横須賀市博研報（自然）Sci. Rept. Yokosuka City Mus., (42): 27–56. Dec. 1994

パーソナルコンピュータによるホタル発光パターンの解析システム

牧野　徹*・鈴木浩文**・大場信義***

Computer analysis system for firefly flash patterns

MAKINO T.*, SUZUKI H.** and OHBA N.***

We developed a computer analysis system for firefly flash patterns. This system consisted of measuring hardware and analysis software. Video-recorded images of flashing fireflies was displayed on a TV monitor. Intensity of luminescence of firefly flashes was then detected by semiconductive photosenser. Electrically transformed signals pass through low pass filter and digitalised through analog digital board, and then fed to a personal computer. Sequential data of the firefly flashing lights were displayed chronologically as waves on a monitor. In this analysis, appropriate sections of the waves can be selected on the computer screen and displayed in greater details. These details can be printed out. Some wave data can be overlaid and be displayed sequentially. Furthermore, the wave data can be subject to spectral analysis using maximum entropy method.

はじめに

ホタルの発光パターンは種特有であり，配偶行動に深く関与し（OHBA, 1983），ホタルの発光コミュニケーションの詳細な解明や，発光行動の適応進化などについて研究する上で，その記録解析は不可欠である。筆者の一人である大場は室内や野外での様々なホタルの発光行動において発せられる発光信号の記録解析方法を段階的に改善してきた（大場，1985）。初期段階では目視観察で発光間隔をストップウオッチで測定した。発光間隔の短い種では誤差が大きかった。次に，8 mmまたは16mmシネカメラを用い，発光を録画する方法を導入したが，フイルム感度の不足，録画時間の制約，経費，録画技術に加え，アナログ的な発光信号に対して解析に不向きであった。次に考案したシステムは発光信号をフォトトランジスタで音声信号に変換し，増幅してテープレコーダに録音するものであった。この音声信号は電気的信号に変換されているので，ペンレコーダなどで波形を描くことが可能になった。しかし，この方法は発光しているホタルが静止し，光センサーでホタルの光信号を追従可能な場合に限られた。この点を解決するために，発光信号をイメージ・インテンシファイアー・チューブを装着させた高感度CCDカメラで録画した。この方法は音声・映像が同時に録画・録音されるために，飛躍的な情報量が記録可能となった。録画された光信号をモニター上に再生し，画像光信号を追従しながら受光増幅して，ペン

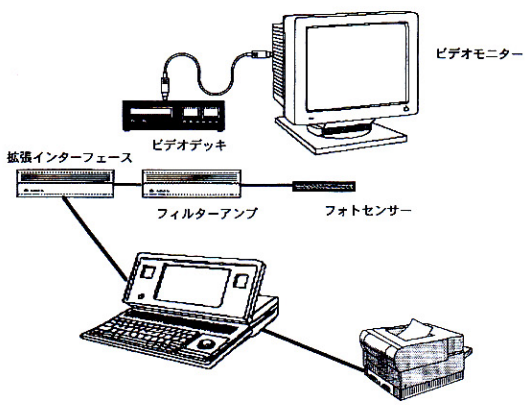
* オリンパス光学工業株式会社バイオメディカルリサーチセンター　Bio-medical Research Center, Olympus Optical Co. Ltd., Hachioji, Tokyo 192.
** 東京都立大学理学部自然史講座　Dept. Natural History, Facul. Sci., Tokyo Metropolitan University, Hachioji, Tokyo 192–03.
*** 横須賀市自然博物館　Yokosuka City Museum, Yokosuka, 238.
原稿受付　1994年9月1日．横須賀市博物館業績第459号.
キーワード：**発光パターン，解析システム，パーソナルコンピュータ，ホタル，ホタル科．** Key words: **flash pattern, analysis system, personal computer, firefly, Lampyridae**

レコーダに波形を描かせるシステムである。このシステムは飛翔発光するホタルや複数個体間の関連など，ホタルのあらゆる発光行動の記録解析にも対応し，発光パターン解析は進展した。しかし，このシステムでは，ペンレコーダで波形を描かせているために，明滅の早い光信号には追従が困難となるほか，描いた波形の比較や並び替え，変形，切り張り，さらに得られたデータ保存・呼び出しに課題が残っていた。そこで筆者らは，この点を解決するために，ビデオ録画したホタルの映像以降の解析処理方法を改善し，パーソナルコンピュータ（以後パソコンと省略する）を用いた新たな発光パターン解析システムを独自に構築したので，そのシステムの詳細とプログラムを付して報告する。

発光パターン解析システム

本システムは，あらかじめビデオカメラ等で撮影されたホタルの発光パターンを解析するものである。ビデオ録画したホタルの映像をモニター上に再生し，発光の強弱を光センサーを用いて電気信号に変える。この際モニター上の発光の軌跡を光センサーで追尾することになる。光センサーからの光電圧出力はフィルターアンプを通して増幅され，アナログ信号をディジタル信号に変換してパソコンに取り込まれて解析される。測定装置は第1図に示すように，ビデオデッキ，パソコン，フィルターアンプ，フォトセンサーから構成される。

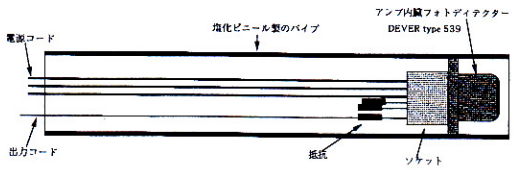
ビデオデッキ，パソコンは市販の製品が使え機能上の制約はあまりない。但し，解析プログラムがPC-9801シリーズ及び互換機で作動するためにIBM/PCやMacintoshでは動作しない。またフォトセンサーからのアナログ信号をディジタル信号に変換するためのA/Dコンバーターを拡張インターフェーススロットに挿入する必要があるので，ノートタイプのパソコンの場合はそれが必要となる。本システムではパソコンとしてPC-9801/NS (NEC)，拡張インターフェースユニットにNOTE-PAC(98)H-4A (CONTEC)，A/Dコンバーターに AD12-16T(98)H (CONTEC) を使用している。A/Dコンバーターは分解能が8ビットから12ビットで，変換速度が13 μsec 程度を満たしていれば基本的にどのメーカーの製品でもよい。しかし，このシステムで他社製品を使う場合はA/Dコンバーターの制御プログラム部分を変更する必要がある。



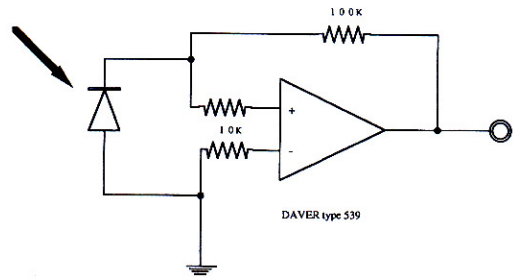
第1図　発光パターン解析システムの装置構成

フォトセンサーは基本的には光電変換を行うものであるから，フォトダイオードやCdSが使える。今回は感度，入手の容易さからDEVAR type 539（コロンビヤ貿易）を用いた。このフォトセンサーはフォトダイオードとプリアンプが一体となったもので，フォトダイオードによる電流変化は電圧変化に変換されて出力される。この素子を第2図のように塩化ビニール製のパイプにとりつけて使用する。第3図にその回路を示す。

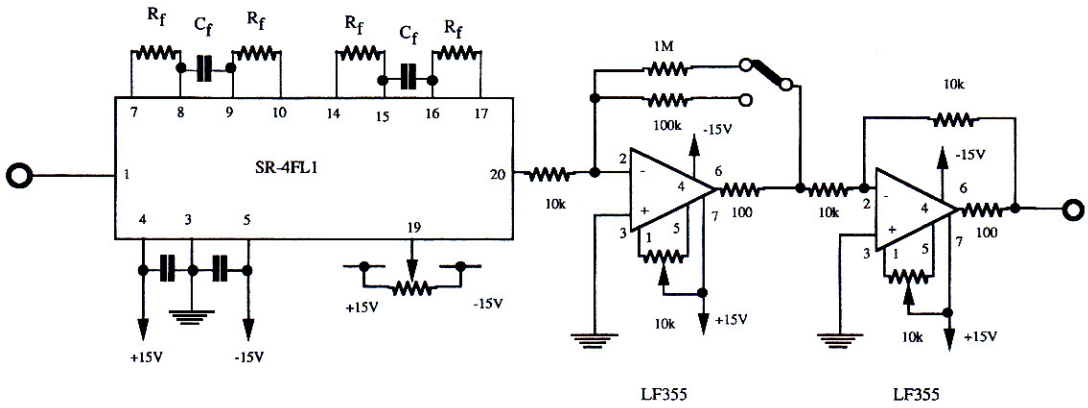
フィルターアンプは相当品（24 dB/octフィルター，差動アンプP-61，いずれもエヌエフ回路設計ブロック社）があるので自作する必要はないが，後述するように比較的簡単に自作できる。フィルターは，アナログ信号をサンプリングしディジタル信号に変換する際に生じる誤差を抑えるためのもので，アナログ波形にサンプリング周波数の1/2以下の周波数成分のみが含まれるようにする。本装置は10 Hz及び50 Hzにカットオフ周波数をもつローパスフィルターとアンプから構成されている。



第2図　フォトセンサーの構造.



第3図　フォトセンサーの回路図.



第4図　フィルターアンプの回路図.

フィルターはエヌエフ回路設計ブロック社のSR-4FL1型抵抗同調フィルターを用いた。このフィルター素子は第4図の回路図に示すRf（抵抗），Cf（コンデンサー）によって目的のカットオフ周波数(fc)が得られ，それらの関係は次の式で表される。

$$Rf = 159/\{(Cf + 0.01) \times fc\} \quad (K\Omega)$$

今，カットオフ周波数を10 Hzとし，コンデンサー容量が2.2 μFであるとすると

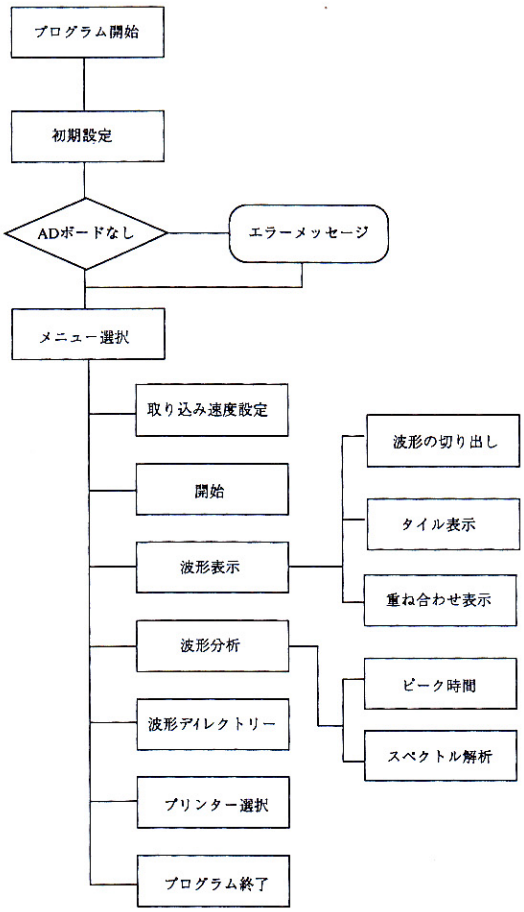
$$Rf = 159/\{(2.2 + 0.01) \times 10\} = 7.19457\,K\Omega$$

となる。但し，コンデンサーは抵抗に比べると容量の誤差が大きく誤差の少ないものを入手しにくいので，抵抗で補正できるようにする必要がある。コンデンサー容量の誤差が±5%あるとすると，Rfは6.85345から7.23056 KΩの範囲が必要となる。カットオフ周波数の精度をあげるには6.8 KΩの金属皮膜抵抗（誤差±1%）に1 KΩの可変抵抗器を直列に接続して使用する。カットオフ周波数を変える時には同様の計算にしたがってコンデンサーと抵抗の値を決め，ロータリースイッチでそれらに切り替える。

フォトダイオードからの出力は既にプリアンプで増幅されているので，増幅率としては1×10^2から1×10^3が必要である。尚，このフィルターアンプは±15Vで動作する。電源にはBCM-15/200（ディテル社）の2出力モジュール電源を使用した。

発光パターン解析プログラム

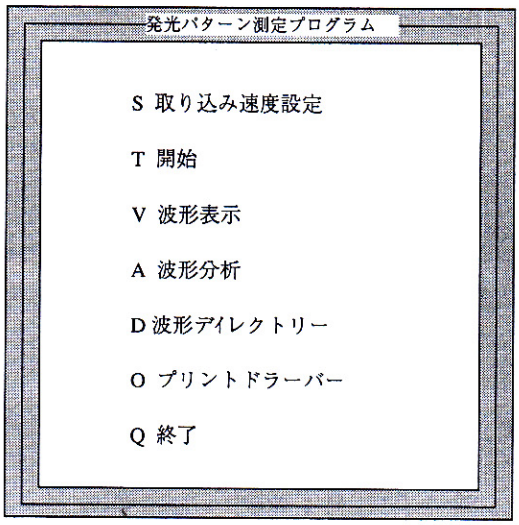
この解析プログラムは，ビデオ映像からの発光強度をディジタル信号に変換してパソコンに取り込む装置の制御とそのデータを解析する部分から構成されている。第5図はこのプログラム全体の構成で，測定した波形の切り出しや編集，時間軸をそろえた複数波形の表示，波形



第5図　発光パターン解析プログラムの構成.

のピーク時間の読み取り，波形のスペクトル解析などができ，いずれの結果もプリンターで印刷できる。

本プログラムを起動すると第6図に示すメインメニ



第6図 発光パターン解析プログラムのメインメニュー.

ューが表示され，矢印キーでカーソルを移動させ各メニューを選択する。

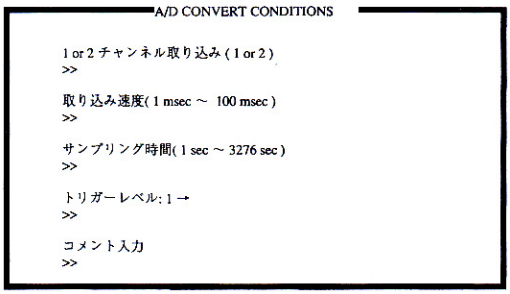
取り込み速度設定

取り込み速度設定を開くと第7図に示す画面が表示され測定条件を設定する。

チャンネル取り込みでは使用するフォトセンサーの数(1または2)を入力する。本システムでは同時に2チャンネルでの取り込みも可能である。

取り込み速度は1 msecから100 msec (10 Hzから1 kHz)の範囲で設定できる。ホタルの発光間隔は速いもので0.1 sec程度なので50 msecより短い間隔でデータを取り込んだ方が良い。通常フィルターアンプのカットオフ周波数は10 Hzでよいが，発光間隔が0.1 sec以下の場合は取り込みの時間間隔をもっと短くしフィルターのカットオフ周波数を50 Hzにする。

サンプリング時間ではデータを取り込む時間を設定する。この解析プログラムで取り扱えるデーター総量は32768ポイントであるので，この値をオーバーすると自動的に再入力となる。サンプリング時間/取り込み速度≦32768の関係を守る必要がある。



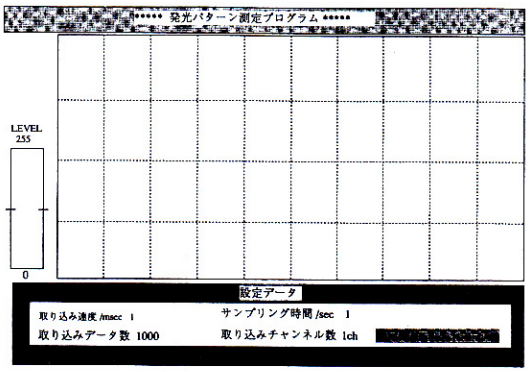
第7図 発光パターン測定の条件設定メニュー.

トリガーレベルは0から255の範囲で設定できる。測定の開始はリターンキーで行うが，トリガーレベルを設定することによってフォトセンサーにトリガーレベル以上の光が入力されると測定が開始される。この機能は1つの録画シーンから複数個体の発光パターンを記録する時に便利である。すなわちビデオ編集によって発光パターンを測定したいシーンの直前に白いマークを画面の隅に入れておく。画面のその部分にフォトセンサーを置いておきビデオテープを再生させると，白いマークが出てきた瞬間から測定が開始される。これによって同じシーンにおける複数個体の測定開始時間をそろえることができる。リターンキーによって測定を開始する場合は200程度の高い値に設定しておけば良い。

コメントは128文字まで入力できる。尚，この取り込み速度設定のモードは再び設定しない限り前の条件が有効となるので，ここで入力したコメントも次の測定に有効となる。これは撮影日時等の同じ条件を再び入力することを避けるためのもので，測定終了後データを保存するときに再びコメントの入力ができるようになっている。

開始

開始を開くと第8図に示す画面が表示される。この状態でリターンキーを押すか，トリガーレベル以上の光信号が入力されると測定が開始される。画面中央の格子の描かれている部分にフォトセンサーから入力されたデータが表示される。画面下の部分には測定条件が表示され，画面左にはトリガーレベルが表示される。測定終了後に測定データを保存するかしないかを選択し，ファイル名とコメントを入力するとDATの拡張子が付けられて保存される。



第8図 発光パターンの測定時の表示画面.

波形表示

このモードでは測定した発光パターンの表示，波形の切り出しや編集，印刷を行う。このモードを開くと第9図の左に示した波形表示のメニューが表示される。

まず表示するデータのファイルを選択する。ファイル選択を開くとファイルパス入力と聞いてくるのでデータファイルが格納されているドライブ，ディレクトリー及びファイル名を示すワイルドカード（例えば*など）を入力する。すると新たなウィンドーが開き指定したファイル名のリストが表示されるので矢印キーを使い表示したいファイルを選択する。同時に表示可能なファイル数は4つ以内である。ファイル選択はESCキーで閉じる。

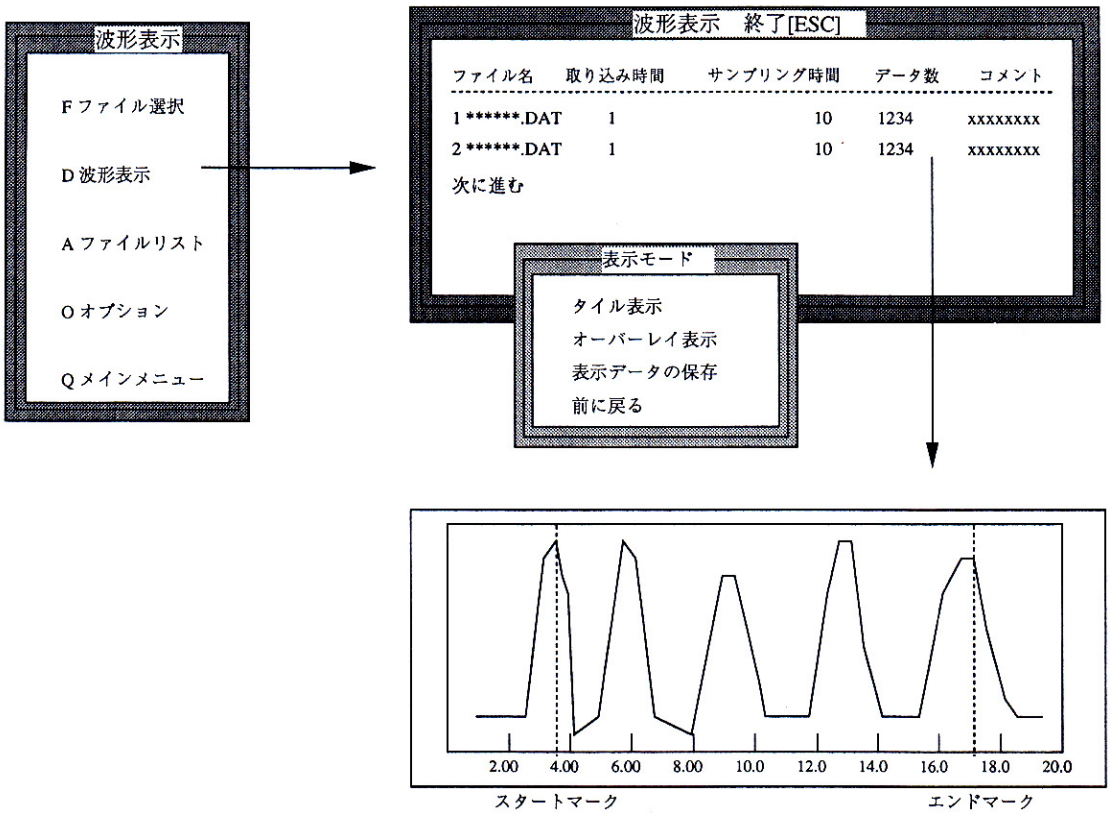
ファイル選択が終了すると第9図に示す波形表示の画

ファイル選択が終了すると第9図に示す波形表示の画面に戻り，そこで波形表示を開くと第9図右上に示すファイルの測定条件やコメントが表示される。矢印キーによってカーソルを動かし表示するファイルを選択すると第9図右下に示す波形データが表示されるので矢印キーで切り出したい時間範囲を決定する。切り出しの始点および終点はF1キーでマークする。カーソルは矢印キーとSHIFTキーを同時に押すことで速く移動させることができる。波形の切り出しはESCキーで終了し，第9図右上に示す画面に戻る。「次に進む」を開くと第9図中程に示したウィンドーが開く。タイル表示は時間軸をそろえてそれぞれの波形を表示し，オーバーレイ表示では選択した全てのデータを重ね合わせて表示する。いずれの表示も時間軸は1番目に選択したファイルデータの時間にそろえられ，F10キーで印刷ができる。また切り出しをした新たな波形はオリジナルの波形とは別のファイル名を付けて保存できる。

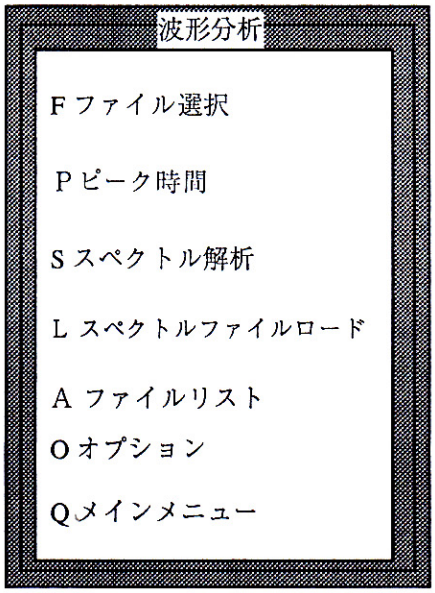
波形表示メニューのファイルリストでは，選択されたファイルの測定条件とコメントを表示する。オプションでは，ファイル選択の際表示されるファイルの並び方をファイル名順，ファイルサイズ順，日付順から選択できる。

波形分析

このモードでは測定したデータや加工した波形からのピーク時間の読み取りと最大エントロピー法によるスペクトル解析が可能である。このモードを開くと第10図に示す波形分析のメニューが表示される。



第9図　波形データの表示メニュー．



第10図 波形データの分析メニュー.

