P3 AND P2=P1 P4=P3.
;
TXYPOS:
        MOV     R0,-(SP)                ;SAVE ADDR OF GATE
        MOV     (R0),R0                 ;GP1-1
        CMP     R0,#1                   ;0 OR 1?
        BLOS    1$                      ;YES
        CLR     R0                      ;SET TO 0
        BR      11$
1$:     INC     R0
        ASL     R0                      ;0,2,4
11$:    MOV     R0,R4
        ASL     R4
        ADD     R4,R0                   ;3*(0,2,4)
        ADD     #OFFXY,R0
        MOV     #3,R4
2$:     MOV     (R0)+,(R1)+             ;SET OFFSET FOR P2-4
```

(13)

```
        DEC     R4
        BNE     2$
        CALL    CALBIT          ;R5=B-TRL-TRU
        MOV     R5,(R1)
        MOVB    3(R3),R0        ;TRU
        ADD     R0,(R1)+        ;B-TRL
        MOV     (R3),(R1)
        ASL     (R1)+           ;THRESHOLD*2
        ASL     R5
        MOV     MAXCHN-2(R5),(R1)
        ASL     (R1)+           ;MAX CH# *2
        MOV     #REDCHN,(R1)
        ADD     R2,(R1)+
        MOV     (SP)+,R0        ;ADDR OF GATE
        ADD     #6,R0           ;SKIP 1ST GATE
        JMP     STSTAD
OFFXY:  0,2,0
        2,2,2
        4,-2,4
;
;
;*****************************************
;  REDINF FOR
;     2-DIM POSITION SPECTRUM (2-PARA)
;       :
;       :
;  SPACE FOR LIST BUF ADDR
;  PRMX*2
;  BX-TRLX
;  THX
;  MAX-X ..2**(BX-TRX)-1
;  BX-TRLY
;  THY
;  MAX-Y ..2**(BX-TRY+EBY)-1  OR TOP OF LIST BUFF
;  BX-TRX
;  START ADDR
;
IPOS2P: SUB     #8.,R1          ;==>NEXT TO PRMX*2
        MOV     -2(R1),-(SP)    ;PRMX*2
        CALL    ADCBIT
        MOV     (SP),(R1)       ;B (=BX=BY)
        SUB     2(R0),(R1)+     ;B-TRLX
        BGE     5$
        MOV     #1,-2(R1)
5$:     MOV     4(R0),(R1)+     ;THX
        MOVB    -1(R0),R4       ;B-TRX
        MOV     R4,8.(R1)       ;SET "B-TRX"
        ASL     R4
        MOV     MAXCHN-2(R4),(R1)+      ;MAX-X
        MOV     (SP),(R1)       ;B
        SUB     8.(R0),(R1)+    ;B-TRLY
        BGT     10$
        MOV     #1,-2(R1)
```

```
10$:    MOV     10.(R0),(R1)+   ;THY
        MOV     (SP)+,R4        ;B
        MOV     -2(R0),R5
        ASR     R5
        ASR     R5
        ASR     R5
        ASR     R5
        BIC     #^C17,R5        ;TRY
        SUB     R5,R4           ;B=TRY
        ADD     4(R3),R4        ;B=TRY+EBY
        BGT     20$
        MOV     #1,R4
20$:    ASL     R4
        MOV     MAXCHN-2(R4),(R1)       ;MAX-Y
;  TEST TOP OF LIST BUF
        MOV     FBUF,R4
        MOV     SSTADR+16.(R2),R5
        MOV     R5,4(R1)        ;SET START ADDR
        SUB     R5,R4           ;AVAILABLE BYTES
        MOV     2(R1),R5        ;B=TRX... BIT SIZE FOR X
        CLC
        ROR     R4      ;# OF WORDS
30$:    ASR     R4      ;AVAILABLE Y-VALUE
        DEC     R5
        BNE     30$
        DEC     R4              ;MAX-Y
        CMP     R4,(R1)
        BGE     40$
        MOV     R4,(R1)         ;REPLACE MAX-Y
40$:    TST     (R0)+
        BPL     40$
        TST     -(R0)
        ADD     #6,R1   ;==>DELIMITER
        JMP     SETDEL
;
;
OTHRED: MOV     #-2,-12.(R1)    ;DUMMY REDSPEC
10$:    TST     (R0)+
        BPL     10$
        TST     -(R0)
        JMP     SETDEL
```

```
;
;
;******  CALCULATE BIT SIZE
;       2(R3)=TRL,TRU
;       -2(R1)=PAR##2 (0,2,...)
;       OUT... R5=B-TRL-TRU
CALBIT: MOV     R4,-(SP)
        MOV     -2(R1),-(SP)    ;PRM##2
        CALL    ADCBIT
        MOV     (SP)+,R5        ;BIT SIZE OF ADC
        MOVB    2(R3),R4        ;TRL
        SUB     R4,R5
        MOVB    3(R3),R4        ;TRU
        SUB     R4,R5           ;B-TRL-TRU
        MOV     (SP)+,R4
        RETURN
;
;
;  IN:  2(SP)=2*PRM# (0,2,4...)
;   OUT:          (SP)=BIT SIZE
ADCBIT:
        MOV     R1,-(SP)
        MOVB    CMACQ+3,R1
        ASL     R1
        CMP     4(SP),R1        ;NORMAL ADC?
        BLO     10$             ;YES
        SUB     R1,4(SP)
        MOV     4(SP),R1        ;EXTRA MODULE # (0,2,4...)
        MOV     EXTRAC(R1),R1
        ASR     R1
        ASR     R1
        ASR     R1
        BIC     #^C37,R1
        BR      20$
10$:    MOV     4(SP),R1
        MOVB    CMCSA+1(R1),R1
20$:    MOV     R1,4(SP)        ;PUT BIT SIZE INTO STACK
        MOV     (SP)+,R1
        RETURN
```