波形表示のモードと同様に解析するファイルを選択する。ピーク時間のモードでは選択したそれぞれの波形データが表示され，矢印キーでカーソルを移動させるとスタートポイントからの時間が表示されるので読み取りたい時間範囲をF1キーでマークする。

スペクトル解析のモードでは，選択されたファイルデータが表示されるのでスペクトル解析するファイルを選んでリターンキーを押すと波形データが画面に表示される。次に矢印キーでカーソルを移動させてスペクトル解析する波形データの領域を決める。始点と終点はF1キーで決定する。但し，スペクトル解析が可能なデータ点数は512点以内であるため，選択された波形データ点数がこの値を超えると自動的にスタートポイントから512点までのデータに変更される。解析範囲が決定したらESCキーまたはリターンキーで最大エントロピー法によるスペクトル解析が開始する。このプログラムは数値演算プロセッサーの有無を自動的に判断して計算するが，計算にはかなりの時間を要するため画面左に計算中の点滅表示をしている。尚，この計算プログラムは日野(1977)を参考に作成した。計算終了後スペクトルが画面に表示されF1キーで保存できる。保存するスペクトルデータには自動的にSPCの拡張子が付けられる。

ここで用いているスペクトルとは波形データに含まれる周期性の頻度を表わしたものである。仮に1 Hzの所に鋭いピークが表われていた場合は波形データには周期が1秒の発光が含まれていたことになる。またピークが表われずになだらかなカーブを描く場合には周期性がないとこになり，複数の鋭いピークが表れる場合には複数の発光周期が含まれていることを意味する。

保存したスペクトルファイルの表示，印刷はファイル選択を開きスペクトルデータを選択し，スペクトルファイルロードを開く。

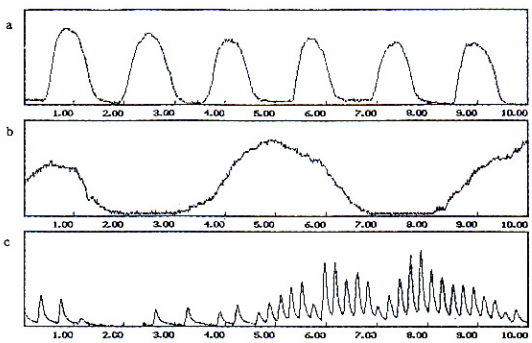
ファイルリスト，オプションのモードは波形表示のモードと同じである。

プリントドライバー

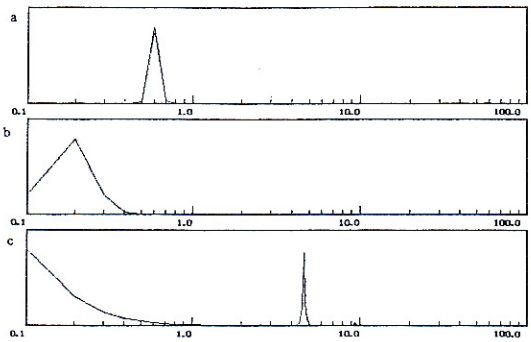
使用するプリンターを選択する。デフォルトはPC-PR201であり，PLOTERを選択するとグラフィックプロッターが使用できる。

測定例

第11図のaとbはゲンジボタルの雄が雌を探しながら飛んでいるときの発光パターンで，aは京都産，bは千葉産のものである。第11図cは沖縄産のヤエヤマボタルの雄の発光パターンである。測定は50 Hzのローパスフィルターを通して10 msec間隔(100 Hz)で行った。発光の間隔は，波形分析モードのピーク時間を開くとカーソルが出てくるので，それを発光強度のピークからピークまで合わせることによって測定できる。ゲンジボタルは中部山岳地帯を境にして西日本と東日本で発光の間隔や活動習性が異なっている（OHBA, 1984; 大場，1988; 1991)）。この例の場合，発光間隔は西日本の京都で約1.7秒，東日本の千葉では約4.5秒であり京都産のゲンジボタルに比べて千葉産のゲンジボタルの発光は極めてゆっくりで間延びしていることが分かる。これに対して



第11図 ゲンジボタルとヤエヤマボタルの発光パターン.
a.ゲンジボタル（京都産），b.ゲンジボタル（千葉産），c.ヤエヤボタル（沖縄産）．横軸の単位は時間（1目盛1秒），縦軸は相対的発光強度.



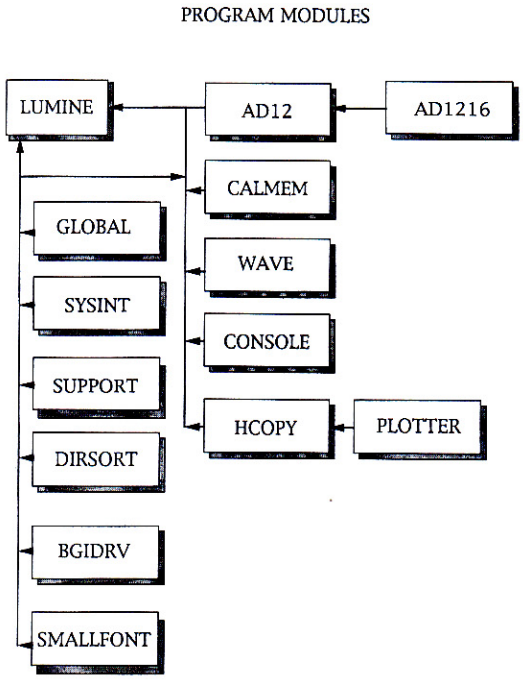
第12図　ゲンジボタルとヤエヤマボタルの発光パターンのスペクトルパターン.
a.ゲンジボタル（京都産），b.ゲンジボタル（千葉産），c.ヤエヤマボタル（沖縄 産）
横軸の単位は周波数（Hz），縦軸は相対強度.

ヤエヤマボタルの雄の発光は非常に短い間隔で閃光を放つ。この例では閃光の間隔は約0.2秒であった。

この3種類の発光パターンを波形分析のモードでスペクトル解析をした結果が第12図で，a，b，cはそれぞれ京都産のゲンジボタル，千葉産のゲンジボタル，沖縄産のヤエヤマボタルのスペクトルパターンである。発光パターンのスペクトルを解析するということは，その発光パターンにどのような周期の波形の繰り返しがあるのかを見ることである。第12図の横軸は波形に含まれる波の周波数で，縦軸はその周波数が含まれている割合を表している。西日本型の京都産のゲンジボタルでは0.6 Hz (1.7 sec) のところにピークが出ている。これに対して東日本型の千葉産のゲンジボタルでは0.2 Hz (5 sec) のところにピークがありその裾は広がっている。これは非常に緩やかな波であることを示している。ヤエヤマボタルでは4.5 Hz (0.2 sec) のところに鋭いピークがあり，パルス状の光であることを示している。また0.1 Hzから0.5 Hzにかけて山がみられるが，これは第11図cの波のベースにあるゆっくりした変動を表している。ビデオによってホタルの飛翔を録画すると，ホタルの移動によって発光の強度はこのように変わってしまうことがあるので，発光パターンのスペクトルを解析するときはそれに最適な領域を選択する必要がある。

プログラムリスト

第1表に本システムのプログラムリストを示す。第13図は本システムを構成する各プログラムモジュール間の関係を示すもので，各プログラムモジュールの機能は次



第13図　ホタル発光パターン解析システムを構成する各プログラムモジュール.

の通りである。

LUMINE	メインプログラム
GLOBAL	全プログラムモジュールに共通な変数の定義
SYSINT	A/D変換ボードの存在確認と初期化および変数の初期化
SUPPORT	全プログラムモジュールの下請け
DIRSORT	ファイルリストの表示
BGIDRV	画面にグラッフィクスを表示するグラフィクドライバー
SMALLFONT	グラフィクドライバーが使用するフォント
AD12	A/D変換ボードの制御
AD1216	A/D変換ボードの制御プログラムの下請け
CALMEM	最大エントロピー法によるスペクトル解析
WAVE	波形表示およびピーク時間解析
CONSOLE	テキスト形ウインドウシステム
HCOPY	画面のハードコピー
PLOTTER	X-Yプロッターの実行

引用文献

日野幹雄 1977. スペクトル解析. 210–236. 朝倉書店.

大場信義 1985. 発光シグナルの記録とその解析法. 植物防疫, **39**(9):46–51.

大場信義 1988. ゲンジボタル. 198ページ. 文一総合出版.

大場信義 1991. ゲンジボタルの遺伝子東西で異なる. 遺伝, **45**(10):8–9.

OHBA N. 1983. Studies on the communication system of Japanese fireflies. *Sci. Rept. Yokosuka City Mus.*, (30): 1–62, pls. 1–6.

OHBA N. 1984. Synchronous flashing in the Japanese firefly, *Luciola cruciata* (Coleoptera:Lampyridae). *Sci. Rept. Yokosuka City Mus.*, (32): 23–33, pls. 1–8.