```
        .PAGE
        .SBTTL  REDUCED SPECTRA
;
;       REDUCED SPECTRUM
;
.=ARDSPC                                        (14)
REDSPC: CLR     ACTIVE+6
;   SET LIST BUF POINTER
        MOV     #REDINF+10.,R0  ;DATA BUF POINTER
1$:     MOV     R0,R1
        SUB     #6,R1           ;NEXT REDINF POINTER
        MOV     PBUF,@R0        ;SET DATA BUF POINT
        TST     @R1             ;END
        BEQ     2$
        MOV     @R1,R0
        ADD     #8.,R0
        BR      1$
2$:     TST     RSDEVT          ;LIST MODE END?
        BEQ     3$              ;NO
        MOV     RSDEVT,R5       ;# OF EVENTS IN THE LAST BUF
        CLR     RSDEVT
        BR      4$
3$:     MOV     WCD,R5          ;# OF EVENTS IN A BUF
4$:     MOV     #REDINF+4,R0    ;NXT RD INF POINT
5$:     MOV     R5,4(R0)        ;SET # OF EVENT FOR LOST DATA
        TST     @R0
        BEQ     NXTEVT
        MOV     @R0,R0
        TST     (R0)+
        BR      5$
NXTEVT: MOV     #REDINF+2,R0
NXTRS:  MOV     (R0)+,R2        ;DISTINGUISH RED SPEC TYPE
        MOV     (R0)+,-(SP)     ;NEXT RED INF POINTER
        ADD     #4,R0
        MOV     R0,R1           ;DATA BUF POINTER
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;  R0,R1==>     "LIST BUF ADDR" IN REDINF
;  (SP)==>      NEXT REDINF TABLE
;  R5=          # OF EVENTS
;
;    R0,R5 MUST BE PRESERVED ON RETURN TO ENDSP
;
        JMP     @REDSRV(R2)
        NXTRED  ;-2...DUMMY  ---MAY BE FOR 11/55
REDSRV: ONEPAR  ;0
        TWOPAR  ;2
        POSITN  ;4
        XYPOS2  ;6
        POSSSD  ;8
        NXTRED  ;10.
        CALCAL  ;12.
        NXTRED  ;14.
        SUMADC  ;16.
        NXTRED  ;18.
        XYPOS           ;20.
;
```

(15)

```
CHKGAT: MOV     (R1)+,R2        ;GATE PAR#*2 (0,2,...)
        BMI     80$             ;GATE END....ADD ONE
        ADD     @R0,R2
        CMP     @R2,(R1)+       ;OFFSET
        BLO     90$             ;OUT OF WINDOW
        CMP     @R2,(R1)+       ;LIMIT
        BLOS    CHKGAT          ;IN WINDOW
        BR      90$
80$:    ADD     #2,(SP)
90$:    RETURN
;                               ;GO TO NEXT RED-SPEC
ENDSP:  ADD     LIST,@R0        ;  NEXT EVENT ADR
        DEC     -2(R0)          ;LOST DATA CNT DOWN
NXTRED: MOV     (SP)+,R0                ;NEXT RED INF POINTER
        BNE     NXTRS           ;LOOK AT NEXT RED. INF.
        DEC     R5              ;ALL END  HAVE LOOKED AT
;                               ;ALL EVENT BLOCK?
        BNE     NXTEVT          ;NO, CONT.
        CLR     RDSPEC          ;RED SPEC NOT BUSY
        JMP     @ADSRCH         ;GO TO "SEARCH"
;
;*****************************
ONEPAR:
        ADD     #14.,R1         ;==>GATE PAR#
        CALL    CHKGAT
        BR      9$
        CALL    UP1             ;ADD ONE
        BR      ENDSP
9$:     CLR     @10.(R0)        ;CLEAR REDCHN
        BR      ENDSP
;
TWOPAR:
        ADD     #14.,R1
        CALL    CHKGAT
        BR      ENDSP
        CALL    UP1
        BR      ENDSP
;
;
;
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;       REDUCED SPECTRUM FOR POSITION DATA
;
;    REDINF
;   4
;   NEXT REDINF POINTER
;   LOST DATA POINTER
;   # OF EVENTS
;   LIST BUF POINTER
;   PAR#*2 (P1)
;   2*(PAR#2-PAR#1)
;   BIT SIZE
;   REDCHN POINTER
```

(16)

```
;   START ADDR
;   GATES
;
;   (P2/(P1+P2))*2**8
;
POSITN:
        ADD     #12.,R1         ;==>GATE PAR#
        CALL    CHKGAT
        BR      9$
        MOV     R0,-(SP)        ;==>LIST BUF POINTER
        MOV     (R0)+,R3        ;DATA BUF ADDR
        ADD     (R0)+,R3        ;+PAR#*2
        CLR     R2              ;POSITION DATA
        MOV     (R3),R1 ;1ST DATA
        ADD     (R0)+,R3
        MOV     (R3),R3         ;2ND DATA
        ADD     R3,R1           ;P1+P2
        MOV     (R0)+,R4        ;BIT SIZE
1$:     ASL     R3
        CMP     R3,R1           ;* DIVIDE
        BLT     2$              ;*    R3(=P2)
        SUB     R1,R3           ;*    BY
        INC     R2              ;*    R1(=P1+P2)
2$:     ASL     R2
        DEC     R4
        BNE     1$
4$:     MOV     R2,@(R0)+       ;SAVE CH#*2 FOR CALCAL
        ADD     (R0),R2         ;+START ADDR
        BIS     #340,@#PSW
        ADD     #1,(R2)         ;INCREMENT
        BCC     5$
        JSR     PC,RDOVF        ;OVERFLOW
5$:     BIC     #340,@#PSW
        MOV     (SP)+,R0
        JMP     ENDSP
9$:     CLR     @8.(R0)
        JMP     ENDSP
;
;
;**********************
;  RED SPEC FOR POSITION SENSITIVE SSD
;   REDINF
;       8
;       NEXT REDINF POINTER
;       LOST DATA POINTER
;       # OF EVENTS
;       LIST BUF POINTER
;       PAR#*2
;       2*(PAR#2-PAR#1)
;       BIT SIZE
;       REDCHN POINTER
;       START ADDR
;       GATES
;
POSSSD:
```

```
        ADD     #12.,R1         ;==>GATE
        CALL    CHKGAT
        BR      9$
        MOV     R0,-(SP)
        MOV     (R0)+,R3        ;DATA BUF PNTR
        ADD     (R0)+,R3        ;DATA POINTER
        MOV     (R3),R1         ;E
        ADD     (R0)+,R3
        MOV     (R3),R3         ;E*X
        MOV     (R0)+,R4        ;BIT SIZE
        CLR     R2              ;X
1$:     CMP     R3,R1
        BLT     2$
        SUB     R1,R3
        INC     R2
2$:     ASL     R2      ;SERVE ALSO FOR BYTE ADDR.
        ASL     R3              ;NEXT BIT
        DEC     R4
        BGT     1$
        MOV     R2,@(R0)+       ;SAVE CH#*2 FOR CALCAL
        ADD     (R0),R2         ;+START ADDR
        BIS     #340,@#PSW
        ADD     #1,(R2)         ;ADD ONE
        BCC     5$
        JSR     PC,RDOVF
5$:     BIC     #340,@#PSW
        MOV     (SP)+,R0
        JMP     ENDSP
9$:     CLR     @8.(R0)
        JMP     ENDSP
;
;*****************************
;
;  RED SPEC IS MADE USING OTHER RED SPEC
;      FOR BOTH SPEC AND GATE (CH# IN REDCHN)
;
;      R0,R1==>LIST BUF POINTER
CALCAL:
        ADD     #8.,R1
10$:    TST     (R1)            ;ALL GATE END?
        BMI     20$             ;YES
        MOV     @(R1)+,R2       ;CH#*2 FOR GATE
        CMP     R2,(R1)+        ;>=OFFSET?
        BLT     2$              ;NO
        CMP     R2,(R1)+        ;=<LIMIT*2?
        BGT     2$              ;NO
        BR      10$             ;CHECK NEXT GATE
20$:    MOV     R0,R1
        TST     (R1)+           ;=>REDCHN POINTER FOR SPC
        MOV     @(R1)+,R2       ;CH#*2 FOR SPEC
        CMP     R2,(R1)+        ;>MAXCHN#*2?
        BHI     2$              ;YES, DO NOT COUNT
        ADD     (R1),R2         ;+START ADDR
        BIS     #340,@#PSW
```

(23)

(24)

```
        ADD     #1,(R2)         ;ADD ONE
        BCC     1$
        CALL    RDOVF           ;OVERFLOW
1$:     BIC     #340,@#PSW
2$:     JMP     ENDSP
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;
;       1 UP SPECTRUM
;
UP1:
        MOV     R0,R1           ;FOR GET PARA#
        TST     (R1)+
        TST     -8.(R0)         ;1 2 PARA?
        BNE     PAR2UP          ;2 PARA
        MOV     (R1)+,R2        ;GET PARA#
        ADD     @R0,R2          ;POINT TO THE DAT BUF
        MOV     @R2,R2          ;GET DATA
        SUB     (R1)+,R2        ;-THRESHOLD
        MOV     (R1)+,R3        ;TRL
        BLE     20$
10$:    ASR     R2              ;TRUNC. LOW BIT
        DEC     R3
        BNE     10$
20$:    CMP     R2,(R1)+        ;>MAX CH#?
        BHI     SAV000          ;YES
        ASL     R2
        MOV     R2,@(R1)+       ;SAVE FOR CALCAL
RS1UP0:
        ADD     @R1,R2          ;GET STRT ADR
        BIS     #340,@#PSW
        ADD     #1,@R2          ;SPEC 1 UP
        BCC     1$              ;NO OV FLW
        JSR     PC,RDOVF        ;OV FLW
1$:     BIC     #340,@#PSW
SAV001:
        RTS     PC
SAV000: CLR     @(R1)+          ;CLR SAVE DATA
        BR      SAV001
PAR2UP: MOV     (R1)+,R2        ;GET PRMX
        MOV     (R1)+,R3        ;GET PRMY
        ADD     @R0,R2          ;GET DATA POINT X
```

```
        MOV     @R2,R2          ;GET X DATA
        ADD     @R0,R3  ;DATA POINT Y
        MOV     @R3,R3          ;GET Y DATA
        MOV     (R1)+,R4        ;GET TRNC X
        BEQ     11$
1$:     ASR     R2
        DEC     R4
        BNE     1$
11$:    MOV     (R1)+,R4        ;GET TRNC Y
        BEQ     21$
2$:     ASR     R3
        DEC     R4
        BNE     2$
21$:    MOV     (R1)+,R4        ;SHIFT Y CTR
3$:     ASL     R3
        DEC     R4
        BNE     3$
        BIS     R3,R2           ;ADR POINTER
        ASL     R2
        BR      RS10P0
;
;       COUNT OVER FLOW
;       R2--DATA
;
RDOVF:  MOV     R1,-(SP)
        MOV     R2,-(SP)
        MOV     R3,-(SP)
        SUB     #60000,R2               ;FOR GET OVF AREA
        MOV     R2,R1
        ASR     R1
        ASR     R1
        ASR     R1
        ASL     R1
        ADD     #40000,R1
        CMP     #57774,R1       ;STATUS AREA?
        BGT     3$              ;NO
        BEQ     4$              ;Y
        MOV     #OVFLOW+2,R1    ;57776
        BR      3$
4$:     MOV     #OVFLOW,R1
3$:     MOV     @R1,R3          ;GET OVF DATA
        BIC     #177770,R2      ;GET BLOCK COUNT
        ADD     ADDNO(R2),R3    ;INC OVF
        BIC     CLRNO(R2),R3    ;CLR    OTHER BLK
        BIC     CLRSLF(R2),@R1  ;CLR CORRESPONDING BIT
        BIS     R3,@R1
        MOV     (SP)+,R3
        MOV     (SP)+,R2
        MOV     (SP)+,R1
        RTS     PC
;
```

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    SUM OF 2-ADC
SUMADC:
        ADD     #14.,R1         ;==>GATE PAR#
        CALL    CHKGAT
        BR      90$
        MOV     R0,R1
        TST     (R1)+           ;==>1ST PAR#
        MOV     (R1)+,R3
        ADD     (R0),R3
        MOV     (R3),R2         ;1ST DATA
        ADD     (R1)+,R3        ;==>2ND DATA
        ADD     (R3),R2         ;SUM OF 2 DATA
        MOV     (R1)+,R3        ;TRL
        BLE     40$
35$:    ASR     R2              ;TRUNC LOW BIT
        DEC     R3
        BNE     35$
40$:    CMP     R2,(R1)+        ;=<MAX CH#
        BHI     50$             ;NO
        ASL     R2
        MOV     R2,@(R1)+       ;SAVE TO REDCHN
        ADD     (R1),R2         ;+START ADDR
        BIS     #340,@#PSW
        ADD     #1,(R2)
        BCC     45$
        CALL    RDOVF
45$:    BIC     #340,@#PSW
        JMP     ENDSP
90$:    CLR     @10.(R0)
        JMP     ENDSP
50$:    CLR     @(R1)+
        JMP     ENDSP
;
;
;*********************************************
; 2-DIM POSITION DETECTOR  (1-PARA)
;
XYPOS:
        ADD     #20.,R1
        CALL    CHKGAT
        BR      99$
        MOV     R0,-(SP)
        MOV     (R0)+,R2        ;LIST BUF ADDR
        ADD     (R0)+,R2        ;==>P1
        MOV     (R2),R1         ;P1
        ADD     (R0)+,R2        ;==>P2
        ADD     (R2),R1         ;P1+P2
        ADD     (R0)+,R2
        MOV     (R2),R3         ;P3
        ADD     (R0)+,R2
        ADD     (R2),R3         ;P3+P4
        ADD     R3,R1           ;P1+P2+P3+P4
        MOV     (R0)+,R4        ;B-TRL
```