第1表　ホタル発光パターンの解析システムを構成するプログラムリスト.

```
LUMINE

{*****************************************************************
          蛍発光パターン解析プログラム

          オリンパス光学工業㈱　ＢＲＣ研究Ｇ
          作成　牧野　繁　平成元年１２月１２日
          Version 0.10:   1989/12/02
          Version 0.50:   Revised on 1989/12/14
          Version 1.00:   Revised on 1990/01/11
          Version 1.10:   Revises on 1992/3/10
          Version 2.00:   Revises on 1993/3/15
*****************************************************************)
{$M 16000,128000,655360}
{$R+}
{$N+}
program Hol(input,output);
{$F+}
uses overlay, dos, crt, graph, global,console, ad12,
     dirsort, sysint, support, wave, calcmem;
{$O wave}
{$O sysint}
{$O support}

const
      OverName:string[80] = 'LUMINE.EXE';
{ OverName:string[80] = 'LUMINE.OVR';}

  (* メインメニューリスト *)
  MainMenu:MenuList=(( Name:'発光ﾊﾟﾀｰﾝ測定ﾌﾟﾛｸﾞﾗﾑ';Command:ESC), { 1 }
      (Name:'取り込み速度設定' ; Command:'S'),                  { 2 }
      (Name:'開　始' ; Command:'I'),                            { 3 }
      (Name:'波形表示' ; Command:'W'),                          { 4 }
      (Name:'波形分析' ; Command:'A'),                          { 5 }
      (Name:'波形ディレクトリー' ;Command:'D'),                 { 6 }
      (Name:'プリンタドライバ' ;Command:'O'),                   { 7 }
      (Name:'終　了' ;Command:'Q'),                             { 8 }
      (Name:'' ;Command:#0),
      (Name:'' ;Command:#0));

  (* ユーティリティーメニューリスト *)
  SubQMenu:MenuList = (( Name: ''; Command: ESC),
      (Name: '保存する' ; Command:'Y'),
      (Name: '保存しない' ; Command:'N'),
      (Name: '' ; Command:#0),
      (Name: '' ; Command:#0),
      (Name: '' ; Command:#0),
      (Name: '' ; Command:#0),
      (Name: '' ; Command:#0),
      (Name: '' ; Command:#0));

  SubAMenu:MenuList = (( Name: ' ﾋﾞｰｸ時間解析 ': Command: ESC),
      (Name: 'ﾌｧｲﾙ選択' ; Command:'F'),
      (Name: 'ﾋﾟｰｸ時間間' ; Command:'P'),
      (Name: '2ﾍﾟｸﾄﾙ解析' ;Command:'S'),
      (Name: '2ﾍﾟｸﾄﾙﾌｧｲﾙｾｰﾌﾞ' ;Command:'L'),
      (Name: 'ﾌｧｲﾙﾘｽﾄ' ; Command:'A'),
      (Name: 'ﾎﾟｲﾝﾄｼｮｯﾄ' ;Command:'O'),
      (Name: 'ﾒｲﾝﾒﾆｭｰ' ; Command:'Q'),
      (Name:'' ;Command:#0),
      (Name:'' ;Command:#0));

  SubFMenu:MenuList = (( Name: ' 波形表示 ': Command: ESC),
      (Name: 'ﾌｧｲﾙ選択' ; Command:'F'),


       (Name: '波形表示' ; Command:'D'),
       (Name: 'ﾌｧｲﾙﾘｽﾄ' ; Command:'A'),
       (Name: 'ﾎﾟｲﾝﾄｼｮｯﾄ' ; Command:'H'),
       (Name: 'ﾒｲﾝﾒﾆｭｰ' ; Command:'Q'),
       (Name:'' ;Command:#0),
       (Name:'' ;Command:#0),
       (Name:'' ;Command:#0),
       (Name:'' ;Command:#0));

  SubPMenu:MenuList = (( Name: ' ﾌﾟﾘﾝﾀﾄﾞﾗｲﾊﾞｰ ': Command: ESC),
      (Name: 'PC-PR201' ; Command:'P'),
      (Name: 'VP-130K' ; Command:'V'),
      (Name: 'NW-9950' ; Command:'N'),
      (Name: 'PLOTTER' ;Command:'X'),
      (Name: 'ﾒｲﾝﾒﾆｭｰ' ; Command:'Q'),
      (Name:'' ;Command:#0),
      (Name:'' ;Command:#0),
      (Name:'' ;Command:#0),
      (Name:'' ;Command:#0));

        SubSMenu:MenuList = (( Name: ' ﾌｧｲﾙｿｰﾄﾓｰﾄﾞ ' ; Command: ESC),
                    (Name: 'ﾌｧｲﾙ名順' ; Command:'N'),
                    (Name: 'ﾌｧｲﾙﾀｲﾑ順' ; Command:'S'),
      (Name: 'ﾌｧｲﾙ作成順' ; Command:'T'),
      (Name: 'ﾒｲﾝﾒﾆｭｰ' ; Command:'Q'),
      (Name:'' ;Command:#0),
      (Name:'' ;Command:#0),
      (Name:'' ;Command:#0),
      (Name:'' ;Command:#0),
      (Name:'' ;Command:#0));

var
  No: ShortInt;
  Cond: TADcond;
  TriggerLevel: word;

procedure Scaler(cond:TADcond);
 var
  i: Word;
  s: String;
begin
 HeadFile(MEASURE);
 Selcolor(graph.BROWN);
 SetLinestyle(SOLIDLN,0,NORMWIDTH);
 SetTextstyle(SMALLFONT, HORIZDIR, 4);
 SetTextJustify(RIGHTTEXT, TOPTEXT);
 Rectangle(50, 19, 635, 275);
 Rectangle(51, 20, 636, 276);
 SetLinestyle(DOTTEDLN,0,NORMWIDTH);
 for i := 1 to 9 do
   line(50+58*i, 19, 50+58*i, 275);
 for i := 1 to 3 do
   line(50, 19+64*i, 635, 19+64*i);
 SetColor(graph.MAGENTA);
 for i:=1 to 10 do begin
   line(50+58*(i-1), 275, 50+58*(i-1), 275-5);
   Str((1.0*cond.Time/10*i):3:1, s);
   OutTextxy(50+58*i, 275+3, s);
   OutTextxy(51+58*i, 275+3, s);
end;
OutTextxy(50, 275+13, '0.0');
OutTextxy(51, 275+13, '0.0');

Selcolor(graph.BROWN);
SetLinestyle(SOLIDLN,0,NORMWIDTH);
Rectangle(10,150,40,250);
Rectangle(11,151,41,251);
 line(5, 200, 10, 200);   (* 中心線 *)
 line(5, 201, 10, 201);
 line(45, 200, 40, 200);
 line(45, 201, 40, 201);
 SetTextJustify(CENTERTEXT, BUTTOMTEXT);
 SetColor(graph.MAGENTA);
 OutTextxy(25, 148, '255');
 OutTextxy(28, 148, '255');
 SetColor(graph.BROWN);
 OutTextxy(25, 148-TextHeight('W'), 'Level');
 OutTextxy(28, 148-TextHeight('W'), 'Level');
 SetTextJustify(CENTERTEXT, TOPTEXT);
 SetColor(graph.MAGENTA);
 OutTextxy(25, 252, '0');
 OutTextxy(28, 252, '0');
 SetColor(graph.BROWN);
 line(10, 250-trunc(1.0*TriggerLevel/4096*100+15), 40, 250-trunc(1.0*TriggerLevel/4096*10
0+16));
 SetFillStyle(LTSLASHFILL, graph.GREEN);
 FLoodFill(12, 251-trunc(1.0*TriggerLevel/4096*100+16), graph.BROWN);
 Selcolor(graph.WHITE);

{* テキストウインドーのオープン *}
 SetFrame(crt.cyan+crt.reverse,crt.green+crt.reverse,' 設定ﾃﾞｰﾀ ');
 SetWindow(2, 20, 69, 23); OpenWindow(1);
 CrtCursor(nodispcursor);
 Textcolor(crt.white);
 gotoxy(1, 1); write('取り込み速度/msec ');
 gotoxy(19, 1); write(cond.Duration);
 gotoxy(26, 1); write('ﾄﾘｶﾞｰﾎﾟｲﾝﾄ時間/sec ');
 gotoxy(45, 1); write(cond.Time);
 gotoxy(1, 2); write('取り込みﾃﾞｰﾀ数');
 gotoxy(19, 2); write(cond.SampleNum);
 Textcolor(crt.RED);
 gotoxy(26, 2); write('取り込みﾁｬﾝﾈﾙ数');
 Textcolor(crt.white);
 gotoxy(45, 2); if DualChannel=False then write('1ch') else write('2ch');
 Textcolor(crt.YELLOW); TextBlink(1);
 gotoxy(49,2); write('測定開始 [ﾀｰﾝﾄｰﾝ]');
 TextBlink(0);
end;{ Scaler }


function CheckStr( st: String):boolean;
 var
  i: Byte;
 begin
  CheckStr := FALSE;
  for i := 1 to Length(st) do
   if ( st[i] in ['0'..'9'] ) = FALSE then Exit;
  CheckStr := TRUE;
 end;

function CheckRange( x: Longint; p1, p2: Longint):boolean;
 begin
  CheckRange := FALSE;
  if ( x >= p1 ) and ( x <= p2 ) then CheckRange := TRUE;
 end;


procedure SetCondition(var cond: TADcond); (* 取り込み設定 *)
 var
  Tstr: String;


    V: Word;
    Code: Integer;
          Temp: Longint;
  begin
   SetFrame(crt.green,crt.white+crt.reverse,'A/D CONVERT CONDITION');
   SetWindow(22,3,59,17): OpenWindow(2);
   textcolor(crt.white);
   crtcursor(crt.dispcursor);
   gotoxy(3,1): write('1 or 2ﾁｬﾝﾈﾙ取り込み(1 or 2)');
   gotoxy(4,2): write('>> ');
  {$R-}
   repeat
     goloxy(7,2): readln(Tstr);
     Val(Tstr, V, Code);
   until (CheckStr(Tstr) and ( Code = 0 ) and  CheckRange(V, 1, 2));
  {$R+}
   if V=2 then DualChannel:=True else Dualchannel:=FALSE;
   with cond do
    begin
     gotoxy(3,4): write('取り込み速度( 1 msec ～ 100 msec )');
     gotoxy(4,5): write('>> ');
  {$R-}
     repeat
       goloxy(7,5): readln(Tstr);
       Val(Tstr, V, Code);
     until (CheckStr(Tstr) and ( Code = 0 ) and  CheckRange(V, 1, 100));
  {$R+}
     Duration := V;

     goloxy(3,7): write('ﾄﾘｶﾞｰﾎﾟｲﾝﾄ時間( 1 sec ～ 3276 sec )');
     gotoxy(4,8): write('>> ');
  {$R-}
      repeat
       goloxy(7,8): readln(Tstr);
       Val(Tstr, V, Code);
         Temp := Round( 1000.0 / Duration * V + 0.5 );
      until (CheckStr(Tstr) and ( Code = 0 ) and  CheckRange(Temp, 10, 32766));
  {$R+}
     Time := V;
     SampleNum := Round( 1000.0 / Duration * Time + 0.5 );

     goloxy(3,10); write('ﾄﾘｶﾞｰﾚﾍﾞﾙ: ',TriggerLevel,' ->');
     gotoxy(4,11): write('>> ');
  {$R-}
     repeat
       goloxy(7,11): readln(Tstr);
       Val(Tstr, V, Code);
     until (CheckStr(Tstr) and ( Code = 0 ) and  CheckRange(V, 0, 255));
  {$R+}
     TriggerLevel := V;

     goloxy(3,12): write('ｺﾒﾝﾄ入力');
     gotoxy(4,13): write('>> ');
     readln(Tstr);
     Comment := Tstr;
    end; { with }
   closewindow(2);
  end; { SetCondition }


procedure StartMeasure(cond: TADcond);
 procedure delay(count:integer);
  var
   i:integer;
```

```
begin
  for i:=1 to count do;
end;

var
 Temp1, Temp2: Word;
 ch:char;
 i: Word;
 No: ShortInt;
 x: single;
 FOREVER: boolean;
begin
 SetVisualPage(0);
 ClearDevice;
 Scaler(cond);
       if ADSTAT=0 then
        begin
                        ADparamSet(cond); {* タイマーセット *}
        end;
 FOREVER := TRUE;
 while FOREVER do
  begin
        if ADSTAT=0 then {* A/Dｶﾝｾｯﾄされてたらｽﾀｰﾄ *}
   if UpTrigger >= (TriggerLevel*10) then FOREVER := FALSE;
   if Keypressed then
    begin
     ch := ReadChar;
     case ch of
      ESC: begin
           CloseWindow(1);
           ClearViewport;
           Exit;
          end;
      RET:  FOREVER := FALSE;
     end;
    end;
  end;

 if ADSTAT=1 then begin
  CloseWindow(1);
  ClearViewport;
  SetVisualPage(1);
            Exit;
 end;
 goloxy(49,2): write('
 Buzzer(50);
 i := 1;
 x := 585/cond.SampleNum;
 if DualChannel then
        begin
                timerstart;
   repeat
                        GetAD3(Temp1, Temp2, $01D0, 1, 2);
                        triggerreset;
     Buffer[1]^[i] := Temp1;
     Buffer[2]^[i] := Temp2;
     PutPixel(50+Trunc(i*x), 275-Trunc(0.0625*Buffer[1]^[i]), graph.GREEN);
     PutPixel(50+Trunc(i*x), 275-Trunc(0.0625*Buffer[2]^[i]), graph.RED);
     inc(i);
                until i > cond.SampleNum;
                timerstop;
        end
 else
  repeat
     ADdataIn(Temp1,1,1);


     Buffer[1]^[i] :  Temp1;
     PutPixel(50+Trunc(i*x), 275-Trunc(0.0625*Buffer[1]^[i]), graph.GREEN);
     inc(i);
  until i > cond.SampleNum;
 With ADcond[1]^ do begin
   Duration := cond.Duration;
   Time := cond.Time;
   SampleNum := cond.SampleNum;
   Comment := cond.Comment;
 end;
 SetColor(graph.WHITE);
 CloseWindow(1);
 No:=1; MenuPosition(2,15);
 MenuAttr(crt.red+crt.reverse,crt.green+crt.reverse,crt.white,crt.white+crt.reverse);
 repeat
  No:=SelectMenu(SubQMenu,No);
  case No of
   1: begin
       WaveSave;
       if DualChannel then begin
        for i:= 1 to ADcond[1]^.SampleNum do Buffer[1]^[i] := Buffer[2]^[i];
        WaveSave;
       end;
       No := 0;
      end;
   2: No := 0;
  end;
 until No=0;
 CrtCursor(nodispcursor);
 TextColor(crt.white);
 ClearViewPort;
 SetVisualPage(1);
 MenuPosition(3,4);
 MenuAttr(crt.green+reverse,crt.yellow+reverse,crt.white,crt.white+reverse);
end; { StartMeasure }


procedure Browser;

 function ADcondLoad:boolean;
 var
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  Name: PathStr;
  F: Integer;
  RN: Word;
 begin
       ADcondLoad := TRUE;
  FSplit(Path, D, N, E);
  for I := 1 to Num do begin
                Name := D + Dir[I-1].Name;
    Assign(F, Name);
    Reset(F, 1);
    BlockRead(F, ADcond[1]^, SizeOf(TADcond), RN);
    if RN<>SizeOF(TADcond) then begin
                        ADcondLoad := FALSE;
                        Close(F);
                        exit;
                end;
    Close(F);
   end;
  end;

 var
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  ch: Char;
  i: Integer;
  S: String[20];
 begin
  GetFile(Path);
  if not ADcondLoad then begin
                Buzzer(10);
                exit;
        end;
  Set(Frame(crt.WHITE, crt.CYAN,' 内容表示 終了[ESC=-] '));
  SetWindow(5,3,75,11); OpenWindow(1);
  CrtCursor(crt.NoCursor);
  TextColor(crt.GREEN);
  goloxy(1,2); Write(' ﾌｧｲﾙ名       取り込み時間  ｻﾝﾌﾟﾘﾝｸﾞ時間  ﾃﾞｰﾀ数  ｺﾒﾝﾄ');
  goloxy(1,3); Write('-------------------------------------------------------------------------
---');
  TextColor(crt.YELLOW);
  for i := 1 to Num do
   With ADcond[i]^ do begin
     goloxy(1,3+i);
                        FSplit(Dir[i-1].Name, D, N, E);
                        Write(N,' ':9 - Length(N), E, ' ':4 - Length(E));
     goloxy(15,3+i); Write(Duration:3);
     goloxy(29,3+i); Write(Time:3);
     goloxy(43,3+i); Write(SampleNum:5);
     if Length(Comment) < 21 then S := Copy(Comment, 1, Length(Comment))
     else S := Copy(Comment, 1, 20);
     goloxy(51,3+i); Write(S);
   end;
  repeat ch := ReadChar; until ch=ESC;
  CloseWindow(1);
  TextColor(crt.WHITE);
 end;


procedure DisplayWave;
 begin
  SetVisualPage(0);
  WaveLoad(Num);
 end;


procedure Analyze;
 var
  M, N: Integer;
  Peeks: Integer;
 begin
  case Num of
    4: begin M :=  95; N :=  80; end;
    3: begin M := 127; N := 110; end;
    2: begin M := 191; N := 170; end;
    1: begin M := 300; N := 300; end;
    else ERROR('ﾅｼ',FATAL)
  end;
  Peeks := 0; MoveFlag := PEEK;
  MoveLine(1, M, N, Peeks);
  MoveFlag := NONE;
 end;


procedure CalcSpectrum;


 begin
        WaveLoad(Num);
  ShowSpectrum;
 end;

procedure Options;
 var
        No:ShortInt;
 begin
  No:=1; MenuPosition(10,3);
  MenuAttr(crt.red+crt.reverse,crt.green+crt.reverse,crt.white,crt.white+crt.reverse);
  repeat
        No:=SelectMenu(SubSMenu,No);
   case No of
    1: begin SortMode := NAME;  No := 0; end;
    2: begin SortMode := SIZE;  No := 0; end;
    3: begin SortMode := TIME;  No := 0; end;
    4: No := 0;
   end;
  until No=0;
  MenuPosition(3,4);
  MenuAttr(crt.green+crt.reverse,crt.yellow+crt.reverse,crt.white,crt.white+reverse);
 end;


 procedure WaveAnalyze;
  label
        REWRITE;
  var
   No:shortint;
  begin
 REWRITE:
   No:=1; MenuPosition(10,3);
   MenuAttr(crt.red+crt.reverse,crt.green+crt.reverse,crt.white,crt.white+crt.reverse);
   repeat
    No:=SelectMenu(SubAMenu,No);
    case No of
     1:GetFile(Path);
     2:begin
         if Num=0 then goto REWRITE;
         HeadTitle(PEEK);
         DisplayWave;
         TileGraph(Num);
         Analyze;
                                Num := 0;

         ClrScr;
         ClearDevice;
         SetVisualPage(1);
        end;
     3:begin
                                if Num=0 then goto REWRITE;
                                CalcSpectrum;
                                Num := 0;

         ClrScr;
         ClearDevice;
         SetVisualPage(1);
                                goto REWRITE;
        end;
     4:begin
                                if Num=0 then goto REWRITE;
                                HeadTitle(NOTITLE);
                                SpectrumDisplay(Num);
                                Num := 0;

         ClrScr;
         ClearDevice;
```

```
        SetVisualPage(1);
      end;
    5:begin
                        if Num<0 then goto REWRITE;
                        Browser;
                        Num := 0;
                    end;
    6:begin
                        Options;
                        goto REWRITE;
                    end;
    7:No:=0;
   end;
  until No=0;
  MoveFlag := NONE;
  MenuPosition(3,4);
  MenuAttr(crt.green+crt.reverse,crt.yellow+crt.reverse,crt.white,crt.white+reverse);
 end;


procedure DisplayMode;
 begin
      WaveLoad(Num);
  FileBrowse;
 end;


procedure WaveBrowser;
 label
      REWRITE;
 var
  No:shortint;
 begin
REWRITE:
  No:=1; MenuPosition(10,3);
  MenuAttr(crt.red+crt.reverse,crt.green+crt.reverse,crt.white,crt.white+crt.reverse);
  repeat
   No:=SelectMenu(SubMenu,No);
   case No of
    1:GetFile(Path);
    2:begin
                        if Num<0 then goto REWRITE;
                        DisplayMode;
                        Num := 0;
      ClrScr;
      ClearDevice;
      SetVisualPage(1);
     end;
    3:begin
                        if Num<0 then goto REWRITE;
                        Browser;
                        Num := 0;
                    end;
            4:begin
                        Options;
                        goto REWRITE;
                    end;
    5:No:=0;
   end;
  until No=0;
  MoveFlag := NONE;
  MenuPosition(3,4);
  MenuAttr(crt.green+crt.reverse,crt.yellow+crt.reverse,crt.white,crt.white+reverse);
 end; { WaveBrowser }



procedure PrintDriver;
 var
  No:shortint;
 begin
  No: 1; MenuPosition(10,3);
  MenuAttr(crt.red+crt.reverse,crt.green+crt.reverse,crt.white,crt.white+crt.reverse);
  repeat
   No:=SelectMenu(SubPMenu,No);
   case No of
    1: begin CopyMode := PC_PR201; No := 0; end;
    2: begin CopyMode := VP_130K;  No := 0; end;
    3: begin CopyMode := NA_9950;  No := 0; end;
    4: begin CopyMode := PLOTER;   No := 0; end;
    5: No := 0;
   end;
  until No=0;
  MenuPosition(3,4);
  MenuAttr(crt.green+crt.reverse,crt.yellow+crt.reverse,crt.white,crt.white+reverse);
 end;


procedure WaveDirectroy;
 label
      LOOP, ESCAPE;
 var
      i:integer;
  ch: Char;
  s: String[38];
      F: file;
  Attr: Word;
 begin
  SetFrame(crt.WHITE+crt.Reverse,crt.WHITE+crt.Reverse,'ディレクトリー入力');
  SetWindow(10,3,35,6); OpenWindow(4);
  TextColor(crt.WHITE);
  CrtCursor(crt.DispCursor);
  gotoxy(1,1); write(' >> ');
LOOP:
  s := 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijkl';
  i := 0;
  gotoxy(4,1);
  repeat
    ch := ReadChar;
    case ch of
    ESC:goto ESCAPE;
    BS,DEL: if (i-1) >= 0 then begin
        Delete(s, i, 1);
        Dec(i);
        Write(38);
      end;
    else begin
          if (i+1)<39 then begin
            Inc(i);
            s[i] := ch;
            Write(ch);
          end;
      end;
    end;{ case }
  until ch=RET;
  s[0] := chr(i-1);
  if Length(s) > 1 then begin
              Direc := Copy(s, 1, Length(s));
              if Direc[Length(Direc)] <> '¥' then begin
                      Assign(F, Direc);
                      GetFAttr(F, Attr);
                      if (DosError = 0) and (Attr and Directory <> 0) then
                      Direc := Direc + '¥';
              end
      end
  else begin
          ClrScr; goto LOOP;
      end;
ESCAPE:
  CloseWindow(4);
      CrtCursor(crt.NoCursor);
  TextColor(crt.WHITE);
  MenuPosition(3,4);
  MenuAttr(crt.green+crt.reverse,crt.yellow+crt.reverse,crt.white,crt.white+reverse);
end;


begin { of main }
  OvrInit(OvrName);
  OvrInitEMS;
      if OvrResult<>ovrOK then Writeln('Overlaymanager error= ',OvrResult);
  systemint;
  funckey(off); copykey(off);
  TextMode(VL8025);
  DualChannel:=false;
  TriggerLevel := 127;

  No:=1; MenuPosition(3,4);
  MenuAttr(crt.green+reverse,crt.yellow+reverse,crt.white,crt.white+reverse);
  repeat
   No:=SelectMenu(MainMenu,No);
   case No of
    1: SetCondition(cond);
    2: StartMeasure(cond);
    3: WaveBrowser;
    4: WaveAnalyze;
    5: WaveDirectroy;
    6: PrintDriver;
    7: No:=0;
   end;
  until No=0;
  TextMode(InitialMode);
  FuncKey(On);
  GraphTerminate;
end.
```

GLOBAL

```
{*************************************************************
            蛍発光パターン解析プログラム
            グローバル変数定義ユニット
            オリンパス光学工業㈱ ＢＲＣ研究Ｇ
            作成 佐野 徹  平成元年１２月２日
            Version 0.10:   1989/12/02
            Version 0.50:   Revised on 1989/12/14
            Version 1.00:   Revised on 1990/01/11
            Version 2.00:   1991/9/10
            Version 2.01:   1992/3/10
*************************************************************)
{$R+}
{$N+}
Unit Global;
Interface
uses dos;

const
  MAXDATA = 32766; (* 取り込みデータの最大値 *)

  MaxDirSize = 80;

  FATAL = -1;       (* ｴﾗｰ処理 *)
  CAUTION = 0;
  WARNING = 1;
  ALERT = 2;

  NONE = 0;
  MEASURE = 1;
  PEEK = 2;
  BROWSE = 3;
  TILE = 4;
  OVERLAY = 5;
  SPECTRUM = 6;
  NOTITLE = 7;

type
  DirRec   = record
               Attr: Byte;
               Time: Longint;
               Size: Longint;
               Name: string[12];
             end;
  DirList  = array[0..MaxDirSize - 1] of DirRec;

  PBuffer = ^TBuffer;
  TBuffer = array[1..MAXDATA] of Integer;
  BufferList = array[1..4] of PBuffer;

  PADcond = ^TADcond;
  TADcond=record
    Duration: Word;
    Time: Word;
    SampleNum: Word;
    Comment: String[128];
  end;
  ADcondList = array[1..4] of PADcond;


  Minmax=record
    MIN : Integer;
    MAX : Integer;
  end;
```

```
  Range record
    p1, p2: Integer;
  end;

  HeadType = ShortInt;
  ErrorType = ShortInt;

        TCopyMode = (PC_PR201, VP_130K, NW_9050, PLOTER);
        TSortMode = (NAME, SIZE, TIME);
var
  F: File;
  Head: HeadType;
  MoveFlag, Flag: ShortInt;
  FBT: Boolean;

{* 測定,表示条件 *}
  Dir: DirList;
  Path: PathStr;
        Direc: DirStr; {* デフォールトのディレクトリー *}
  Num, Dnum: Integer;
  TimeRange: Range;
  DualChannel:boolean; {* 2チャンネル入力か否か '92/3/10 *}

{* 作業ﾒﾓﾘ領域 *}
  Buffer: BufferList;
  ADcond: ADcondList;
  CopyMode: TCopyMode;
  SortMode: TSortMode;

  AllocMem: array[1..4] of Word; {* 確保したメモリ容量を記録する *}

implementation
var
        cf: file;
        Attr: word;
begin
        SortMode := NAME; {* デフォールトのソートモード *}
        CopyMode := PC_PR201; {* デフォールトのプリンターモード *}
        GetDir(0,Direc); {* デフォールトのディレクトリー *}
        if Direc[Length(Direc)] <> '¥' then begin
              Assign(cf, Direc);
              GetFAttr(cf, Attr);
              if (DosError = 0) and (Attr and Directory <> 0) then
              Direc := Direc + '¥';
        end;

        Getmem(Buffer[1], MAXDATA*SizeOf(integer));
        Getmem(Buffer[2], MAXDATA*SizeOf(integer));
        Getmem(Buffer[3], MAXDATA*SizeOf(integer));
        Getmem(Buffer[4], MAXDATA*SizeOf(integer));
        Getmem(ADcond[1], SizeOf(TADcond));
        Getmem(ADcond[2], SizeOf(TADcond));
        Getmem(ADcond[3], SizeOf(TADcond));
        Getmem(ADcond[4], SizeOf(TADcond));
end.
```

```
  GraphFreeMemPtr := @MyFreeMem;

  GraphDriver := PC98;
  GraphMode := PC98C16;
  InitGraph(GraphDriver, GraphMode, ''); { activate graphics }
  ErrorCode := GraphResult;             { error? }
  if ErrorCode <> grOk then
   begin
    Writeln(GraphErrorMsg(GraphResult));
    Halt(1);
   end;
 end; { graphinitialize }


procedure GraphTerminate;
 begin
    closegraph;
    clrscr;
    textcolor(crt.white);
 end; { graphterminate }


procedure Usage;
 begin
  writeln('使用方法');
  writeln('DOT /H .. このﾒｯｾｰｼﾞを表示して終了');
 end;


procedure systeminit;
 var
  Argc, i:integer;
  Argv:string;
 begin
  argc:=ParamCount;
  while argc <> 0 do
   begin
    argv:=Paramstr(argc);
    if (argv[1]='/') or (argv[1]='-') then begin
      case argv[2] of
       'H','h','?': begin usage; Halt(0);end;
      end;{ of case }
    end;
    dec(argc);
   end;{ of while }
  ADPORT:=$01D0;
        ADInit;
  GraphInitialize;
  SetActivePage(0);
  SetVisualPage(1);
  ClearDevice;

{*  for i:=0 to 4 do GetMem(Dir[i], SizeOf(DirRec)); *}
  Num := 0; Dnum := 0; MoveFlag := NONE;
  CopyMode := PC_PR201;
 end; { systeminit }

end.
```