27

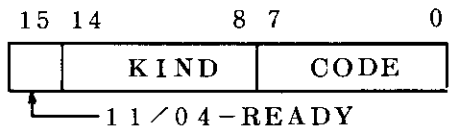
```
        CLR     R2
1$:     ASL     R3
        CMP     R3,R1
        BLO     2$
        SUB     R1,R3
        INC     R2
2$:     ASL     R2
        DEC     R4
        BNE     1$
        SUB     (R0)+,R2        ;-THRESH*2
        BMI     3$
        CMP     R2,(R0)+        ;=<MAX CH#?
        BLOS    4$              ;YES
        MOV     -2(R0),R2
        BR      4$
3$:     CLR     R2
        TST     (R0)+
4$:     MOV     R2,@(R0)+       ;SAVE IN REDCHN
        ADD     (R0),R2
        BIS     #340,@#PSW
        ADD     #1,(R2)
        BCC     5$
        CALL    RDOVF
5$:     BIC     #340,@#PSW
        MOV     (SP)+,R0
        JMP     ENDSP
99$:    CLR     @16.(R0)
        JMP     ENDSP
;****************************************
;  2-DIM POSITION DETECTOR (2-PARA)
XYPOS2: MOV     R0,-(SP)
        MOV     R5,-(SP)
        MOV     (R0)+,R2        ;LIST BUF ADDR
        ADD     (R0)+,R2        ;==>P0
        MOV     (R2)+,Q00       ;P0
        MOV     (R2)+,R3        ;P1
        MOV     (R2)+,R1        ;P2
        ADD     R3,R1           ;P1+P2
        MOV     (R2)+,Q22       ;P3
        CALL    DIV2
        MOV     R2,R5           ;NX
        MOV     Q22,R3          ;P3
        MOV     Q00,R1          ;P0
        ADD     R3,R1           ;P0+P3
        CALL    DIV2
        MOV     (R0)+,R4        ;B-TRX
2$:     ASL     R2              ;SHIFT NY FOR NX
        DEC     R4
        BNE     2$
        BIS     R5,R2           ;NY,NX
        ASL     R2
        ADD     (R0),R2         ;+START ADDR
        INC     (R2)
```

```
        MOV     (SP)+,R5
        MOV     (SP)+,R0
        JMP     ENDSP
Q00:    0
Q22:    0
;
DIV2:   MOV     (R0)+,R4        ;B=TRLX(Y)
        CLR     R2
1$:     ASL     R2
        ASL     R3
        CMP     R3,R1
        BLO     2$
        SUB     R1,R3
        INC     R2
2$:     DEC     R4
        BNE     1$
        SUB     (R0)+,R2        ;NX(Y)=NX(Y)-THX(Y)
        BMI     3$
        CMP     R2,(R0)+        ;MAX=X(Y)
        BLOS    4$
        MOV     -2(R0),R2       ;TRUNC AT MAX
        RETURN
3$:     CLR     R2
        TST     (R0)+
4$:     RETURN
;
ADDNO:  1,20,400,10000
CLRNO:  177760,177417,170377,7777
CLRSLF: 17,360,7400,170000
        .END
```

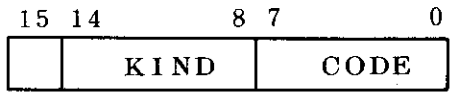
附録5　DR11Cとインターラプトステータスワード

11／55と11／04の交信のためにDR11Cが使用されるが，交信を容易にするために Fig. 3 に示したように2ワードのインターラプトステータスワードを共有メモリーに設けている。これらのビット構成は次のようになっている。

11／04→11／55（57774番地）



11／55→11／04（57776番地）



CODEとKINDは下に示すように交信の内容を示すために使用される。11／04−READYのビットは交信の同期をとるために使用され，11／04が新たに11／55にデータを送る際と11／55からデータを受信したというインターラプトを受けた際に1にセットされ，11／55のDR11Cのドライバーがデータを受けとった時クリアーされる。11／55でIO. RLBのQIOが出された時このビットがセットされていないとI／Oステータスワードに IE. DNRが返される。この時一般にはもう一度QIOを出す。CODEとKINDは次のように使用されている。両者とも8進数で示す。

(1)　11／04→11／55

CODE	KIND	意　味
1	0	START ON OK
	1	〃 NG
2	0	START OFF OK
	1	〃 NG
3	0	EXTERNAL START
4	0	EXTERNAL STOP (LAM GRADER, INT.MODULE)
	1	〃 (SCALER)
5	0	リストバッファー0 FULL
	1	〃 1 〃
10*	0	スケーラーのデータ送信
11	0	SCALER CLEAR OK
	1	〃 NG
12	0	PRESET値変更 OK
	1	〃 NG

13	0	[MEMORY] [ADVANCE] OK
	1	〃 NG
15	0	[INIT] 時のデータ。正常受信
	2	CAMAC NO-X
	3	RUN-ENABLE
100	0	DR11C DETACH OK

ここで＊）はDR11Cを通じてデータが送られる場合で第1ワードが転送語数第2ワード以降がデータである。これ以外の場合はDR11Cの出力レジスターはクリアーされる。

(2) 11／55→11／04

CODE	KIND	意味
1	0	[START] ON
2	0	[START] OFF
3	0	EXTERNAL START 受信
4	0	EXTERNAL STOP 受信
5	0	リストバッファー0 “UNLOAD”
	1	〃 1 〃
11	0	[SCALER] [CLEAR]
12＊)	0	PRESET値変更 PS1, PS2
	1	〃 PS1
	2	〃 PS2
13＊)	0	[MEMORY] [ADVANCE] R. S.は固定
	1	〃 R. S.もADVANCE
15		COMM04による [INIT] 時の送信
	0	データ収集モード全パラメーター
	1	LOADER
	2	EDITOR
	3	マイクロプログラム
	4	リストモードとりやめ
	5	2パラメーターPHAとりやめ
	6＊)	1パラメーターPHAとりやめ
	7	R. S.変更
	10	送信終了
	17	送信打切り（ABORT）
20	0	[LIST・BUF] [CLEAR]
	1	リストバッファーの残りに－1をセット
21	0	インターラプトステータスワードに相当するオーバーフローエリアのクリアー
100	0	DR11C DETACH
	1	DATACQによるDR11C ATTACH

ここで＊）はDR11Cを通じてデータが11／04に送られる場合で第1ワードが転送語数，第2ワード以降がデータとなる。 INIT 時の送信ではデータは共有メモリーを通じて送られる。ただし1パラメーターのPHAとりやめの場合はとりやめるADCインターフェースの番号がDR11Cを通じて1語ずつ送られる。

11／55側からDR11Cを使用する際はIO. RLBとIO. WLBのQIOを使用する。それぞれのパラメーターブロックの使用法は次の通りである。

QIO$ IO. RLB, ……, ＜buff addr, n, ST04, 100000＞

QIO$ IO. WLB, ……, ＜buff addr, n, ST55＞

ここで“buff addr”はタスク内のバッファーの番地，nはバイト数，ST04とST55はそれぞれのインターラプトステータスワードの番地（タスク内の），10000は11／04－READYのビットを示している。IO. RLBの場合一般には11／04からの転送語数が不明なので最大値をバイト数に指定する。実際の転送語数はI／Oステータスワードの第2ワードに入る。なおIO. WLBの場合にはバッファーの先頭にインターラプトステータスワードを入れる。逆にIO. RLBの場合はインターラプトステータスワードがバッファーの第1ワードに入れられる。

附録6　イベントフラグと“SEND-DATA”

タスク間の交信のためにコモンイベントフラグと“SEND-DATA”ディレクティブが使われる。

(1)　コモンイベントフラグ

フラグ	関連タスク	意　味
33	RDSP55→INIT	RDSP55 Active
	COMM04→INIT	11／04との交信でエラー
	EDICMC→INIT	CAMAC構成に変化あり
	LODPRG→COMM04	ATOMICロード・エラー
34	RDSP55→INIT	REDINF作成完了
	DATACQ→MEMORY	DR11C Detach完了
35	MEMORY→DATACQ	DR11C Detach完了
41	RTA →OPECON	RTA EXIT
42	サブタスク →INIT	次の文字入力要求
	DSCON →OPECON	DSCON EXIT
43	DSCON →DSUP →RTDSUP	ディスプレイ完了
44	DATACQ→OPECON →INIT	リストダンプMTクローズ完了
45	INIT →サブタスク	文字入力あり
46	HDCOPY→INIT →DSCON →DSMSG	ハードコピー完了
	INIT, OPECON	HDCOPY オフでセット
47	OPECON→DSMSG	プリセット値変更完了
	MEMORY→DSMSG	MEMORY MAP変更完了
48	PSが使用	
49	〃	
50	OPECON→DSCON →DSMSG	イベントあり
51	DSUP →DSCON RTDSUP→	ディスプレィデータ作成完了
52	DATACQ→INIT →COMM04 →MEMORY	DATACQ Active

53	OPECON→INIT →RTA →DUMP	コンソールI/O Kill 完了
54	サブタスク →INIT OPECON→INIT →RTA	EXIT, Rerun オペコンイベント発生
	INIT	オペコン，TTY入力発生
55	COMM04→INIT	COMM04 EXIT
56	OPECON→INIT	オペコンI/O Kill完了

(2) SEND-DATA

"SEND-DATA"ディレクティブは13ワードのデータを他のタスクに送ることができるが，データ収集システムではいつも先頭の1ワードだけを使用している。

データ	関連タスク	意味
0	INIT →DATACQ COMM04 → OPECON→DSUP →DSCON	DR11C Detach, EXIT 要求 EXIT要求
1	INIT →RDSP55	REDINF作成要求
2	OPECON→DATACQ INIT →	リストダンプファイルクローズ要求 (E-O-V)
3	OPECON→DATACQ	リストバッファークリアー要求
4	INIT LSTOUT→OPECON RTA → DATACQ→RDSP55 MEMORY→DATACQ	コンソールDetach完了 "BUFFER-FULL" DR11C Detach 要求
5	INIT →OPECON	オペコンDetach完了
6	INIT →COMM04	11/04との交信打切要求
7	INIT →サブタスク	NEXT イベント発生
8	INIT →サブタスク	NEW イベント発生
9	INIT →OPECON RTA →	コンソールDetach要求
10	INIT →OPECON	オペコンDetach要求
11	INIT →OPECON	INIT 終了通知

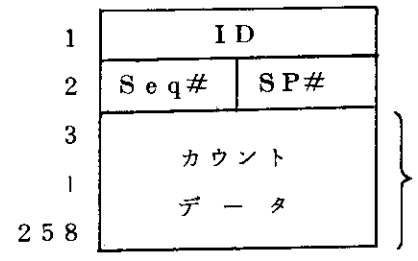
附録7　磁気テープのFORMAT

(1)　スペクトル・ダンプ

スペクトルの磁気テープへのダンプは800BPIで，258ワード単位のブロックで行われる。テープの先頭にRSX−11Mの標準のボリュームヘッダー（80バイト）がつけられる。1回の測定単位でダンプされ，各ダンプ毎にIDがつけられ，ダンプ毎に1ずつインクリメントされる。ダンプファイル指定時に複数回のダンプを指定した時もその毎にIDは1ずつ大きくなる。またダンプ毎にEOFマークがつけられる。実際にはダンプの度にEOFが2個（EOV）がつけられ，次のダンプでEOFを1個消してからダンプを行う。したがってダンプの途中で止めない限りスペクトルダンプの磁気テープにはEOVが付いている。テープ上のブロック構成をFig. 4 に示す。

各IDに対して“MEMORY MAP”の画面の順につけられたスペクトル番号（SP#）の順にスペクトルがダンプされる。各スペクトルのはじめにはヘッダーブロックがつけられる。ヘッダーブロックの内容はスペクトルによりFig. 5−7 のように異なる。

スペクトルのデータはまず下位16ビットのスペクトルブロックが次に上位4ビットのオーバーフローブロックの順でダンプされる。各々次のようになっている。



ここでSeq#は1つのスペクトル内でつけられたブロック番号で次のようになっている。

ヘッダーブロック　　：0

スペクトルブロック　　：1，2，3……

オーバーフローブロック：−1，−2，−3……

(2)　リスト・ダンプ

リストバッファーのダンプはスペクトルダンプの場合と異なり1600BPIの記録密度で行われる。テープの先頭にRSX11Mの標準のボリュームヘッダーがつけられることはスペクトルの場合と同じであるが，ダンプのブロックサイズはリストバッファーのサイズ＋1ワードである。第1ブロックはヘッダーブロックであり INIT の情報が入っている。しかし途中で収集モードが変更になった場合でも，磁気テープを代えないかぎり，ヘッダーブロックは書かれない。（この場合にはコンソールに警告がでる。）またダンプ毎にEOFをつけることをせず，リストモードがとりやめになった時と磁気テープへのダンプをやめた時および PHA オフの時にだけEOVを書く。したがって上の場合以外で途中で磁気テープをはずすと，後の使