SYSINT

```
{*****************************************************************
            蛍発光パターン解析プログラム
            下請けオーバレーユニット
            オリンパス光学工業㈱　ＢＲＣ研究Ｇ
            作成　牧野　徹　平成元年１２月２日
            Version 0.10:   1988/12/02
            Version 0.50:   Revised on 1989/12/14
            Version 1.00:   Revised on 1990/01/11
            Version 2.00:   1991/9/10
*****************************************************************}
{$F+}
{$S+}
{$N+}
{$O+}
Unit Sysint;
Interface
uses overlay, global, dirsort,adl2;

procedure GraphInitialize;
procedure GraphTerminate;
procedure Usage;
procedure systeminit;

var
  PreGraphExitProc: Pointer;

Implementation
uses crt, dos, graph, bgidriv, smallfont;

procedure MyGetMem( var p: Pointer; size: Word);
 begin
  System.GetMem(p, size*16);
 end;


procedure MyFreeMem(var p: Pointer; size: Word);
 begin
  if p<>nil then begin
    System.FreeMem(p, size*16);
    p := nil;
  end;
 end;


procedure MyExitProc;
 begin
  ExitProc := PreGraphExitProc;
  CloseGraph;
 end;


procedure GraphInitialize;
 var
  GraphDriver : integer;  { The Graphics device driver }
  GraphMode   : integer;  { The Graphics mode value }
  ErrorCode   : integer;  { Reports any graphics errors }
 begin
  if RegisterBGIdriver(@PC98DriverProc) < 0 then Halt(1);
  if RegisterBGIfont(@SmallFontProc) < 0 then Halt(1);

  PreGraphExitProc := ExitProc;
  ExitProc := @MyExitProc;
  GraphGetMemPtr := @MyGetMem;
```

SUPPORT

```
{*****************************************************************
            蛍発光パターン解析プログラム
            下請けオーバレーユニット
            オリンパス光学工業㈱　ＢＲＣ研究Ｇ
            作成　牧野　徹　平成元年１２月２日
            Version 0.10:   1988/12/02
            Version 0.50:   Revised on 1989/12/14
            Version 1.00:   Revised on 1990/01/11
            Version 2.00:   1991/9/10
                                        Version 2.00a:  1992/3/10
                                        Version 2.00b:  1993/3/15
*****************************************************************}
{$F+}
{$S+}
{$N+}
{$O+}
unit Support;
Interface
uses overlay, dos, dirsort, global;

procedure Buzzer(num:integer);
procedure Error(mes: String; ErrorNum:ErrorType);
procedure HeadTitle(H: HeadType);
procedure GetFile(var Path: PathStr);
function log10(x: Single): Single;
function packedout(x:Single): String;

Implementation
uses crt, graph, console, sysinit;

function log10(x: Single): Single;
 begin
  log10 := ln(x) / ln(10);
 end;


procedure Buzzer(num:integer);
 begin
  sound(250);
  delay(num);
  Nosound;
 end;{ buzzer }


procedure Error(mes: String; ErrorNum:ErrorType);
 var
    ch:char;
          i:byte;
 begin
  TextBank(0);
  SetFrame(crt.green+crt.reverse,crt.white+crt.reverse,' ERROR ');
  setwindow(17,4,52,9); openwindow(5);
        textcolor(crt.yellow);
        gotoxy(2,2); write(mes);
        textcolor(crt.red);
        case ErrorNum of
         FATAL :
                                        begin
                                              gotoxy(2,4);
                                              write('Hit ESC Key !!');
                                        end;
        CAUTION, WARNING , ALERT:
                                        begin
```

```
                                        gotoxy(2,4);
                                        write('Hit ESC Key !!');
                                    end;
        end;
        repeat ch:=readchar; until ch=ESC;
        textcolor(crt.white);
        if ErrorNum FATAL then begin
                SetVisualPage(0);
                SetActivePage(0);
                GraphTerminate;
                TextMode(InitialMode);
                Halt(1);
        end;
  closewindow(5);
end;{ Error }


procedure HeadTitle(H: HeadType);
 begin
  SetLinestyle(SOLIDLN,0,NORMWIDTH);
  Setcolor(graph.CYAN);
  SetFillstyle(CloseDotFill,graph.CYAN);
  Settextstyle(DEFAULTFONT,HORIZDIR,1);
  SetTextjustify(CENTERTEXT,TOPTEXT);
  case H of
    MEASURE:begin
                                                ClrScr;
            Rectangle(0,0,639,17);
            FloodFill(2,2,graph.CYAN);
            Setcolor(graph.BROWN);
            OutTextxy(319,2,'***** 発光ﾊﾟﾀｰﾝ測定ﾌﾟﾛｸﾞﾗﾑ *****');
            OutTextxy(320,2,'***** 発光ﾊﾟﾀｰﾝ測定ﾌﾟﾛｸﾞﾗﾑ *****');
          end;
    PEEK:begin
                                                ClrScr;
            Rectangle(257,0,639,17); }
            Rectangle(0,0,639,17);
            FloodFill(2,2,graph.CYAN);
            Setcolor(graph.BROWN);
            OutTextxy(250,2,'***** 発光ﾊﾟﾀｰﾝ測定ﾌﾟﾛｸﾞﾗﾑ *****');
            OutTextxy(251,2,'***** 発光ﾊﾟﾀｰﾝ測定ﾌﾟﾛｸﾞﾗﾑ *****');
            TextColor(crt.CYAN):TextReverse(Reverse);
            gotoxy(1,1); write('切り替え f1');
            gotoxy(70,1); write('ﾌｧｲﾙ f10');
         end;
    BROWSE:begin
                                                ClrScr;
            Rectangle(0,0,639,17);
            FloodFill(2,2,graph.CYAN);
            Setcolor(graph.BROWN);
            OutTextxy(250,2,'***** 発光ﾊﾟﾀｰﾝ測定ﾌﾟﾛｸﾞﾗﾑ *****');
            OutTextxy(251,2,'***** 発光ﾊﾟﾀｰﾝ測定ﾌﾟﾛｸﾞﾗﾑ *****');
            TextColor(crt.CYAN):TextReverse(Reverse);
            gotoxy(1,1); write('ﾏｰｸ  f1');
            gotoxy(70,1); write('ﾌｧｲﾙ f10');
          end;
            SPECTRUM:begin
                                                ClrScr;
            Rectangle(0,0,639,17);
            FloodFill(2,2,graph.CYAN);
            Setcolor(graph.BROWN);
            OutTextxy(350,2,'***** 発光ﾊﾟﾀｰﾝ測定ﾌﾟﾛｸﾞﾗﾑ *****');
            OutTextxy(351,2,'***** 発光ﾊﾟﾀｰﾝ測定ﾌﾟﾛｸﾞﾗﾑ *****');
            TextColor(crt.CYAN):TextReverse(Reverse);
            gotoxy(1,1); write('ﾏｰｸ  f1');


            gotoxy(9,1); write('ﾒﾆｭｰへ戻る f2');
            gotoxy(70,1); write('ﾌｧｲﾙ f10');
          end;
            NONE,TILE,OVERLAY:
                  begin
                                                ClrScr:
            TextColor(crt.CYAN):TextReverse(Reverse);
            gotoxy(70,1); write('ﾌｧｲﾙ f10');
          end;
            NOTITLE:
                  begin
                                                ClrScr:
            TextColor(crt.CYAN):TextReverse(Reverse);
            gotoxy(1,1); write('ﾒﾆｭｰへ戻る f1');
            gotoxy(70,1); write('ﾌｧｲﾙ f10');
          end;
  end;
  TextColor(crt.WHITE);
  TextReverse(NoReverse);
  Setcolor(graph.WHITE);
end;{ HeadTitle }


procedure GetFile(var Path: PathStr);
 var
  i: integer;
  ch: Char;
  s: String[20];
 begin
  SetFrame(crt.WHITE+crt.Reverse, crt.CYAN,'ﾌｧｲﾙﾈｰﾑ入力');
  SetWindow(5,3,30,7); OpenWindow(1);
  TextColor(crt.CYAN);
  CrtCursor(crt.DispCursor);
  Path := '*.DAT';
  s := 'ABCDEFGHIJKLMNOPQR';
  i := 0;
  gotoxy(2,2); write(Path);
  gotoxy(2,2);
  repeat
    ch := ReadChar;
    case ch of
    ESC:
      begin
        CrtCursor(crt.NoCursor);
        CloseWindow(1);
        TextColor(crt.WHITE);
        Exit;
      end;
    BS: if (i-1) >= 0 then begin
          Delete(s, i, 1);
          Dec(i);
          Write(#8);
      end;
    else begin
          if (i+1)<21 then begin
            Inc(i);
            s[i] := ch;
            Write(ch);
          end;
      end;
    end;
  until ch=RET;
  s[0] := Chr(i-1);
  if Length(s) <> 0 then Path := Copy(s, 1, Length(s));
  CloseWindow(1);
```

```
  CrtCursor(crt.NoCursor);
  TextColor(crt.WHITE);
  PrintFiles(Path);
  SelectFiles(Dir, Num);
 end;

function packedowl(x: Single): String;
 var
    s:string[5];
 begin
       Str(x:5:2, s);
  packedowl := s;
 end;

end.
```

DIRSORT

```
{*****************************************************************}
                蛍発光パターン解析プログラム
                ﾌｧｲﾙ読み込みユニット
                オリンパス光学工業㈱　ＢＲＣ研究Ｇ
                作成　牧野　徹　平成元年１２月２日
                Version 0.10:   1989/12/02
                Version 0.50:   Revised on 1989/12/14
                Version 1.00:   Revised on 1990/01/11
                Version 2.00:   1991/9/10
*****************************************************************}
{$R+}
{$N+}
Unit DirSort;
Interface
uses Dos, global;
procedure PrintFiles(var Path: PathStr);
procedure SelectFiles(var Files: DirList; var Num: Integer);

Implementation
uses Crt, Console;

const
  MonthStr: array[1..12] of string[3] = (
    'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
    'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec');

type
  LessFunc = function(X, Y: DirRec): Boolean;

var
  Count: Integer;
  Less: LessFunc;
  Dir: DirList;

function NumStr(N, D: Integer): String;
begin
  NumStr[0] := Chr(D);
  while D > 0 do
  begin
    NumStr[D] := Chr(N mod 10 + Ord('0'));
    N := N div 10;
    Dec(D);
  end;
end;

{$F+}

function LessName(X, Y: DirRec): Boolean;
begin
  LessName := X.Name < Y.Name;
end;

function LessSize(X, Y: DirRec): Boolean;
begin
  LessSize := X.Size < Y.Size;
end;

function LessTime(X, Y: DirRec): Boolean;
begin
  LessTime := X.Time > Y.Time;
end;

{$F-}
```

```
procedure QuickSort(L, R: Integer);
var
  I, J: Integer;
      X, Y: DirRec;
begin
  I := L;
  J := R;
  X := Dir[(L + R) div 2];
  repeat
    while Less(Dir[I], X) do Inc(I);
    while Less(X, Dir[J]) do Dec(J);
    if I <= J then
    begin
      Y := Dir[I];
      Dir[I] := Dir[J];
      Dir[J] := Y;
      Inc(I);
      Dec(J);
    end;
  until I > J;
  if L < J then QuickSort(L, J);
  if I < R then QuickSort(I, R);
end;

procedure GetCommand(var Path:PathStr);
var
  Attr: Word;
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  F: File;
begin
        case SortMode of
                NAME: Less := LessName;
                SIZE: Less := LessSize;
                TIME: Less := LessTime;
                else
                Less := LessName;
        end;
  FSplit(Path, D, N, E);
  if N = '' then N := '*';
  if E = '' then E := '.*';
        if D = '' then D := Direc;
  Path := D + N + E;
{       Path := FExpand(Path);
        if Path[Length(Path)] <> '\' then
        begin
                Assign(F, Path);
                GetFAttr(F, Attr);
                if (DosError = 0) and (Attr and Directory <> 0) then
                        Path := Path + '\';
        end;
}
end;


procedure FindFiles(Path: PathStr);
var
  F: SearchRec;
begin
  Count := 0;
  FindFirst(Path,  Directory + Archive, F);
  while (DosError = 0) and (Count < MaxDirSize) do begin
    Move(F.Attr, Dir[Count], Length(F.Name) + 10);


    Inc(Count);
    FindNext(F);
  end;
end;

procedure SortFiles;
begin
  if (Count <> 0) and (@Less <> nil) then
    QuickSort(0, Count - 1);
end;


procedure GetFiles(var N: NameStr;var E: ExtStr; I: Integer);
 var
  Q: Integer;
 begin
      with Dir[I] do begin
    Q := Pos('.', Name);
    if Q > 1 then begin
      N := Copy(Name, 1, Q - 1);
      E := Copy(Name, Q + 1, 3);
    end else begin
    N := Name;
    E := '';
    end;
  end;
 end;


procedure PrintFiles(var Path: PathStr);
var
  I, J: Integer;
  T: DateTime;
  N: NameStr;
  E: ExtStr;
begin
  GetCommand(Path);
  FindFiles(path);
  SortFiles;
  if Count = 0 then Exit;
  SetFrame(crt.Yellow,crt.Yellow+reverse,'ﾌｧｲﾙ選択 '+Path+' 終了[ESC] 選択[RET]');
  SetWindow(16,3,76,23); OpenWindow(2);
  TextColor(crt.WHITE);
  for I := 0 to Count-1 do begin
      GetFiles(N, E, I);
      Write(N, ' ': 9 - Length(N), E, ' ': 4 - Length(E));
    if I and 3 <> 3 then Write(' ':2)
    else WriteLn;
  end;
end;


procedure SelectFiles(var Files: DirList; var Num: Integer);
 var
  I, J: Integer;
  N: NameStr;
  E: ExtStr;
  S: PathStr;
  ch: Char;

procedure TurnOnOff(i: Integer; OnOff:Word);
 procedure NormalAttr;
  begin
    TextColor(crt.WHITE);
    TextBEverse(NoReverse);
  end;

 procedure ReverseAttr;
  begin
    TextColor(crt.YELLOW);
   TextBEverse(Reverse);
  end;

 var
  p, q:Integer;
 begin
  p := ( i mod 4)*15 ; q := (i div 4);
  gotoxy(1+p, q+1);
  if OnOff=Reverse then begin
    ReverseAttr;
    GetFiles(N, E, i);
      Write(N, ' ': 9 - Length(N), E, ' ': 4 - Length(E));
  end else begin
    NormalAttr;
    GetFiles(N, E, i);
      Write(N, ' ': 9 - Length(N), E, ' ': 4 - Length(E));
  end;
 end;


function cup(var i: Integer): Integer;
 var
  a, b: Integer;
 begin
  a := i div 4; b := i-4;
  if (a = 0) or (b < 0) then begin
    cup := i;
    Exit;
  end;
  TurnOnOff(i, crt.NoReverse);
  Dec(i,4);
  TurnOnOff(i, crt.Reverse);
  cup := i;
 end;


function cdown(var i: Integer): Integer;
 var
  a, b, c: Integer;
 begin
  a := i div 4; b := COUNT div 4; c := i + 4;
  if ( a = b) or (c > COUNT - 1) then begin
    cdown := i;
    Exit;
  end;
  TurnOnOff(i, crt.NoReverse);
  Inc(i, 4);
  TurnOnOff(i, crt.Reverse);
  cdown := i;
 end;


function cright(var i: Integer): Integer;
 var
  a: Integer;
 begin
  a := COUNT - 1;
  if i = a then begin
    cright := i;


    Exit;
  end;
  TurnOnOff(i, crt.NoReverse);
  Inc(i);
  TurnOnOff(i, crt.Reverse);
  cright := i;
 end;


function cleft(var i: Integer): Integer;
 begin
  if i = 0 then begin
    cleft := i;
    Exit;
  end;
  TurnOnOff(i, crt.NoReverse);
  Dec(i);
  TurnOnOff(i, crt.Reverse);
  cleft := i;
 end;

 begin
  CrtCursor(crt.NoCursor);
  I := 0; Num := 0;
  SetFrame(crt.GREEN+crt.Reverse, crt.CYAN+crt.REVERSE,'選択ﾌｧｲﾙ');
  SetWindow(2, 3, 15, 9); OpenWindow(3);
  ChangeWindow(2);
    TurnOnOff(I, crt.Reverse);
  repeat
    ch := ReadChar;
    case ch of
      Up:    I := cup(I);
      Down:  I := cdown(I);
      Left:  I := cleft(I);
      Right: I := cright(I);
      RET:   begin
               if Num <= 3 then begin
                                                    Files[Num].Attr := Dir[I].Attr;
                                                    Files[Num].Time := Dir[I].Time;
                                                    Files[Num].Size := Dir[I].Size;
                                                    Files[Num].Name := Dir[I].Name;
                                                    ChangeWindow(3);
                                                    TextColor(crt.YELLOW);
                                                    gotoxy(I, Num+1);
                                                    Write(Files[Num].Name);
                                                    Inc(Num);
                                                    ChangeWindow(2);
                                                    TextColor(crt.WHITE);
                                                end;
                                            end;
             end;
    until (ch=ESC);
            CloseWindow(2);
    CloseWindow(3);
   CrtCursor(crt.DispCursor);
 end;

end.
```

BGIDRV

```
{ Copyright (c) 1985, 1990 by Borland International, Inc. }

unit BGIDriv;
{
  BGILINK.PAS によって使用されるユニットです。このユニットは、すべて
  の BGI グラフィックスドライバがリンクされた唯一のユニットになります。
  これによって、ドライバファイルを .EXE ファイルに容易にリンクすること
  ができます。
  詳しくは、BGILINK.PAS を参照してください。
}
interface

procedure PC98DriverProc;

implementation

{$L PC98.OBJ }
procedure PC98DriverProc; external;
end.
```

SMALFONT

```
Unit SmalFont;

interface
procedure SmallFontProc;

implementation
{$L SMALL.OBJ }
procedure SmallFontProc; external;
end.
```