用上不便なことがあるので注意を要する。なおヘッダーブロックのサイズもリストバッファーのサイズ＋1ワードである。

テープのブロック構成をFig. 8 に，ヘッダーブロックの構成をFig. 9 に示す。

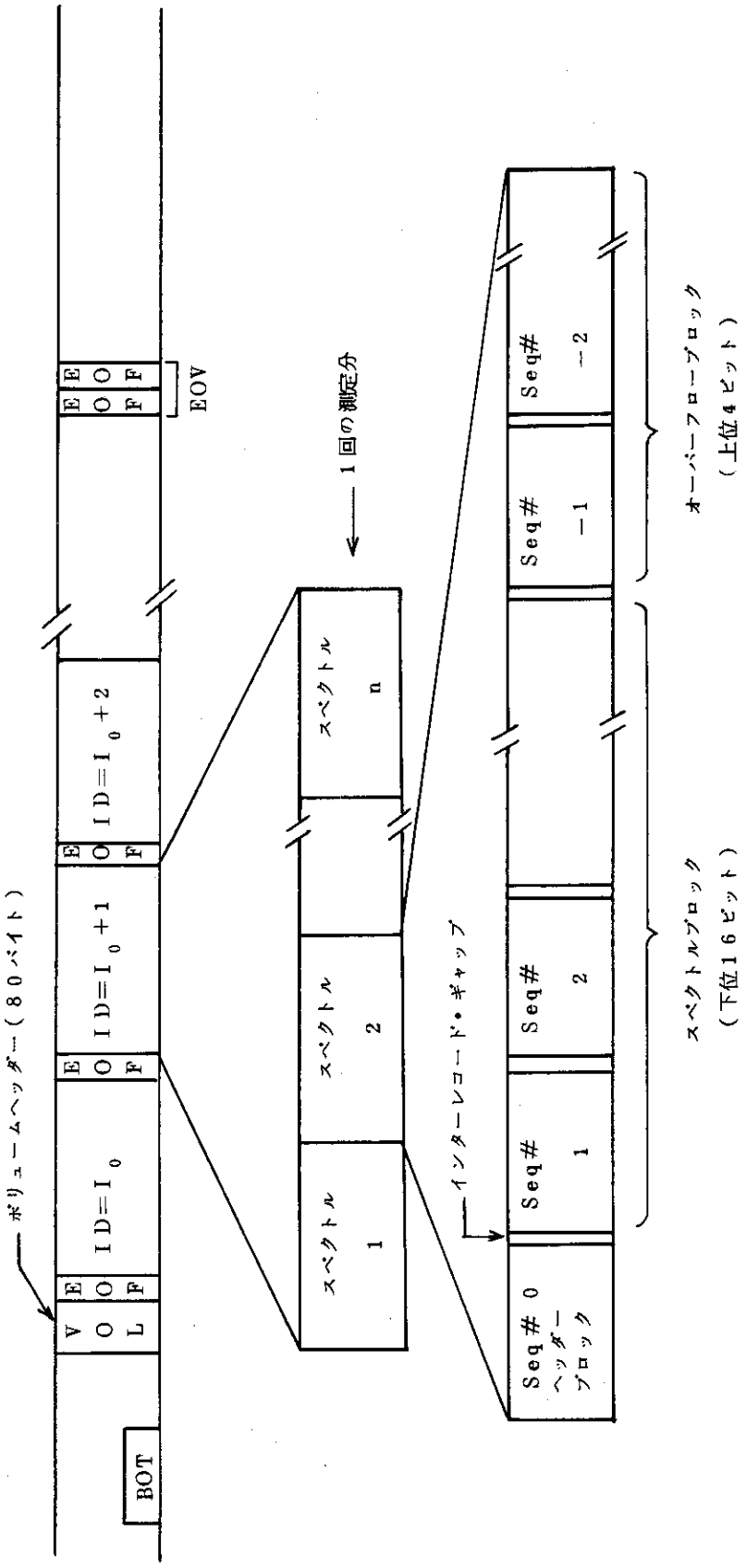


Fig. 4　スペクトル・ダンプ・テープのブロック構成

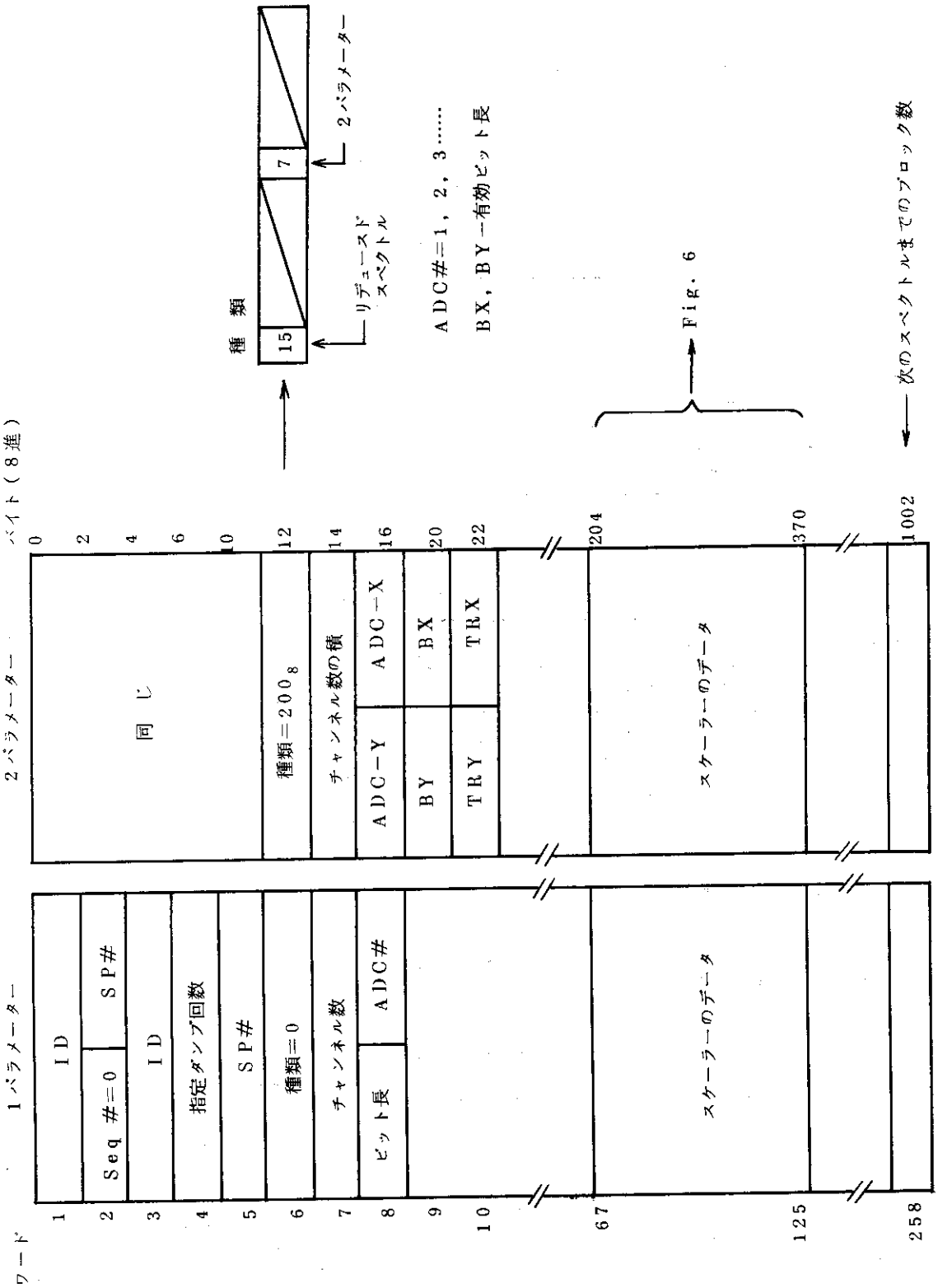


Fig. 5　PHAモードのヘッダーブロック