AD12

```
(****************************************************************************
*******            AD変換ボード制御プログラム                          ****
*******         コンテックAD12−16Tボード対応                          ****
*******                                                                ****
*******       オリンパス光学工業㈱  BRC研究G                           ****
*******       作成 牧野 徹  平成元年12月2日                            ****
*******       Version 0.10                                             ****
*******       Version 0.20  1992/3/10                                  ****
****************************************************************************)
unit ad12;
interface
var
  ADERROR: ShortInt;
  ADSTAT: ShortInt;
  ADPORT:WORD;

procedure ADInit;
procedure ADdataIn(var Buffer; dnum:word; chan:byte);
procedure ADdataIn2(var Buffer1, Buffer2; dnum:word; chan1, chan2:byte);
procedure ADparamSet(var cond);
procedure GetAD(NCOUNT,DSG,DFS,PORT:WORD;CHAN:BYTE);
procedure GetAD2(NCOUNT:WORD;VAR DATA1,DATA2;PORT:WORD;CHAN1,CHAN2:BYTE);
procedure GetAD3(VAR DATA1,DATA2;PORT:WORD;CHAN1,CHAN2:BYTE);
procedure IntervalSet(var c0,c1,c2:word);
function  OpTrigger: word; (* 光トリガーを追加 '92/3/10 *)
procedure wait;
procedure triggerreset;
procedure timerstart;
procedure timerstop;
implementation

type
  TADcond=record
    Duration: Word;
    Time: Byte;
    SampleNum: Word;
    Comment: String[128];
  end;


{$L ad1216}
procedure GetAD(NCOUNT,DSG,DFS,PORT:WORD;CHAN:BYTE);external;
procedure GetAD2(NCOUNT:WORD;VAR DATA1,DATA2;PORT:WORD;CHAN1,CHAN2:BYTE);external;
procedure GetAD3(VAR DATA1,DATA2;PORT:WORD;CHAN1,CHAN2:BYTE);external;


procedure delay(count:integer);
 var
  i:integer;
 begin
   for i:=1 to count do;
 end;


procedure ADInit;
 var
  cl,ch:byte;
 begin
   cl:=port[ADPORT]; ch:=port[ADPORT+1];
   if (cl=$FF) and (ch=$FF) then begin
     writeln('AD12−16がセットされていません');
     ADSTAT := 1;
   end else begin


     writeln('AD12−16がセットされています');
     ADSTAT := 0;
   end;
 end;

procedure ADparamSet(var cond);
 var
  cth0,ctl0,cth1,ctl1,cth2,ctl2:byte;
  p:word;
  al:integer;
 begin
  with TADcond(cond) do begin
    cth0 := 0; ctl0 := 4;
    cth1 := 0; ctl1 := 10;
    p := 100 * Duration;
    cth2 := Hi(p); ctl2 := Lo(p);
   end;
  p:=ADPORT+2;
  port[p]:=$00; (*  内部タイマー停止 *) delay(20);
  p:=ADPORT+1;
  port[p]:=$01; (* トリガー入力ステータスリセット *)
  p:=ADPORT+$e;
  port[p]:=$34; (* 内部カウンター0選択 *) delay(20);
  p:=ADPORT+8;
  port[p]:=ctl0;(* カウンター0下位データ *) delay(20);
  port[p]:=cth0;(* カウンター0上位データ *) delaY(20);
  p:=ADPORT+$e;
  port[p]:=$74; (* 内部カウンター1選択 *) delay(20);
  p:=ADPORT+$a;
  port[p]:=ctl1;(* カウンター1下位データ *)   delay(20);
  port[p]:=cth1;(* カウンター1上位データ *)   delaY(20);
  p:=ADPORT+$e;
  port[p]:=$B4; (* 内部カウンター2選択 *) delay(20);
  p:=ADPORT+$c;
  port[p]:=ctl2;(* カウンター2下位データ *)  delay(20);
  port[p]:=cth2;(* カウンター2上位データ *)  delay(20);
 end; { ADParamSet }


procedure ADdataIn(var Buffer; dnum:word; chan:byte);
 begin
  ADERROR:=0;
       getad(dnum,seg(Buffer),ofs(Buffer),ADPORT,chan);
 end;{ ADdataIn }

procedure ADdataIn2(var Buffer1, Buffer2; dnum:word; chan1, chan2:byte);
 begin
  ADERROR:=0;
       getad2(dnum, Buffer1, Buffer2, ADPORT,chan1, chan2);
 end;


procedure IntervalSet(var c0,c1,c2:word);
 var
  c0l,c0h,c1l,c1h,c2l,c2h:byte;
 begin
  c0l:=lo(c0); c0h:=hi(c0);
  c1l:=lo(c1); c1h:=hi(c1);
  c2l:=lo(c2); c2h:=hi(c2);

  port[ADPORT+2]:=0;
  port[ADPORT+1]:=1;
  port[ADPORT+$e]:=$34;
  port[ADPORT+8]:=c0l;
  port[ADPORT+8]:=c0h;
```

```
  port[ADPORT+$c]:=$74;
  port[ADPORT+$a]:=c11;
  port[ADPORT+$a]:=c1h;
  port[ADPORT+$c]:=$b4;
  port[ADPORT+$c]:=c2l;
  port[ADPORT+$c]:=c2h;
  port[ADPORT+2]:=$10;  (* タイマースタート *)
 end;{ intervalset }


 function UpTrigger: word;
  var
   temp: array[1..5] of word;
   sum: word;
   i: integer;
  begin
   sum := 0;
   for i:= 1 to 5 do
    begin
                ADdatain(temp[i],1,1);
     sum := sum + temp[i];
         end;
         Uptrigger := sum div 5;
  end;


 procedure wait;
  var
   ah:byte;
  begin
   ah:=0;
   while(ah and $10)=0 do ah:=port[ADPORT+1];
  end;{ wait }

 procedure triggerreset;
  begin
        port[ADPORT+1] := 1;
  end;


 procedure timerstart;
  begin
        port[ADPORT+2] := $10;
  end;


 procedure timerstop;
  begin
   port[ADPORT+2]:=0;
  end;{ timerstop }

 end. { implementation }
```

```
IOWAIT  2
        AND   AX,0FFFH
  MOV   ES:[SI],AX
  INC   SI
  INC   SI
;
  INC   DX
  MOV   AL,00000001B
  OUT   DX,AL ;トリガーリセット
  IOWAIT  1
  DEC   CX
  CMP   CX,0H
  JNE   @@1
;
  INC   DX
  MOV   AL,00000000B
  OUT   DX,AL   ;タイマーリセット
  IOWAIT  1
  POP   SI
  POP   ES
  RET
GETAD ENDP
;
;
GETAD2 PROC FAR NCOUNT:WORD, DATA1:DWORD, DATA2:DWORD, PORT:WORD, CHAN1:BYTE, CHAN2:BYTE
  PUBLIC GETAD2
  PUSH  ES
  PUSH  SI
        PUSH  DS
        PUSH  DI
  MOV   CX,NCOUNT
  LDS   SI,DATA1          ;第一変数のアドレスをDS,SIにロード
  LES   DI,DATA2          ;第二変数のアドレスをES,DIにロード
  MOV   DX,PORT
  INC   DX
  INC   DX
  MOV   AL,00010000B
  OUT   DX,AL ;タイマースタート
  IOWAIT  1
  DEC   DX
;
@@3:
  IN  AL,DX ;タイマーカウント待
  IOWAIT  2
  AND   AL,00010000B
  JZ    @@3
;
  DEC   DX
  MOV   AL,CHAN1 ;使用チャンネルの選択
        DEC   AL
  OR    AL,00010000B ;変換スタートコマンドの設定
  OUT   DX,AL   ;変換スタート
  IOWAIT  1
  INC   DX
;
@@4:
  IN AL,DX  ;変換待
  IOWAIT  2
  AND   AL,10000000B
  JNZ   @@4
;
  DEC   DX
  IN    AX,DX  ;データ入力第一データ
  IOWAIT  2
        AND   AX,0FFFH
```

AD1216

```
;*********************************************************
;**  OLYMPUS OPTICAL CO., Ltd.                          ***
;**        R.D.C., RESEARCH GROUP                       ***
;**        T. MAKINO                                    ***
;**        1989/12/2                                    ***
;**    多チャンネルの入力をサポート                     ***
;**    REVISED ON 1989/12/7                             ***
;**        1992/3/10 Revised Ver 1.01                   ***
;**        1992/4/28 Revised Ver 1.02                   ***
;*********************************************************
.MODEL TPASCAL
.CODE
LOCALS @@
; PROCEDURE GETAD(NCOUNT,DSEG,DOFS,PORT:WORD,CHAN:BYTE);
; PROCEDURE GETAD2(NCOUNT,DSEG1,DOFS1,DSEG2,DOFS2,PORT:WORD,CHAN1,CHAN2:BYTE);
;
IOWAIT  MACRO COUNT
  REPT  COUNT
  JMP SHORT $+2
  ENDM
ENDM
;
;
;
GETAD PROC FAR NCOUNT:WORD,DSG:WORD,DOF:WORD,PORT:WORD,CHAN:BYTE
  PUBLIC GETAD
  PUSH  ES
  PUSH  SI
  MOV   CX,NCOUNT
  MOV   AX,DSG
  MOV   ES,AX
  MOV   AX,DOF
  MOV   SI,AX
  MOV   DX,PORT
  INC   DX
  INC   DX
  MOV   AL,00010000B
  OUT   DX,AL ;タイマースタート
  IOWAIT  1
  DEC   DX
;
@@1:
  IN  AL,DX ;タイマーカウント待
  IOWAIT  2
  AND   AL,00010000B
  JZ    @@1
;
  DEC   DX
  MOV   AL,CHAN ;使用チャンネルの選択
        DEC   AL
  OR    AL,00010000B ;変換スタートコマンドの設定
  OUT   DX,AL   ;変換スタート
  IOWAIT  1
  INC   DX
;
@@2:
  IN AL,DX  ;変換待
  IOWAIT  2
  AND   AL,10000000B
  JNZ   @@2
;
  DEC   DX
  IN    AX,DX  ;データ入力
```

```
  MOV   DS:[SI],AX
  INC   SI
  INC   SI
;
  MOV   AL,CHAN2 ;使用チャンネルの選択
        DEC   AL
  OR    AL,00010000B ;変換スタートコマンドの設定
  OUT   DX,AL   ;変換スタート
  IOWAIT  1
  INC   DX
;
@@5:
  IN AL,DX  ;変換待
  IOWAIT  2
  AND   AL,10000000B
  JNZ   @@5
;
  DEC   DX
  IN    AX,DX  ;データ入力第２データ
  IOWAIT  2
        AND   AX,0FFFH
  MOV   ES:[DI],AX
  INC   DI
  INC   DI
;
  INC   DX
  MOV   AL,00000001B
  OUT   DX,AL ;トリガーリセット
  IOWAIT  1
  DEC   CX
  CMP   CX,0H
  JNE   @@3
;
  INC   DX
  MOV   AL,00000000B
  OUT   DX,AL   ;タイマーリセット
  IOWAIT  1
        POP   DI
        POP   DS
  POP   SI
  POP   ES
  RET
GETAD2 ENDP
;
GETAD3 PROC FAR DATA1:DWORD,DATA2:DWORD,PORT:WORD,CHAN1:BYTE,CHAN2:BYTE
  PUBLIC GETAD3
  PUSH  ES
  PUSH  SI
        PUSH  DS
        PUSH  DI
  LDS   SI,DATA1          ;第一変数のアドレスをDS,SIにロード
  LES   DI,DATA2          ;第二変数のアドレスをES,DIにロード
  MOV   DX,PORT
  INC   DX
;
@@6:
  IN  AL,DX ;タイマーカウント待
  IOWAIT  2
  AND   AL,00010000B
  JZ    @@6
;
  DEC   DX
  MOV   AL,CHAN1 ;使用チャンネルの選択
        DEC   AL
  OR    AL,00010000B ;変換スタートコマンドの設定
```

```
  OUT   DX,AL     ;変換スタート
  IOWAIT  1
  INC   DX
;
@@7:
  IN AL,DX  ;変換待
  IOWAIT  2
  AND   AL,10000000B
  JNZ   @@7
;
  DEC   DX
  IN    AX,DX  ;データ入力第一データ
  IOWAIT  2
        AND   AX,0FFFH
  MOV   DS:[SI],AX
;
  MOV   AL,CHAN2 ;使用チャンネルの選択
        DEC   AL
  OR    AL,00010000B ;変換スタートコマンドの設定
  OUT   DX,AL     ;変換スタート
  IOWAIT  1
  INC   DX
;
@@8:
  IN AL,DX  ;変換待
  IOWAIT  2
  AND   AL,10000000B
  JNZ   @@8
;
  DEC   DX
  IN    AX,DX  ;データ入力第2データ
  IOWAIT  2
        AND   AX,0FFFH
  MOV   ES:[DI],AX
        POP   DI
        POP   DS
  POP   SI
  POP   ES
  RET
GETAD3 ENDP
CODE  ENDS
  END
```

CALCMEM

```
(**************************************************************************
*******    最大エントロピー法                                         ****
*******                                                               ****
*******    オリンパス光学工業㈱  ＢＲＣ研究Ｇ                         ****
*******    作成  牧野  徹  平成２年２月１１日                         ****
*******    Version 0.10                                               ****
*******    Version 0.20                                               ****
*******    Version 0.30 '93/3/15                                      ****
**************************************************************************)
(*  MMAX  予測誤差フィルターの項数
    LMAX  自己相関関数を求める最大ラグ数
    NMAX  データ数
    DT    データ間隔
    X(i) 入力データ、平均値を除去したデータ
    B(i) one-sideスペクトル
    G(i) 予測誤差フィルター
    C(i) 自己相関係数
    FPE  予測誤差の分散
    AIC  赤池情報基準
*)
{$F+}
{$E+}
{$N+}
{$D+}
unit CalcMem;
interface
uses global;

procedure SpectrumDisplay(Num:integer);
procedure Mem(var nn: Word; var Data: PBuffer);

implementation
uses crt, dos, graph, console, support, hcopy;

const
  NNMAX=1025;
  MMMAX=256;
        factor:integer = 176;

  SubSMenu:MenuList = (( Name: ''; Command: ESC),
      (Name: '保存する'; Command:'F'),
      (Name: '保存しない'; Command:'D'),
      (Name: ''; Command:#0),
      (Name: ''; Command:#0),
      (Name: ''; Command:#0),
      (Name: ''; Command:#0),
      (Name: ''; Command:#0),
      (Name: ''; Command:#0),
      (Name: ''; Command:#0));

type
  PNmaxType = ^NmaxType;
  NmaxType = array[1..NNMAX] of Single;
  PN2maxType = ^N2maxType;
  N2maxType = array[1..(NNMAX Div 2)] of Single;

  PMmaxType = ^MmaxType;
  MmaxType = array[1..MMMAX] of Single;

var
  G, GG: PMmaxType;
  b1, b2: PMmaxType;
  F, E: PN2maxType;
```

```
  dt: Single;
  pm, tempC1: Single;
  NMAX, MMAX, NYQ: Integer;
  SADcond:TADcond;


procedure PlotSpectrum(Orig:Integer);forward;

procedure SpectrumSave;
 label
  LOOP;
 var
  fn: file;
  D: DirStr;
  N: NameStr;
  Ex: ExtStr;
  I: Integer;
  Name: PathStr;
  ch: Char;
  s: String[20];
  Wn: Word;
        Tstr:string[80];
 begin
  SetFrame(crt.WHITE+crt.Reverse,crt.WHITE+crt.Reverse,'ﾌｧｲﾙ名入力');
  SetWindow(2,17,27,21); OpenWindow(4);
  SetFrame(crt.GREEN+crt.Reverse,crt.GREEN+crt.Reverse,'コメント入力');
  SetWindow(2,18,75,21); OpenWindow(5);
  TextColor(crt.WHITE);
  CrtCursor(crt.DispCursor);
  gotoxy(1,1); write('>> ');
  readln(Tstr);
  SADcond.Comment := Tstr;
  CloseWindow(5);
  CrtCursor(crt.DispCursor);
  gotoxy(1,2); write('>> ');
  TextColor(crt.YELLOW);
LOOP:
  s := 'ABCDEFGHIJKLMNOPQRST';
  I := 0;
  gotoxy(4,2);
  repeat
    ch := ReadChar;
    case ch of
    ESC:
      begin
        CrtCursor(crt.NoCursor);
        CloseWindow(4);
        TextColor(crt.WHITE);
        Exit;
      end;
    BS,DEL: if (i-1) >= 0 then begin
        Delete(s, i, 1);
        Dec(i);
        Write(#8);
      end;
    else begin
          if (i+1)<21 then begin
            Inc(i);
            s[I] := ch;
            Write(ch);
          end;
       end;
    end;
  until ch=RET;


  s[0] := Chr(I-1);
  if Length(s) > 1 then Name := Copy(s, 1, Length(s))
  else begin ClrScr; goto LOOP; end;
  FSplit(Name, D, N, Ex);
  if Ex = '' then Ex := '.SPC';
        if D='' then D := Direc;
  Name := D + N + Ex;
  s := FSearch(Name, D);
  if s <> '' then goto LOOP;
  {$I-}
  Assign(fn, Name);
  Rewrite(fn, 1);
  {$I+}
  if IOResult<>0 then begin
    ERROR('ﾌｧｲﾙ名が不正です', ALERT);
    goto LOOP;
  end;
  BlockWrite(fn, SADcond, SizeOf(TADcond), WN);
  if WN<>SizeOf(TADcond) then ERROR('ｾｰﾌﾞｴﾗｰ', FATAL);
  BlockWrite(fn, E^, SizeOf(Single)*NYQ, WN);
  if WN<>SizeOf(Single)*NYQ then ERROR('ｾｰﾌﾞｴﾗｰ', FATAL);
  BlockWrite(fn, F^, SizeOf(Single)*NYQ, WN);
  if WN<>SizeOf(Single)*NYQ then ERROR('ｾｰﾌﾞｴﾗｰ', FATAL);
  Close(fn);
  CloseWindow(4);
  CrtCursor(crt.NoCursor);
  TextColor(crt.WHITE);
 end;


procedure TileSpectrum(Num: Integer);

procedure SpectrumLoad(Lnum:integer);
 var
  D: DirStr;
  N: NameStr;
  Ex: ExtStr;
  Name: PathStr;
  RN: Word;
        fn : file;
 begin
  FSplit(Path, D, N, Ex);
        Name := D + Dir[Lnum-1].Name;
  Assign(fn, Name);
  Reset(fn, 1);
  BlockRead(fn, SADcond, SizeOf(TADcond), RN);
  if RN<>SizeOf(TADcond) then ERROR('ﾛｰﾄﾞｴﾗｰ', FATAL);
        NYQ := SADcond.SampleNum;
        if MemAvail < NYQ*SizeOf(Single)*2 then
                ERROR('ﾒﾓﾘが不足です',FATAL);
        GetMem(F, NYQ*SizeOf(Single));
 GetMem(E, NYQ*SizeOf(Single));
        BlockRead(fn, E^, NYQ*SizeOf(Single), RN);
        if RN<>NYQ*SizeOf(Single) then ERROR('ﾛｰﾄﾞｴﾗｰ', FATAL);
        BlockRead(fn, F^, NYQ*SizeOf(Single), RN);
        if RN<>NYQ*SizeOf(Single) then ERROR('ﾛｰﾄﾞｴﾗｰ', FATAL);
        Close(fn);
        end;

        procedure Allocatemem;
        begin
        freeMem(E, NYQ*SizeOf(Single));
                freeMem(F, NYQ*SizeOf(Single));
        end;
```

```
procedure figurecaption(loc:integer);
begin
                OutTextxy(109, 18 + loc+3, '0.1');
   OutTextxy(110, 18 + loc+3, '0.1');
                OutTextxy(109+factor, 18 + loc+3, '1.0');
   OutTextxy(110+factor, 18 + loc+3, '1.0');
                OutTextxy(109+factor*2, 18 + loc+3, '10.0');
                OutTextxy(110+factor*2, 18 + loc+3, '10.0');
   OutTextxy(630, 18 + loc+3, '100.0');
                OutTextxy(631, 18 + loc+3, '100.0');
{
                OutTextxy(374, 18+ loc+3 + 20,'f / Hz');
                OutTextxy(375, 18+ loc+3 + 20,'f / Hz');
}
 end;{ figurecaption }

 var
  i, j, k:Integer;
      lx: single;
begin
 SetColor(graph.BROWN);
 SetLineStyle(SOLIDLN,0,NORMWIDTH);
 SetTextStyle(SMALLFONT, HORIZDIR, 4);
 SetTextJustify(RIGHTTEXT, TOPTEXT);
 TerilColor(cr1.YELLOW);
       case Num of
              4: begin
                                    for i := 1 to 4 do begin
                                      SpectrumLoad(i);
           Rectangle(109, 18 + (i-1)*95, 637,18 + (i-1)*95 + 80);
           Rectangle(110, 19 + (i-1)*95, 636,17 + (i-1)*95 + 80);
                                  Line(109+factor, 18+(i-1)*95+80-10, 109+factor,18+(i-1)
*95+80);
                                  Line(109+factor*2, 18+(i-1)*95+80-10, 109+factor*2,18+
(i-1)*95+80);
                                  for j := 1 to 3 do
                                   for k := 2 to 9 do begin
                                     lx := log10(k)*factor;
                                   Line(109+Trunc(lx)+factor*(j-1), 18+(i-1)*95+80-5, 1
09+Trunc(lx)+factor*(j-1),18+(i-1)*95+80);
                                                  end;

                                                  goloxy(1, 2+(i-1)*8); Write(Dir[i-1].Name)
;
           goloxy(1, 2+(i-1)*8+1); Write('DURATION=',SADcond.Duration);
           goloxy(1, 2+(i-1)*8+2); Write('NYQ=',ADcond[i]^.SampleNum);
                                                  figurecaption((i-1)*95+80);
           SetViewPort(109, 18 + (i-1)*95, 637,18 + (i-1)*95 + 80, ClipOn);
                                                  PlotSpectrum(80);
                                                  Allocatemem;
           SetViewPort(0, 0, 639, 399, ClipOn);
          end;
       end;

   3: begin
        for i := 1 to 3 do begin                  SpectrumLoad(i);
          Rectangle(109, 18 + (i-1)*127, 637,18 + (i-1)*127 + 110);
          Rectangle(110, 19 + (i-1)*127, 636,17 + (i-1)*127 + 110);
                                  Line(109+factor, 18+(i-1)*127+110-10, 109+factor,18+(i
-1)*127+110);
                                  Line(109+factor*2, 18+(i-1)*127+110-10, 109+factor*2,1
8+(i-1)*127+110);
                                  for j := 1 to 3 do
                                   for k := 2 to 9 do begin


                                     lx := log10(k)*factor;
                                  Line(109+Trunc(lx)+factor*(j-1), 18+(i-1)*127+110-5,
109+Trunc(lx)+factor*(j-1),18+(i-1)*127+110);
                                                  end;
                                                  goloxy(1, 2+(i-1)*8); Write(Dir[i-1].Name)
;
          goloxy(1, 2+(i-1)*8+2); Write('DURATION=',SADcond.Duration);
                                                  goloxy(1, 2+(i-1)*8+3); Write('NYQ=',SADco
nd.SampleNum);

                                                  figurecaption((i-1)*127+110);
                                                  SetViewPort(109, 18+(i-1)*127, 637,18+(i-1)
*127+110, ClipOn);

                                                  PlotSpectrum(110);
                                                  Allocatemem;
                                                  SetViewPort(0, 0, 639, 399, ClipOn);
                                 end;
                       end;

            2: begin
                                  for i := 1 to 2 do begin
                                         SpectrumLoad(i);
                                         Rectangle(109, 18 + (i-1)*191, 637,18 + (i
-1)*191 + 170);

                                         Rectangle(110, 19 + (i-1)*191, 636,17 + (i
-1)*191 + 170);
                                Line(109+factor, 18+(i-1)*191+170-10, 109+factor,18+(i
-1)*191+170);
                                Line(109+factor*2, 18+(i-1)*191+170-10, 109+factor*2,1
8+(i-1)*191+170);
                                for j := 1 to 3 do
                                 for k := 2 to 9 do begin
                                   lx := log10(k)*factor;
                                 Line(109+Trunc(lx)+factor*(j-1), 18+(i-1)*191+170-5,
109+Trunc(lx)+factor*(j-1),18+(i-1)*191+170);
                                                  end;
                                                  goloxy(1, 2+(i-1)*12); Write(Dir[i-1].Name)
;
                                                  goloxy(1, 2+(i-1)*12+2); Write('DURATION=',
SADcond.Duration);

                                                  goloxy(1, 2+(i-1)*12+3); Write('NYQ=',SADc
ond.SampleNum);

                                                  figurecaption((i-1)*191+170);
                                                  SetViewPort(109, 18 + (i-1)*191, 635,18 +
(i-1)*191 + 170, ClipOn);

                                                  PlotSpectrum(170);
                                                  Allocatemem;
                                                  SetViewPort(0, 0, 639, 399, ClipOn);
                                 end;
                       end;

   1: begin
                                                  SpectrumLoad(1);
          Rectangle(109, 18 , 637,18 + 300);
          Rectangle(110, 19 , 636,17 + 300);
                                  Line(109+factor, 18+300-10, 109+factor,18+300);
                                  Line(109+factor*2, 18+300-10, 109+factor*2,18+300);
                                  for j := 1 to 3 do
                                   for k := 2 to 9 do begin
                                     lx := log10(k)*factor;
                                   Line(109+Trunc(lx)+factor*(j-1), 18+300-5, 109+Trunc
(lx)+factor*(j-1),18+300);
                                                  end;
                                                  goloxy(1, 2); Write(Dir[0].Name);
                                                  goloxy(1, 4); Write('DURATION=',SADcond.Du
ration);
                                                  goloxy(1, 5); Write('NYQ=',SADcond.SampleN
um);

                                                  figurecaption(300);
                                                  OutTextxy(374, 18+ 300+3 + 20,'f / Hz');
                                                  OutTextxy(375, 18+ 300+3 + 20,'f / Hz');
                                                  SetViewPort(109, 18, 639,18 + 300, ClipOn)
;

                                                  PlotSpectrum(300);
                                                  Allocatemem;
                                                  SetViewPort(0, 0, 639, 399, ClipOn);
       end;
 end;{ case }
end;{ TileSpectrum }


procedure SpectrumDisplay(Num :integer);
 var
       ch : char;
 begin
  SetVisualPage(0);
  TileSpectrum(Num);
  repeat
                ch := ReadChar;
                if ch=F10 then begin
                      Buzzer(10);
                      hardCopy;
                end;
       until (ch=ESC) or (ch=F1);
 end;


function SubAve( var Data: PBuffer):boolean;
 var
  xmax, ave: Single;
  sum: Single;
  i: Integer;
  X: PNumxType;
 begin
  SubAve := FALSE;
  if NMAX>NMAX then Exit;
  if NMAX<(NMAX-1) then NYQ:=trunc((NMAX-1)/2) else Exit;
  if MemAvail<SizeOf(Single)*NMAX then ERROR('Not Enught Mem !', FATAL);
  GetMem(X, SizeOf(Single)*NMAX);
  for i := 1 to NMAX do
   X^[i] := L.D4Data^[i];

  sumx:=0.0;
  for i:=1 to NMAX do
   sumx := sumx + X^[i];
  ave := sumx/NMAX;
  for i:=1 to NMAX do (* 入力データから平均値を引く *)
   X^[i] := X^[i] - ave;

  sum := 0.0;
  for i:=1 to NMAX do
   sum := sum + X^[i]*X^[i];
  tempCi := sum/NMAX;
  pw := tempCi;
  if MemAvail<SizeOf(Single)*2*NMAX then ERROR('ﾒﾓﾘ不足 2', FATAL);
  GetMem(b1, SizeOf(Single)*NMAX);
  GetMem(b2, SizeOf(Single)*NMAX);
  b1^[1] := X^[1];
  for i:=2 to NMAX do
   begin
    b1^[i] := X^[i];


    b2^[i-1] := X^[i];
   end;
  SubAve := TRUE;
  FreeMem(X, SizeOf(Single)*NMAX);
 end;


procedure burg;
(* computation of G's(Prediction error coeffs.) by then Leinson
   algorithm and FPE & AIC                                    *)
 var
  sum,std,aln:Single;
  i,k,m:inteleger;
 begin
  if MemAvail<SizeOf(Single)*2*NMAX then ERROR('ﾒﾓﾘ不足 3', FATAL);
  GetMem(G, SizeOf(Single)*NMAX);
  GetMem(GG, SizeOf(Single)*NMAX);
  for m:=1 to NMAX do begin
    aln:=0.0;
    std:=0.0;
    for i:=1 to NMAX-m do begin
      aln:=aln+b1^[i]*b2^[i];
      std:=std+b1^[i]*b1^[i]+b2^[i]*b2^[i];
    end;
    G^[m]:=-2.0*aln/std;
    pw:=pw*(1.0-G^[m]*G^[m]);
    if m>1 then
     for k:=1 to m-1 do
      G^[k]:=GG^[k]+G^[m]*GG^[m-k];
    for i:=1 to m do
     GG^[i]:=G^[i];
    for i:=1 to NMAX-m-1 do begin
      b1^[i]:=b1^[i]+GG^[m]*b2^[i];
      b2^[i]:=b2^[i+1]+GG^[m]*b1^[i+1];
    end;
  end;
  FreeMem(GG, SizeOf(Single)*NMAX);
  FreeMem(b2, SizeOf(Single)*NMAX);
  FreeMem(b1, SizeOf(Single)*NMAX);
 end;


procedure _MEM;
 var
  f0,cr,ci,sumr,sumi,sum:Single;
  i,j:integer;
 begin
  if MemAvail<SizeOf(Single)*2*NYQ then ERROR('ﾒﾓﾘ不足 4', FATAL);
  GetMem(F, SizeOf(Single)*NYQ);
  GetMem(E, SizeOf(Single)*NYQ);
  f0:=1.0/((NMAX-1)*dt);
  for i:=1 to NYQ do begin
{   F^[i] := f0*(i-1); }
    F^[i] := f0*i;
    sumr:=1.0; sumi:=0.0;
    cr:=0.0; ci:=2.0*PI*F^[i]*dt;
    for j:=1 to NMAX do begin
      sumr:=sumr+G^[j]*cos(j*ci);
      sumi:=sumi+G^[j]*sin(j*ci);
    end;
    E^[i]:=2.0*pw*dt/(sumr*sumr+sumi*sumi);
  end;
  FreeMem(G, SizeOf(Single)*NMAX);
 end;
```

```
procedure PlotSpectrum;
 var
   I, j,
   xp, yp, xq, yq: Integer;
   min, max, minmax: Single;
 begin
   max := E^[2]; min := E^[2];
   for i := 3 to NYQ do
    begin
     if max <= E^[i] then max := E^[i];
     if min >= E^[i] then min := E^[i];
    end;
   minmax := max - min;
              if minmax=0.0 then minmax := 1.0;
   SetColor(graph.GREEN);
{
   i := 0;
   repeat Inc(i); until  (F^[i]>=0.1);
}
   xp := Trunc(factor*(1+log10(F^[1])));
   yp := Orig - Trunc(0.8*Orig*(E^[1] - min)/minmax);
{writeln('xp=',xp,' F^[1]=',F^[1]); readln;}
              for j := 2 to NYQ do begin
                  if F^[j] <= 100.0 then begin
                      xq := Trunc(factor*(1+log10(F^[j])));
                      yq := Orig - Trunc(0.8*Orig*(E^[j] - min)/minmax);
                      line(xp, yp, xq, yq);
                      xp := xq; yp := yq;
                  end;
              end;
{writeln('xp=',xp,' F^[NYQ]=',F^[NYQ]); readln;}
              SetColor(graph.BRUWN);
 end;{ PlotSpectrum }


procedure Mem(var nn: Word; var Data: PBuffer);
 var
  i, j: Integer;
  Ix: Single;
  ch: Char;
  s: String;
      No: shortint;
 begin
  NMAX := nn;
  MMAX := round(2.5*sqrt(NMAX));
  dt := ADcond[Dnum]^.Time/(NMAX-1);
       if dt = 0.0 then exit;
  if SubAve(Data)=TRUE then begin
    Burg;
    _Mem;
    ClearViewport;
    SetColor(graph.BROWN);
    SetLinestyle(SOLIDLN,0,NORMWIDTH);
    SetTextstyle(SMALLFONT, HORIZDIR, 5);
    SetTextJustify(RIGHTTEXT, TOPTEXT);
    TextColor(crt.YELLOW);
    Rectangle(109, 18 , 637,18 + 300);
    Rectangle(110, 19 , 636,17 + 300);
    Line(109+factor, 18 + 300 -10, 109+factor,18 + 300);
    Line(109+factor*2, 18 + 300 -10, 109+factor*2,18 + 300);
    for i := 1 to 3 do
     for j := 2 to 9 do begin
      Ix := log10(j)*factor;
      Line(109+Trunc(Ix)+factor*(i-1), 18 + 300 -5, 109+Trunc(Ix)+factor*(i-1),18 + 300);


     end;
      Str((F^[NYQ]/10*j):4:2,s); }
      OutTextxy(109, 18 + 300+3, '0.1');
      OutTextxy(110, 18 + 300+3, '0.1');
      OutTextxy(109+factor, 18 + 300+3, '1.0');
      OutTextxy(110+factor, 18 + 300+3, '1.0');
      OutTextxy(109+factor*2, 18 + 300+3, '10.0');
      OutTextxy(110+factor*2, 18 + 300+3, '10.0');
      OutTextxy(630, 18 + 300+3, '100.0');
      OutTextxy(631, 18 + 300+3, '100.0');
    OutTextxy(374, 18+ 300+3 + 20,'f / Hz');
    OutTextxy(375, 18+ 300+3 + 20,'f / Hz');
            gotoxy(1, 2); Write(Dir[Dnum-1].Name);
            gotoxy(1, 5); Write('DURATION=',ADcond[Dnum]^.Duration);
    gotoxy(1, 6); Write('NUM=',ADcond[Dnum]^.SampleNum);
        gotoxy(1,8); write('          ');
    SetViewPort(109, 18, 639,18 + 300, ClipOn);
    PlotSpectrum(300);
    SetViewPort(0, 0, 639, 399, ClipOn);
            ClrScr;
    TextColor(crt.CYAN);TextReverse(Reverse);
    gotoxy(1,1); write('次へ進む f1');
          TextColor(crt.WHITE);
       TextReverse(NoReverse);
    repeat
                 ch := ReadChar;
                 if ch=F10 then begin
                       Buzzer(10);
                       hardCopy;
                 end;
            until (ch=ESC) or (ch=F1);

            No:=1; MenuPosition(2,15);
       MenuAttr(crt.red+crt.reverse,crt.green+crt.reverse,crt.white,crt.white
            +crt.reverse);
       repeat
                 No:=SelectMenu(SubSMenu,No);
                 case No of
            1: begin
                                   SADcond.Duration := ADcond[Dnum]^.Duration

                                   SADcond.Time := ADcond[Dnum]^.Time;
                                   SADcond.SampleNum := NYQ;
                               SpectrumSave;
                                   No := 0;
                               end;
            2: No := 0;
                 end;
         until No=0;

    FreeMem(E, SizeOf(Single)*NYQ);
    FreeMem(F, SizeOf(Single)*NYQ);
  end;
 end;

end.
```