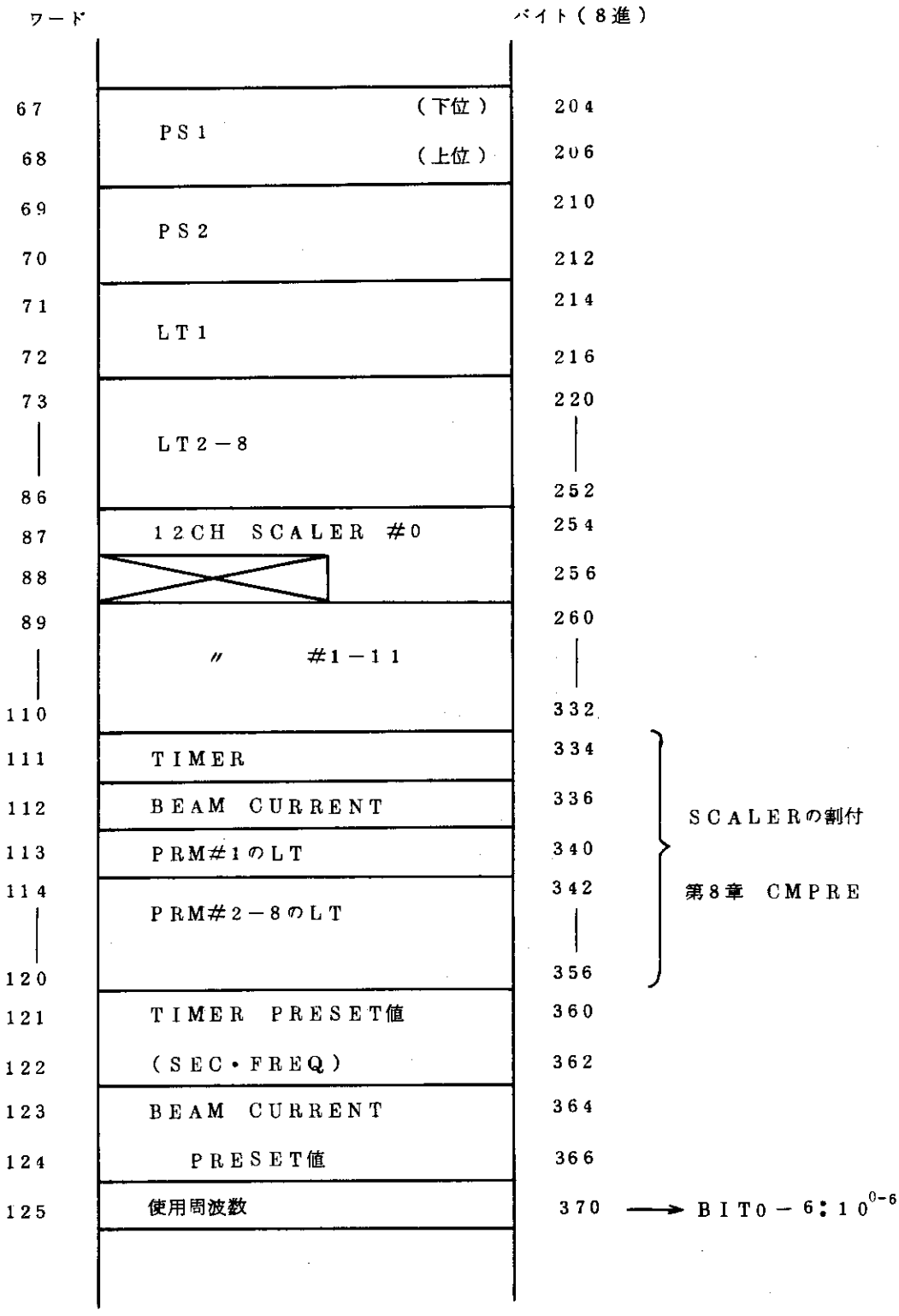


Fig. 6 ヘッダーブロックのスケーラーデータ

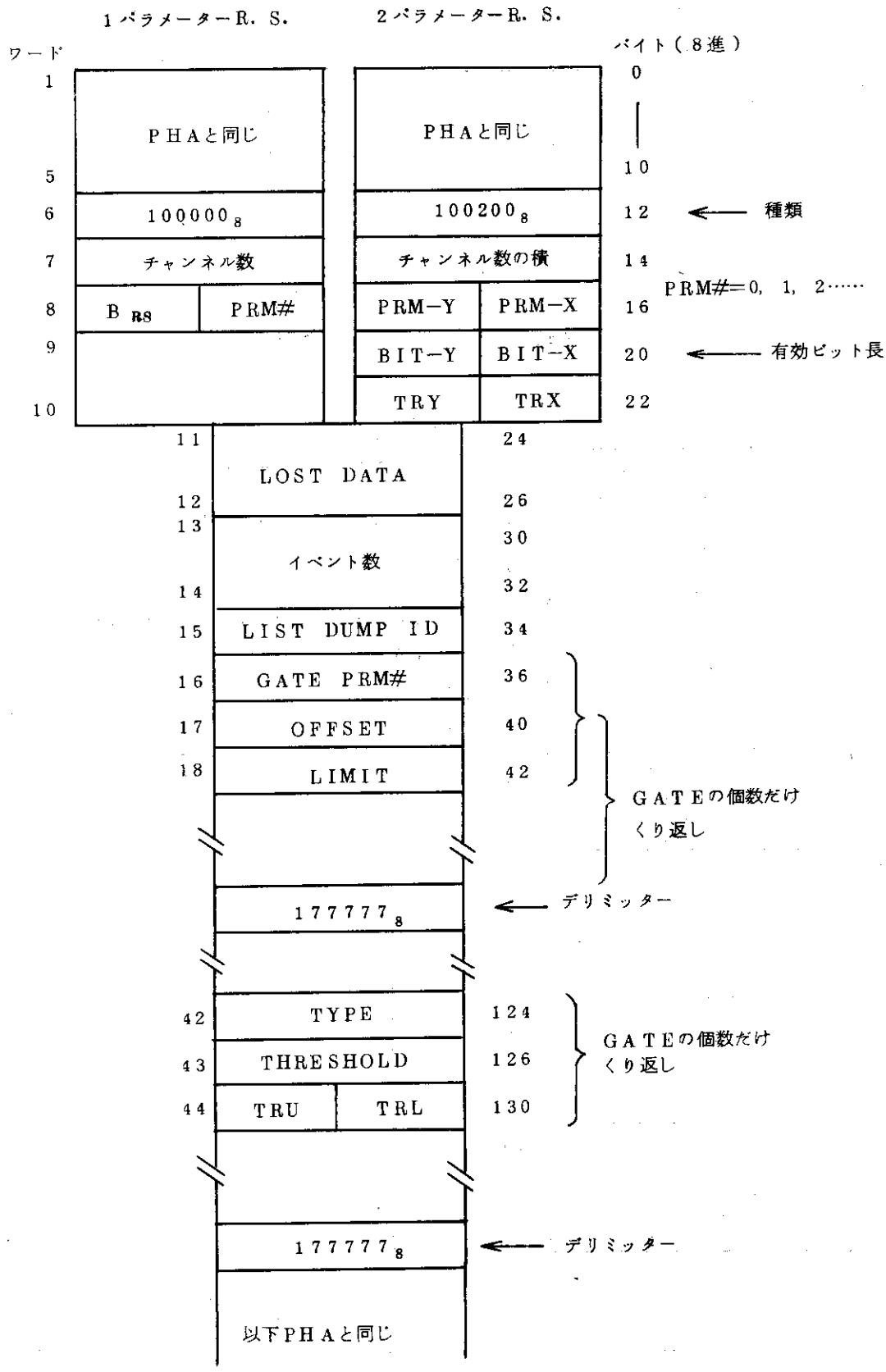


Fig. 7　リデュースドスペクトルのヘッダーブロック

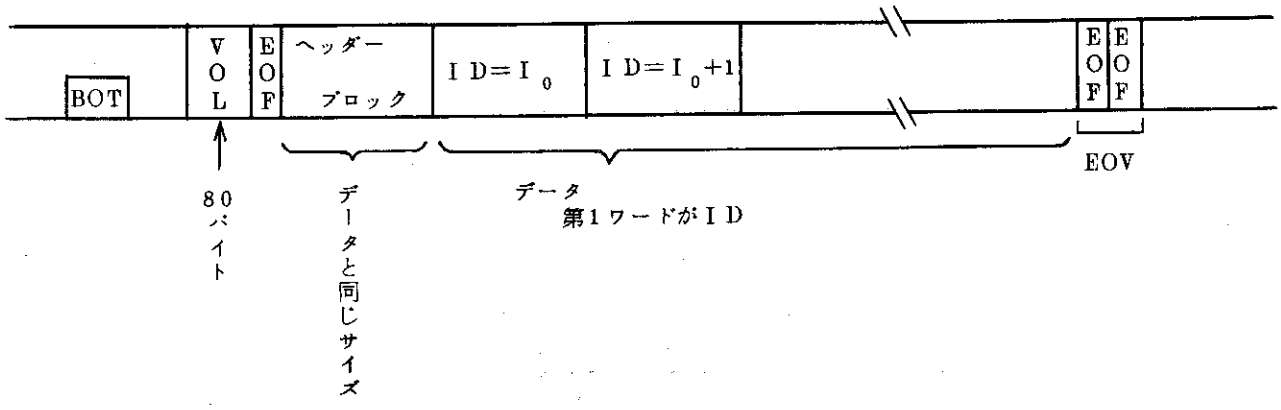


Fig. 8　リストダンプ磁気テープのブロック構成

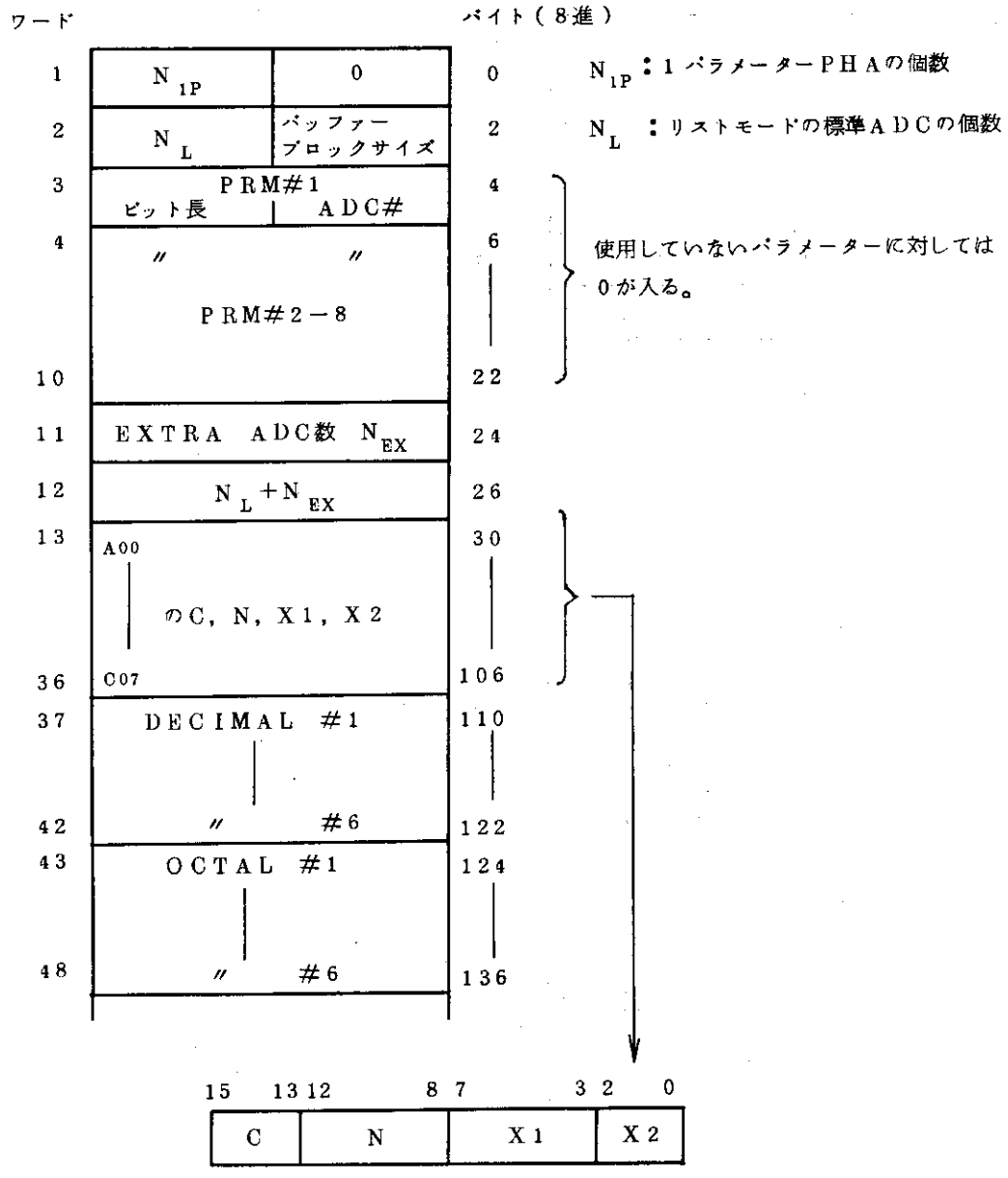


Fig. 9　リストダンプ磁気テープのヘッダーブロック

参　考　文　献

1)　菊地，富田，河原崎，大内，竹内，丸山：JAERI-M 9136，原研20MVタンデム加速器データ収集・処理システム

2)　BiRa 社：MBD-11 Programmer's Manual

3)　DEC社：PDP-11 Processor Handbook

4)　DEC社：MACRO-11 Reference Manual

5)　理経コンピューター社：原研20MVタンデム・データ収集システム （内部仕様書）