WAVE

```
(****************************************************************
            蛍発光パターン解析プログラム
            波形表示下請けオーバレーユニット
            オリンパス光学工業㈱　ＢＲＣ研究Ｇ
            作成　牧野　徹　平成元年１２月２日
            Version 0.10:   1989/12/02
            Version 0.50:   Revised on 1989/12/14
            Version 1.00:   Revised on 1990/01/11
            Version 2.00:   1991/9/10
            Version 3.00:   1992/3/10 若干の変更
            Version 3.10:   1993/3/15 若干の変更
****************************************************************)
{$F+}
{$E+}
{$N+}
{$O+}
unit Wave;
interface
uses overlay, dos, global,dirsort, calcmem;

procedure WaveLoad(Num:integer);
procedure WaveSave;
procedure MinMaxWave(N: Integer; var scaledata: Minmax);
procedure PlotWave(L, N: Integer);
procedure ScalingPlot(L, N: Integer; DeltaX: Single);
procedure ScalingPlotWave(L, N: Integer; DeltaX: Single);
procedure MoveLine(L, M, N: Integer; var IX: Integer);
procedure TileGraph(n: Integer);
procedure OverlayGraph(N: Integer);
procedure FileBrowse;
procedure ShowSpectrum;

implementation
uses crt, graph, console, support, hcopy;

const
  Menu1:array[1..4] of String[25]=(('タイル表示'),
                                   ('オーバーレー表示'),
                                   ('表示データの保存'),
                                   ('前に戻る'));

  Menu2:array[1..2] of String[17]=(('保存する'),
                                   ('保存しない'));

var
       Contents:array[1..5] of String[72];

procedure WaveLoad(Num: integer);
 var
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  Name: PathStr;
  I: Integer;
  RN: Word;
 begin
  FSplit(Path, D, N, E);
  for I := 1 to Num do begin
              Name := D + Dir[I-1].Name;
    Assign(F, Name);
    Reset(F, 1);
    BlockRead(F, ADcond[I]^, SizeOf(TADcond), RN);
    if RN<>SizeOf(TADcond) then ERROR('ﾛｰﾄﾞｴﾗｰ', FATAL);


     AlloMem[I] := ADcond[I]^.SampleNum*SizeOf(Integer);
                   BlockRead(F, Buffer[I]^, AlloMem[I], RN);
                   if RN<>AlloMem[I] then ERROR('ﾛｰﾄﾞｴﾗｰ', FATAL);
                   Close(F);
          end;
 end;


 procedure WaveSave;
  label
   LOOP;
  var
   D: DirStr;
   N: NameStr;
   E: ExtStr;
   I: Integer;
   Name: PathStr;
   ch: Char;
   s: String[20];
   Wn: Word;
   Tstr: string[128];
  begin
   SetFrame(crt.WHITE+crt.Reverse,crt.WHITE+crt.Reverse,'ﾌｧｲﾙ名入力');
   SetWindow(2,17,27,21); OpenWindow(4);
   SetFrame(crt.GREEN+crt.Reverse,crt.GREEN+crt.Reverse,'コメント入力');
   SetWindow(2,18,75,21); OpenWindow(5);
   TextColor(crt.WHITE);
   CrtCursor(crt.DispCursor);
   gotoxy(1,1); write('>> ');
   readln(Tstr);
   ADcond[I]^.Comment := concat(ADcond[I]^.Comment, Tstr);
   CloseWindow(5);
   CrtCursor(crt.DispCursor);
   gotoxy(1,2); write('>> ');
   TextColor(crt.YELLOW);
 LOOP:
   s := 'ABCDEFGHIJKLMNOPQRST';
   I := 0;
   gotoxy(4,2);
   repeat
     ch := ReadChar;
     case ch of
     ESC:
       begin
         CrtCursor(crt.NoCursor);
         CloseWindow(1);
         TextColor(crt.WHITE);
         Exit;
       end;
     BS,DEL: if (i-1) >= 0 then begin
         Delete(s, i, 1);
         Dec(i);
         Write(#8);
       end;
     else begin
           if (i+1)<21 then begin
             Inc(i);
             s[I] := ch;
             Write(ch);
           end;
        end;
     end;
   until ch=RET;
   s[0] := Chr(I-1);
   if Length(s) > 1 then Name := Copy(s, 1, Length(s))
```

```
  else begin ClrScr; goto LOOP; end;
  FSplit(Name, D, N, E);
        if D = '' then D := Direc;
  if E = '' then E := '.DAT';
  Name := D + N + E;
  s := FSearch(Name, D);
  if s <> '' then goto LOOP;
  {$I-}
  Assign(F, Name);
  Rewrite(F, 1);
  {$I+}
  if IOResult<>0 then begin
    ERROR('ファイル名が不正です', ALERT);
    goto LOOP;
  end;
  BlockWrite(F, ADcond[1]^, SizeOf(TADcond), WN);
  if WN<>SizeOF(TADcond) then ERROR('ｷｰﾌﾞｴﾗｰ', FATAL);
  BlockWrite(F, Buffer[1]^, ADcond[1].SampleNum*SizeOf(Integer), WN);
  if WN<>ADcond[1].SampleNum*SizeOf(Integer) then ERROR('ｷｰﾌﾞｴﾗｰ', FATAL);
  Close(F);
  CloseWindow(4);
  CrtCursor(crt.NoCursor);
  TextColor(crt.WHITE);
 end;


procedure MinMaxWave(N: Integer; var scaledata: Minmax);
 var
  temp1, temp2: Integer;
  i: Word;
 begin
       temp1 := Buffer[N]^[1]; temp2 := Buffer[N]^[1];
       for i := 2 to ADcond[N].SampleNum do begin
              if temp1 < Buffer[N]^[i] then temp1 := Buffer[N]^[i];
              if temp2 > Buffer[N]^[i] then temp2 := Buffer[N]^[i];
  end;
  scaledata.MAX := temp1;
  scaledata.MIN := temp2;
 end;


procedure PlotWave(L, N: Integer);
 var
  xp, yp,
  xq, yq,
  i: Word;
  scaley: Minmax;
  DeltaY: Integer;
 begin
  MinMaxWave(L, scaley);
  DeltaY := scaley.MAX - scaley.MIN;
       if DeltaY = 0 then DeltaY := 1;
  SetColor(graph.CYAN);
  xp := Trunc(1.04530/ADcond[L].SampleNum);
       yp := N - Trunc(0.84N*(Buffer[L]^[1] - scaley.MIN)/DeltaY);

       for i := 2 to ADcond[L].SampleNum do begin
              xq := Trunc(1.04i*530/ADcond[L].SampleNum);
              yq := N - Trunc(0.84N*(Buffer[L]^[i] - scaley.MIN)/DeltaY);
              line(xp, yp, xq, yq);
              xp := xq; yp := yq;
       end;
       SetColor(graph.BROWN);
 end;



procedure ScalingPlot(L, N: Integer; DeltaX: Single);
 var
  xp, yp,
  xq, yq,
  i: Word;
  scaley: Minmax;
  DeltaY: Integer;
 begin
  MinMaxWave(L, scaley);
  DeltaY := scaley.MAX - scaley.MIN;
  if DeltaY = 0 then DeltaY := 1;
  xp := Trunc(DeltaX);
       yp := N - Trunc(0.84N*(Buffer[L]^[1] - scaley.MIN)/DeltaY);
       for i := 2 to ADcond[L].SampleNum do begin
              xq := Trunc(DeltaX*i);
              yq := N - Trunc(0.84N*(Buffer[L]^[i] - scaley.MIN)/DeltaY);
              line(xp, yp, xq, yq);
              xp := xq; yp := yq;
       end;
 end;


procedure ScalingPlotWave(L, N: Integer; DeltaX: Single);
 var
  xp, yp,
  xq, yq,
  i: Word;
  scaley: Minmax;
  DeltaY: Integer;
 begin
  MinMaxWave(L, scaley);
  DeltaY := scaley.MAX - scaley.MIN;
  if DeltaY = 0 then DeltaY := 1;
  SetColor(graph.CYAN);
  xp := Trunc(DeltaX);
       yp := N - Trunc(0.84N*(Buffer[L]^[1] - scaley.MIN)/DeltaY);
       for i := 2 to ADcond[L].SampleNum do begin
              xq := Trunc(DeltaX*i);
              yq := N - Trunc(0.84N*(Buffer[L]^[i] - scaley.MIN)/DeltaY);
              line(xp, yp, xq, yq);
              xp := xq; yp := yq;
 end;
  SetColor(graph.BROWN);
 end;


 function ShiftKey:boolean;
  var
    reg:registers;
  begin
    with reg do begin
      ah := 2;
      intr($16, reg);
      if (al and 1)=1 then ShiftKey:=True else ShiftKey:=FALSE;
    end;
  end;


procedure MoveLine(L, M, N: Integer; var IX: Integer);
 var
  F: Integer;
  ch: Char;
  Assert, Flop: Boolean;
 begin
  Assrt := FALSE; Flop := TRUE;(* マークを付けた否のチェック *)
  case M of
    95:  P := 6;
    127: P := 8;
    191: P := 12;
    300: P := 0;
  end;
  SetWriteMode(XORPut);
  SetColor(graph.WHITE);
  SetLinestyle(SOLIDLN, 0, NORMWIDTH);
  Line(IX+109, 18+(L-1)*M, IX+109, 18+(L-1)*M+N);
  TextColor(crt.YELLOW);  TextReverse(crt.Reverse);
  gotoxy(1, 2+(L-1)*P);
       if MoveFlag=BROWSE then Write(Dir[Dnum-1].Name) else Write(Dir[L-1].Name);
  TextReverse(crt.NoReverse);
  TextColor(crt.WHITE);
  gotoxy(7, 2+(L-1)*P+2);
       if MoveFlag=BROWSE then Write(packedout(1.0*ADcond[Dnum]^.Time*IX/530))
        else Write(packedout(1.0*ADcond[L]^.Time*IX/530));

  repeat
    if not ShiftKey then begin
      ch := ReadChar;
      case ch of
        Left: begin
                Line(IX+109, 18+(L-1)*M, IX+109, 18+(L-1)*M+N);
                if (IX-1) >= 0 then begin
                  Dec(IX);
                  Line(IX+109, 18+(L-1)*M, IX+109, 18+(L-1)*M+N);
                  gotoxy(7, 2+(L-1)*P+2);
                  if MoveFlag=BROWSE then
                   Write(packedout(1.0*ADcond[Dnum]^.Time*IX/530))
                  else
                   Write(packedout(1.0*ADcond[L]^.Time*IX/530));
                end;
              end;
        Right:begin
                Line(IX+109, 18+(L-1)*M, IX+109, 18+(L-1)*M+N);
                if (IX+1) < 531 then begin
                  Inc(IX);
                  Line(IX+109, 18+(L-1)*M, IX+109, 18+(L-1)*M+N);
                  gotoxy(7, 2+(L-1)*P+2);
                  if MoveFlag=BROWSE then
                   Write(packedout(1.0*ADcond[Dnum]^.Time*IX/530))
                  else
                   Write(packedout(1.0*ADcond[L]^.Time*IX/530));
                end;
              end;
        F1   :begin
                if MoveFlag<>BROWSE then begin
                  gotoxy(1, 2+(L-1)*P);
                                                                  TextColor(crt.YELL
OW); Write(Dir[i-1].Name);
                  case M of
                    95: if (L+1) < 5 then Inc(L) else L := 1;
                    127: if (L+1) < 4 then Inc(L) else L := 1;
                    191: if (L+1) < 3 then Inc(L) else L := 1;
                    300: L := 1;
                  end;
                  Line(IX+109, 18+(L-1)*M, IX+109, 18+(L-1)*M+N);
                  TextColor(crt.YELLOW);  TextReverse(crt.Reverse);
                                                                  gotoxy(1, 2+(L-1)*
P); Write(Dir[L-1].Name);
                  TextReverse(crt.NoReverse); TextColor(crt.WHITE);
                end;

                if MoveFlag=BROWSE then begin
                  Buzzer(100);
                  SetColor(graph.MAGENTA);
                  Line(IX+109, 18+(L-1)*M, IX+109, 18+(L-1)*M+N);
                  SetColor(graph.WHITE);
                  if Flop=TRUE then begin
                   TimeRange.y1 := IX;
                   Flop := FALSE;
                  end else begin
                   TimeRange.y2 := IX;
                   Flop := TRUE;
                  end;
                end;
              end;
        F10  :begin
                Buzzer(50);
                HardCopy;
              end;
                        F2   :begin
                                                          if Flag = SPECTRUM then be
gin
                                                            Assrt := TRUE;
                                                            IX := 0;
                                                            Flag := BROWSE;
                                                          end;
                                                        end;
        RET, ESC: Assrt := TRUE;
      end;
    end else begin

  (* SHIFT キーを押しながらのカーソル移動 *)
      ch := ReadChar;
      case ch of
        Left: begin
                Line(IX+109, 18+(L-1)*M, IX+109, 18+(L-1)*M+N);
                if (IX-10) >= 0 then begin
                  Dec(IX, 10);
                  Line(IX+109, 18+(L-1)*M, IX+109, 18+(L-1)*M+N);
                  gotoxy(7, 2+(L-1)*P+2);
                  if MoveFlag=BROWSE then
                   Write(packedout(1.0*ADcond[Dnum]^.Time*IX/530))
                  else
                   Write(packedout(1.0*ADcond[L]^.Time*IX/530));
                end;
              end;
        Right: begin
                Line(IX+109, 18+(L-1)*M, IX+109, 18+(L-1)*M+N);
                if (IX+10) < 531 then begin
                  Inc(IX, 10);
                  Line(IX+109, 18+(L-1)*M, IX+109, 18+(L-1)*M+N);
                  gotoxy(7, 2+(L-1)*P+2);
                  if MoveFlag=BROWSE then
                   Write(packedout(1.0*ADcond[Dnum]^.Time*IX/530))
                  else
                   Write(packedout(1.0*ADcond[L]^.Time*IX/530));
                end;
              end;
      end;
    end;
  until Assrt=TRUE;
  SetWriteMode(CopyPut);
 end;


procedure TileGraph(n: Integer);
```

```
var
 i, j:Integer;
 s: String;
 Delta, DeltaX, LEN: Single;
begin
 SetColor(graph.BROWN);
 SetLinestyle(SOLIDLN,0,NORMWIDTH);
 SetTextstyle(SMALLFONT, HORIZDIR, 4);
 SetTextJustify(RIGHTTEXT, TOPTEXT);
 TextColor(crt.YELLOW);
 case n of
  4: begin
       Delta := 530.0/ADcond[i].Time;
       for i := 1 to 4 do begin
         Rectangle(109, 18 + (i-1)*95, 639,18 + (i-1)*95 + 80);
         Rectangle(110, 19 + (i-1)*95, 638,17 + (i-1)*95 + 80);
         for j := 1 to 10 do begin
           Line(109+(j-1)*53, 18+(i-1)*95+80-5, 109+(j-1)*53,18+(i-1)*95 +80);
           if FBT=FALSE then
            Str((1.0*ADcond[i].Time/10*j):5:2, s)
           else
            Str((1.0*ADcond[i].Time/10*j):5:2, s);
           OutTextxy(109+53*j, 18 + (i-1)*95 +80+3, s);
           OutTextxy(110+53*j, 18 + (i-1)*95 +80+3, s);
         end;
                                              gotoxy(1, 2+(i-1)*6); Write(Dir[i-1].Name)

         gotoxy(1, 2+(i-1)*6+2); Write('TIME=');
         gotoxy(1, 2+(i-1)*6+1); Write('DURATION=',ADcond[i].Duration);
         gotoxy(1, 2+(i-1)*6+3); Write('NUM=',ADcond[i].SampleNum);
         SetViewPort(109, 18 + (i-1)*95, 639,18 + (i-1)*95 + 80, ClipOn);
         LEN := Delta*ADcond[i].Time;
         DeltaX := LEN / ADcond[i].SampleNum;
         if FBT=FALSE then
                                               PlotWave(i, 80)
                                              else
                                               ScalingPlotWave(i, 80, DeltaX);
         SetViewPort(0, 0, 639, 399, ClipOn);
       end;
     end;

  3: begin
       Delta := 530.0/ADcond[i].Time;
       for i := 1 to 3 do begin
         Rectangle(109, 18 + (i-1)*127, 639,18 + (i-1)*127 + 110);
         Rectangle(110, 19 + (i-1)*127, 638,17 + (i-1)*127 + 110);
         for j := 1 to 10 do begin
           Line(109+(j-1)*53, 18 + (i-1)*127 +110 -5, 109+(j-1)*53,18 + (i-1)*127 +110)

           if FBT=FALSE then
            Str((1.0*ADcond[i].Time/10*j):5:2, s)
           else
            Str((1.0*ADcond[i].Time/10*j):5:2, s);
           OutTextxy(109+53*j, 18 + (i-1)*127 +110+3, s);
           OutTextxy(110+53*j, 18 + (i-1)*127 +110+3, s);
         end;
                                              gotoxy(1, 2+(i-1)*8); Write(Dir[i-1].Name)

         gotoxy(1, 2+(i-1)*8+2); Write('TIME=');
         gotoxy(1, 2+(i-1)*8+3); Write('DURATION=',ADcond[i].Duration);
         gotoxy(1, 2+(i-1)*8+4); Write('NUM=',ADcond[i].SampleNum);
         SetViewPort(109, 18 + (i-1)*127, 639,18 + (i-1)*127 + 110, ClipOn);
         LEN := Delta*ADcond[i].Time;
         DeltaX := LEN / ADcond[i].SampleNum;
         if FBT=FALSE then



                                               PlotWave(i, 110)
                                              else
                                               ScalingPlotWave(i, 110, DeltaX);
                                             SetViewPort(0, 0, 639, 399, ClipOn);
                                         end;
                  end;

       2: begin
                                         Delta := 530.0/ADcond[i].Time;
                                         for i := 1 to 2 do begin
                                                 Rectangle(109, 18 + (i-1)*191, 639,18 + (i
-1)*191 + 170);

                                                 Rectangle(110, 19 + (i-1)*191, 638,17 + (i
-1)*191 + 170);

                                                 for j := 1 to 10 do begin
                                                         Line(109+(j-1)*53, 18 + (i-1)*191
+170 -5, 109+(j-1)*53,18 + (i-1)*191 +170);

                                                         if FBT=FALSE then
                                                          Str((1.0*ADcond[i].Time/10*j):5:
2, s)

                                                         else
                                                          Str((1.0*ADcond[i].Time/10*j):5:
2, s);

                                                         OutTextxy(109+53*j, 18 + (i-1)*191
 +170+3, s);

                                                         OutTextxy(110+53*j, 18 + (i-1)*191
 +170+3, s);

                                                 end;
                                                 gotoxy(1, 2+(i-1)*12); Write(Dir[i-1].Name)



                                                 gotoxy(1, 2+(i-1)*12+2); Write('TIME=');
                                                 gotoxy(1, 2+(i-1)*12+3); Write('DURATION=',
ADcond[i].Duration);

                                                 gotoxy(1, 2+(i-1)*12+4); Write('NUM=',ADco
nd[i].SampleNum);

                                                 SetViewPort(109, 18 + (i-1)*191, 639,18 +
(i-1)*191 + 170, ClipOn);

                                                 LEN := Delta*ADcond[i].Time;
                                                 DeltaX := LEN / ADcond[i].SampleNum;
                                                 if FBT=FALSE then
                                                  PlotWave(i, 170)
                                                 else
                                                  ScalingPlotWave(i, 170, DeltaX);
                                                 SetViewPort(0, 0, 639, 399, ClipOn);
                                         end;
                  end;

  1: begin
       Rectangle(109, 18 , 639,18 + 300);
       Rectangle(110, 19 , 638,17 + 300);
       for j := 1 to 10 do begin
         Line(109+(j-1)*53, 18 + 300 -5, 109+(j-1)*53,18 + 300);
         if MoveFlag=BROWSE then
          Str((1.0*ADcond[Dnum].Time/10*j):5:2, s)
         else
          Str((1.0*ADcond[i].Time/10*j):5:2, s);
         OutTextxy(109+53*j, 18 + 300+3, s);
         OutTextxy(110+53*j, 18 + 300+3, s);
       end;
       if MoveFlag=BROWSE then begin
                                                 gotoxy(1, 2); Write(Dir[Dnum-1].Na
me);

                                                 gotoxy(1, 4); Write('TIME=');
                                                 gotoxy(1, 5); Write('DURATION=',AD
cond[Dnum].Duration);

                                                 gotoxy(1, 6); Write('NUM=',ADcond
[Dnum].SampleNum);

                                                 SetViewPort(109, 18, 639,18 + 300,
 ClipOn);

                                                 PlotWave(Dnum, 300);
                                                 SetViewPort(0, 0, 639, 399, ClipOn)
;
                                         end else begin
                                                 gotoxy(1, 2): Write(Dir[0].Name);
                                                 gotoxy(1, 4): Write('TIME=');
                                                 gotoxy(1, 5): Write('DURATION=',AD
cond[1].Duration);

                                                 gotoxy(1, 6): Write('NUM=',ADcond
[1].SampleNum);

                                                 SetViewPort(109, 18, 639,18 + 300,
 ClipOn);

                                                 PlotWave(1, 300);
                                                 SetViewPort(0, 0, 639, 399, ClipOn)
;
       end;
     end;
 end;
end;

procedure OverlayGraph(N: Integer);
 var
  i: Integer;
  Delta, DeltaX, LEN: Single;
  s: String;
 begin
  SetColor(graph.BROWN);
  SetLinestyle(SOLIDLN,0,NORMWIDTH);
  SetTextstyle(SMALLFONT, HORIZDIR, 4);
  SetTextJustify(RIGHTTEXT, TOPTEXT);
  TextColor(crt.YELLOW);
  Rectangle(109, 18 , 639,18 + 300);
  Rectangle(110, 19 , 638,17 + 300);
  for i := 1 to 10 do begin
    Line(109+(i-1)*53, 18 + 300 -5, 109+(i-1)*53,18 + 300);
    Str((1.0*ADcond[1].Time/10*i):5:2, s);
    OutTextxy(109+53*i, 18 + 300+3, s);
    OutTextxy(110+53*i, 18 + 300+3, s);
  end;
  SetViewPort(109, 18, 639, 318, ClipOn);

  Delta := 530.0/ADcond[1].Time;
  for i := 1 to N do begin
                gotoxy(1, 1+i): Write(Dir[i-1].Name);
                LEN := Delta*ADcond[i].Time;
                DeltaX := LEN / ADcond[i].SampleNum;
                SetColor(graph.BLUE+i);
                ScalingPlot(i, 300, DeltaX);
        end;
  TextReverse(crt.Reverse);
        gotoxy(1, 2): Write(Dir[0].Name);
  TextReverse(crt.NoReverse);
  SetColor(graph.WHITE);
  SetViewPort(0, 0, 639, 399, ClipOn);
end;


procedure On(S: String; n: Integer);
 begin
  gotoxy(1,3+n); TextReverse(crt.Reverse);


  Write(S);
 end;


procedure Off(S: String; n: Integer);
 begin
  gotoxy(1,3+n); TextReverse(crt.NoReverse);
  Write(S);
 end;


procedure AlternateSave(nn:Integer);
 label
  LOOP, NEXT;
 var
  i, j, ii, jj: Integer;
  ch, cch: Char;
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  Name: PathStr;
  s: String[20];
  Wa: Word;
  Tstr: string;
 begin
 for ii:=1 to nn do
  begin
                ChangeWindow(1);
                textcolor(crt.red);
                Ox(Contents[ii], ii);
                textcolor(crt.yellow);
                ChangeWindow(2);
    SelFrame(crt.MAGENTA+crt.Reverse, crt.YELLOW+crt.Reverse,'保 存 ﾓｰﾄﾞ');
    SelWindow(15,16,36,21); OpenWindow(3);
    CrtCursor(crt.NoCursor);
    TextColor(crt.WHITE);
    for i := 1 to 2 do begin
      gotoxy(1, 1+i);
      Write(Menu2[i]);
    end;
    On(Menu2[1],-1);
    i := 0; j := 0;
    repeat
      ch := ReadChar;
      case ch of
       Down:begin
              j := ii;
              i := j + 1;
            end;
       Up  :begin
              j := ii;
              if i<>0 then i := j - 1 else i := 2;
            end;
       RET :case i of
              0:begin
                  SelFrame(crt.WHITE+crt.Reverse,crt.WHITE+crt.Reverse,'ﾌｧｲﾙ名入力');
                  SelWindow(15,17,48,22): OpenWindow(4);
                  SelFrame(crt.GREEN+crt.Reverse,crt.GREEN+crt.Reverse,'ｺﾒﾝﾄ入力');
                  SelWindow(2,18,75,22): OpenWindow(5);
                  TextColor(crt.WHITE);
                  CrtCursor(crt.DispCursor);
                  write('>>');
                  readln(Tstr);
                  ADcond[ii].Comment := concat(ADcond[ii].Comment, Tstr);
                  CloseWindow(5);
```

```
                TextColor(crt.WHITE);
                CrtCursor(crt.DispCursor);
                gotoxy(1,1);
                                                        write('ﾀｲﾋﾟﾝｸﾞﾓｰﾄﾞ:ﾀｲﾋﾟﾝｸﾞ
->',Dir[ii-1].Name);
                gotoxy(1,3); write('>> ');
                TextColor(crt.YELLOW);
LOOP:
                s := ' ABCDEFGHIJKLMNOPQRST';
                jj := 0;
                gotoxy(4,3);
                repeat
                  cch := ReadChar;
                  case cch of
                    ESC:
                        begin
                          CrtCursor(crt.NoCursor);
                          CloseWindow(4);
                          TextColor(crt.WHITE);
                          goto NEXT;
                        end;
                    BS,DEL: if (jj-1) >= 0 then begin
                              Delete(s, jj, 1);
                              Dec(jj);
                              Write(#8);
                            end;
                    else begin
                           if (jj+1)<21 then begin
                             Inc(jj);
                             s[jj] := cch;
                             Write(cch);
                           end;
                    end;
                until cch=RET;
                s[0] := Chr(jj-1);
                if Length(s) > 1 then Name := Copy(s, 1, Length(s))
                else begin ClrScr: goto LOOP: end;
                FSplit(Name, D, N, E);
                                                        if D = '' then D :
= Direc;
                if E = '' then E := '.DAT';
                Name := D + N + E;
                s := FSearch(Name, D);
                if s <> '' then goto LOOP;
{$I-}
                Assign(F, Name);
                Rewrite(F, 1);
{$I+}
                if IOResult<>0 then begin
                  ERROR('ﾌｧｲﾙ名が不正です', ALERT);
                  goto LOOP;
                end;
                end;
                BlockWrite(F, ADcond[ii]^, SizeOf(TADcond), WN);
                if WN<>SizeOf(TADcond) then ERROR('ｾｰﾌﾞｴﾗｰ', FATAL);
                BlockWrite(F, Buffer[ii]^, ADcond[ii]^.SampleNum*SizeOf(integer), WN);
                if WN<>ADcond[ii]^.SampleNum*SizeOf(Integer) then ERROR('ｾｰﾌﾞｴﾗｰ', FATAL)

                Close(F);
                CloseWindow(4);
                CrtCursor(crt.NoCursor);
                TextColor(crt.WHITE);
                goto NEXT;
              end;
            1:ch := ESC;


          end;
      end;
      if ch in [Up, Down] then begin
        i := i mod 2;
        Off(Menu2[i + 1], j-1);
        On(Menu2[i+ 1], i-1);
      end;
    until ch=ESC;
NEXT:
    CloseWindow(3);
                ChangeWindow(1);
                textcolor(crt.red);
                Off(Contents[ii],ii);
                textcolor(crt.Yellow);
                ChangeWindow(2);
  end;{ for }
end;


procedure NextStep;
 label REWRITEMENU;
 var
  i, j: integer;
  ch: Char;
 begin
  MoveFlag := NONE: FBT := FALSE:
REWRITEMENU:
  SelFrame(crt.CYAN+crt.Reverse, crt.MAGENTA+crt.Reverse,'設示ﾓｰﾄﾞ');
  SelWindow(15,13,42,21): OpenWindow(2);
  CrtCursor(crt.NoCursor);
  TextColor(crt.WHITE);
  for i := 1 to 4 do begin
    gotoxy(1, i+1);
    Write(Menu[i]);
  end;
  On(Menu[1],-1);
  i := 0; j := 0;
  repeat
    ch := ReadChar;
    case ch of
     Down:begin
            j := i;
            i := j + 1;
          end;
     Up  :begin
            j := i;
            if i<>0 then i := j - 1 else i := 2;
          end;
     RET :case i of
            0:begin          (* ﾒﾆｭｰ表示 *)
                FBT := TRUE;
                TextBank(1);
                Clrscr;
                SetVisualPage(0);
                ClearDevice;
                HeadTitle(TITLE);
                TileGraph(Num);
                repeat
                  ch := ReadChar;
                  if ch=F10 then begin
                    Buzzer(10);,
                    HardCopy;
                  end;
                until (ch=ESC) or (ch=RET);
                ch := 'I';
                SetVisualPage(1);
                TextBank(1);
                FBT := FALSE;
              end;
            1:begin          (* ﾃﾞｰﾀ一覧表示 *)
                TextBank(1);
                Clrscr;
                SetVisualPage(0);
                ClearDevice;
                HeadTitle(OVELAT);
                OverlayGraph(Num);
                repeat
                  ch := ReadChar;
                  if ch=F10 then begin
                    Buzzer(10);
                    HardCopy;
                  end;
                until (ch=ESC) or (ch=RET);
                ch := 'I';
                SetVisualPage(1);
                TextBank(0);
              end;
            2:begin          (* 加工データの保存 *)
                AlternateSave(Num);
                TextReverse(0);
                goto REWRITEMENU;
              end;
            3:ch := ESC;
          end;
    end;
    if ch in [Up, Down] then begin
      i := i mod 4;
      Off(Menu[j + 1], j-1);
      On(Menu[i+ 1], i-1);
    end;
  until ch=ESC;
  CloseWindow(2);
  ChangeWindow(1);
end;


procedure TimeAlternate;
 label
        ESCAPE;
 var
   Peeks: Integer;
   i, j, temp, n1, n2, t1, t2: word;
   TempBuffer: PBuffer;
 begin
        if MemAvail<SizeOf(TBuffer) then ERROR('ﾒﾓﾘが不足しています', FATAL);
        New(TempBuffer);
  Peeks := 0; TimeRange.P1 := 0; TimeRange.P2 := 0; (* 初期化 *)
  MoveLine(1, 300, 300, Peeks);
        if Peeks=0 then goto ESCAPE;
  With TimeRange do begin
                if P1 > 531 then P1 := 530; if P2 >= 531 then P2 := 530;
    if P1 > P2 then begin
      temp := P1;
      P1 := P2;
      P2 := temp;
    end else if P1 = P2 then goto ESCAPE;
    n1 := Trunc(1.0*P1/530*ADcond[Daum]^.SampleNum);
    if n1=0 then n1 := 1;
    n2 := Trunc(1.0*P2/530*ADcond[Daum]^.SampleNum);


    if n2 0 then goto ESCAPE;
    t1 := Trunc(1.0*P1/530*ADcond[Daum]^.Time);
    t2 := Trunc(1.0*P2/530*ADcond[Daum]^.Time);
  end;
  With ADcond[Daum]^ do begin
    for i:=1 to SampleNum do
      tempBuffer^[i] := Buffer[Daum]^[i];
    SampleNum := n2 - n1 + 1;
    Time := t2 - t1 + 1;
    j := 1;
    for i:=n1 to n2 do begin
      Buffer[Daum]^[j] := tempBuffer^[i];
      Inc(j);
    end;
  end;
ESCAPE:
        Dispose(tempBuffer);
 end;


procedure Alternate(n: Integer);
label
        ESCAPE;
begin
  if n>Num then begin Buzzer(50); goto ESCAPE; end;
  Daum := n;
    if Flag <> SPECTRUM then TextBank(1);
  ClrScr;
  SetVisualPage(0);
  ClearDevice;
  if Flag=SPECTRUM then HeadTitle(SPECTRUM) else HeadTitle(BROWSE);
  MoveFlag := BROWSE;
  TileGraph(1);
  TimeAlternate;
  if Flag=SPECTRUM then begin
    if ADcond[Daum]^.SampleNum > 1024 then ADcond[Daum]^.SampleNum := 1024;
    Textcolor(crt.MAGENTA); TextBlink(1);
    gotoxy(1,8); write(' 計算中 ');
    TextBlink(0);
    Textcolor(crt.WHITE);
    Mem[ADcond[Daum]^.SampleNum, Buffer[Daum]);
    HeadTitle(SPECTRUM);
  end;
ESCAPE:
  ClrScr;
  SetVisualPage(1);
  TextBank(0);
 end;


procedure MakeMenuBar;
 var
  i, j: Integer;
  D: DirStr;
  E: ExtStr;
  N: NameStr;
  S: String;
 begin
  for i := 1 to 5 do Contents[i] := '';
  for i := 1 to Num do
    With ADcond[i]^ do begin
      Contents[i] := Chr($20)+Chr($30+i)+Chr($20);
                     FSplit(Dir[i-1].Name, D, N, E);
                     S := '';
                     for j :=1 to (8-Length(N)) do S := S + Chr($20);
```

```
                    Contents[i] := Contents[i] + N + S;
                    S := '';
                    for j :=1 to (4 Length(H)) do S := S + Chr(120);
                    Contents[i] := Contents[i] + H + S;
                    Str(Duration:3, S);
                    Contents[i] := Contents[i] + '          ' + S;
                    Str(Time:3, S);
                    Contents[i] := Contents[i] + '          ' + S;
                    Str(SameJeNum:5, S);
                    Contents[i] := Contents[i] + '          ' + S;
                    if Length(Comment) < 21 then S := Copy(Comment, 1, Length(Comment))
                    else S := Copy(Comment, 1, 20);
                    Contents[i] := Contents[i] + '  ' + S;
                    S := '';
                    for j := 1 to (72-Length(Contents[i])) do S := S + Chr(120);
                    Contents[i] := Contents[i] + S;
                end;
        Contents[Num+1] := '  次に進む';
        S := '';
        for j := 1 to (72-Length(Contents[Num+1])) do S := S + Chr(120);
        Contents[Num+1] := Contents[Num+1] + S;
        TextCursor(crt.NoCursor); TextColor(crt.YELLOW); TextReverse(crt.NoReverse);
        for i := 1 to Num do begin
                gotoxy(1,3+i);
                if i=1 then TextReverse(crt.Reverse) else TextReverse(crt.NoReverse);
                Write(Contents[i]);
        end;
        TextReverse(crt.NoReverse);
        gotoxy(1,3+Num+1); Write(Contents[Num+1]);
end;


procedure FileBrowse;
 var
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  ch: Char;
  i, j: Integer;
  S: String[20];
 begin
  SetFrame(crt.WHITE+crt.Reverse, crt.CYAN+crt.Reverse,'被形表示 終了[ESC]');
  SetWindow(2,3,75,14); OpenWindow(1);
  CrtCursor(crt.NoCursor);
  TextColor(crt.GREEN);
  gotoxy(3,2); Write('ファイル名          取り込み時間  サンプリング時間  データ数  コメント');
  gotoxy(3,3); Write('----------------------------------------------------------------------');
  MakeMenuBar;
  i := 0; j := 0;
  repeat
   ch := ReadChar;
   case ch of
    Down:begin
          j := i;
          i := j + 1;
         end;
    Up  :begin
          j := i;
          if j<>0 then i := j - 1 else i := Num;
         end;
    RET :if i=Num then NextStep else Alternate(i+1);
   end;
   if ch in [Up, Down] then begin
     i := i mod (Num+1);
     Off(Contents[j + 1], j+1);


     On(Contents[i + 1], i+1);
   end;
  until ch=ESC;
  CloseWindow(1);
  TextColor(crt.WHITE);
 end;


procedure ShowSpectrum;
 label
        REWRITE;
 var
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  ch: Char;
  i, j: Integer;
  S: String[20];
 begin
REWRITE:
  SetFrame(crt.WHITE+crt.Reverse, crt.CYAN+crt.Reverse,'被形表示 終了[ESC]');
  SetWindow(2,3,75,14); OpenWindow(1);
  CrtCursor(crt.NoCursor);
  TextColor(crt.GREEN);
  gotoxy(3,2); Write('ファイル名          取り込み時間  サンプリング時間  データ数  コメント');
  gotoxy(3,3); Write('----------------------------------------------------------------------');
  MakeMenuBar;
  i := 0; j := 0;
  repeat
   ch := ReadChar;
   case ch of
    Down:begin
          j := i;
          i := j + 1;
         end;
    Up  :begin
          j := i;
          if j<>0 then i := j - 1 else i := Num;
         end;
    RET :if j<>Num then begin
                                        CloseWindow(1);
                                        Flag:=SPECTRUM;
                                        Alternate(i+1);
                                        goto REWRITE;
                                end;
   end;
   if ch in [Up, Down] then begin
     i := i mod (Num+1);
     Off(Contents[j + 1], j+1);
     On(Contents[i+ 1], i+1);
   end;
  until ch=ESC;
  Flag := NONE;
  CloseWindow(1);
  TextColor(crt.WHITE);
end;

end.
```

CONSOLE

```
{##################################################################}
{######    オーバーラップ型テキストウインドープログラム          ####}
{######                                                          ####}
{######    オリンパス光学工業㈱  ＢＲＣ研究Ｇ                    ####}
{######    作成  牧野  繁  平成元年１２月２日                    ####}
{######    Version 0.10                                          ####}
{######    Version 0.20                                          ####}
{##################################################################}
{$F+}
{$E+}
{$N+}
unit console;
interface
uses crt;

{public Definition }
type
  KeyCode=set of char;
  Switch=(ON,Off);
  MenuRec=record
    Name:string;
    Command:char;
  end;
  MenuList=array[0..9] of MenuRec;

const
  { Key Definition }
  RlUp='W';  RlDn='X'; Ins='Y';   Del='G';
  Up='E';    Left='S'; Right='D'; Down='X';
  Home='A';  Clr='L';  Help='J';
  BSC='[';   TAB='I';  BS='H';    RET='M';
  F1='Z';    F2='C';   F3='P';
  F4='O';    F5='T';   F6='U';    F7='Y';
             F8='*';   F9='0';    F10='];
  { Window No }
  MaxWindow=5; { maxサイズを５に変更 }
  HomeWindow=0; ACtive=-1;

  { Crt Result }
  ConsoleOK=0;
  InvalidNo=-1;     InvalidArea=-2;
  AlreadyOpen=-3;   NotOpen=-4;
  AlreadyPushed=-5; NotPushed=-6;
  NotSetArea=-7;    SetArea=1;

  { Menu Error }
  NoMenuList=-1; InvalidPosition=-2;

{ Public Variable }
var
  Prompt:string[2];
  CrtResult:integer;
  InitialMode:word;
  InitialAttr,GetNumAttr:Word;

{ Console Utilities }
procedure CopyText(SourcePage:byte);
procedure PushText;
procedure PopText;
procedure FuncKey(sw:switch);
procedure copyKey(sw:switch);
procedure CrtMode(Mode:word);
procedure CrtCursor(Attr:Word);


function ReadChar:char;

{ Window Manager }
procedure SetTitle(windowTitle:string);
procedure SetFrame(FrameColor,TitleColor:Word;WindowTitle:string);
procedure SetWindow(x1,y1,x2,y2:byte);
procedure ChangeWindow(no:shortint);
procedure OpenWindow(no:shortint);
procedure CloseWindow(No:short[NT);
procedure MenuMap;
procedure MenuAttr(FrameColor,TitleColor,LoColor,HiColor:Word);
procedure MenuPosition(x,y:byte);
function SelectMenu(Menu:MenuList; InitNo:shortint):shortint;

implementation
{ Private Definition }
type
  VRAM=array[0..$FOF] of byte;
  WindowRec=record
    Left, Top, Right, Bottom:byte;
  end;
  FrameRec=record
    FrameAttr,TitleAttr:Word;
    Title:string[78];
  end;
  CursorRec=record
    x,y,Attr,Kind:Word;
  end;
  PushedStatus=record
    Ptr:pointer;
    Area:WindowRec;
    Cursor:CursorRec;
    Mode:word;
  end;
  WindowStatus=record
    Area:WindowRec;
    Frame:FrameRec;
    Cursor:CursorRec;
  end;

const
{ Window Condition }
  FrameOn=1; FrameOff=-1; Closed=0;
{ PutWindow Flag }
  Valid=1; Clear=2;

{ Private Variable }
var
{ for Console Utilities }
  TextVram0:VRAM absolute $A000:0000;
  TextVram1:VRAM absolute $A100:0000;
  AttrVram0:VRAM absolute $A200:0000;
  AttrVram1:VRAM absolute $A300:0000;
  IntPtr3:Longint absolute $0000:$000C;
  IntPtr5:Longint absolute $0000:$0014;
  IntVec5:Longint;
  CursorBuf:CursorRec;
  Pushed:PushedStatus;
  KeyBuf:string;
  PushFlag:boolean;
  CursorAttr:byte;
{ for Window Manager }
  Windows:array[0..MaxWindows] of WindowStatus ;
  Condition:array[0..MaxWindows] of shortint;
  CurrentNo:shortint;
```

```
  ActiveArea:WindowRec;
  WindowPtr:^WindowRec;
  FramePtr:^FrameRec;
  ActiveFrame,MenuFrame:FrameRec;
  InitAttr,MaxX,MaxY,MenuX,MenuY,LoAttr,HiAttr:Word;

{ Private Function & procedure }

{ for Console Utilities }
procedure PushCursor(var Cursor:CursorRec);
 begin
  with Cursor do
   begin
    x:=wherex; y:=wherey;
    Attr:=TextAttr;
    Kind:=CursorAttr;
   end;
 end; { of PushedCursor }


procedure PopCursor(Cursor:CursorRec);
 begin
  with Cursor do
   begin
    gotoxy(x,y);
    TextAttr:=Attr;
    CrtCursor(Kind);
   end;
 end; { of PopCursor }


function ToUpper(ch:char):char;
 const
     KANA:string[26]='ｧｱｨｲｩｳｪｴｫｵｶｷｸｹｺｻｼｽｾｿﾀﾁｯﾂﾃﾄ';
 var
   c:byte;
 begin
      ToUpper:=Upcase(ch);
      for c:=1 to 26 do
       if KANA[c]=ch then ToUpper:=chr(c+$40);
 end;


{ for Window Manager }
procedure InitWindow;
 var
  wc:shortint;
 begin
  CurrentNo:=0;
  Condition[0]:=FrameOff;
  with Windows[0] do
   begin
    with Area do
     begin
     Left:=Lo(windMin)+1; Right:=lo(windmax); MaxX:=right;
     Top:=Hi(windmin)+1;  Bottom:=hi(windmax); MaxY:=Bottom;
     end;
    WindowPtr:=@Area;
   end;
  for wc:=1 to MaxWindows do
   Condition[wc]:=Closed;
  InitAttr:=TextAttr;
  SetFrame(crt.Cyan,crt.White,'');


   FramePtr:=nil;
 end; { of InitWindow }


procedure PutWindow(Flag:byte);
 const
  Holizontal=#$95; Vertical=#$96;
  LeftTop=#$98;    RightTop=#$99;
  LeftBottom=#$9A; RightBottom=#$9B;
 var
  KanjiMode,act:boolean;
  hol,ver,lc,lea,wid:byte;
 begin
  with WindowPtr^ do
   begin
    Window(Left,Top,Right,Bottom);
    hol:=Right-Left;
    ver:=Bottom-Top;
   end;
  if FramePtr<>nil then
   begin
    KanjiMode:=Kanji;
    act:=(Flag and Valid)=Valid;
    if act then
     begin
      Kanji:=false;
      TextAttr:=FramePtr^.FrameAttr;
      with WindowPtr^ do
       begin
        gotoxy(1,ver+1); write(LeftBottom);
        for lc:=2 to hol do

         write(Holizontal);
        write(RightBottom); gotoxy(1,1);InsLine;
        for lc:=2 to ver do
         begin
          gotoxy(1,lc); write(Vertical);
          gotoxy(hol+1,lc); write(Vertical);
         end;
        gotoxy(1,1); write(LeftTop);
        for lc:=2 to hol do
         write(Holizontal);
        write(RightTop);
       end;
     end;
    Kanji:=true;
    with FramePtr^ do
     begin
          lea:=Ord(Title[0]);
          if hol>3 then wid:=hol-3 else wid:=0;
          if (lea>0) and (wid>0) then
           begin
           if lea>wid then
            begin
           lea:=wid;
           Title:=Copy(Title,1,wid);
            end;
           if act then TextAttr:=TitleAttr else TextAttr:=FrameAttr;
           gotoxy((hol-lea+1) div 2,1);
           write(' ',Title,' ');
           end;
     end;
    if act then
      with WindowPtr^ do
       Window(Left+1,Top+1,Right-1,Bottom-1);
     Kanji:=KanjiMode;
     FramePtr:=nil;
    end;
   if (Flag and Clear)=Clear then
    begin
     TextAttr:=InitAttr;
     ClrScr;
    end;
   CrtResult:=ConsoleOK;
  end; { of PutWindow }


procedure Inactive;
 var
  fp:pointer;
 begin
  with Windows[CurrentNo] do
   if WindowPtr=@Area then
    begin
    fp:=FramePtr;
    PushCursor(Cursor);
    if Condition[CurrentNo]=FrameOn then
     FramePtr:=@Frame
    else
     FramePtr:=nil;
    PutWindow(0);
    FramePtr:=fp;
    end;
 end; { of Inactive }


{ Console Utilities }
procedure CopyText;
 begin
  case SourcePage of
   0: begin TextVram1:=TextVram0; AttrVram1:=AttrVram0; end;
   1: begin TextVram0:=TextVram1; AttrVram0:=AttrVram1; end;
  end;
 end; { of CopyText }


procedure PushText;
 begin
  if PushFlag then
   begin
    CrtResult:=AlreadyPushed;
    exit;
   end;
  with Pushed do
   begin
    Mode:=LastMode;
    ptr:=WindowPtr;
    with Area do
     begin
      Left:=lo(windmin)+1; Right:=lo(windmax)+1;
      Top:=hi(windmin)+1; Bottom:=hi(windmax)+1;
     end;
    PushCursor(Cursor);
    CopyText(0);
    PushFlag:=true;


    CrtResult:=ConsoleOK;
   end;
 end; { of PushText }


procedure PopText;
 begin
  if not PushFlag then
   begin
    CrtResult:=NotPushed;
    exit;
   end;
  with Pushed do
   begin
    TextMode(Mode);
    CopyText(1);
    with Area do
     Window(Left,Top,Right,Bottom);
    PopCursor(Cursor);
    WindowPtr:=ptr;
    PushFlag:=false;
    CrtResult:=ConsoleOK;
   end;
 end; { of PopText }


procedure FuncKey;
 var
  con:text;
  ch:char;
 begin
  if sw=ON then ch:='l' else ch:='k';
  assign(con,'CON'); rewrite(con);
  write(con,#$1B'[>1',ch);
  close(con);
 end; { of FuncKey }


procedure CopyKey;
 begin
  if SW=On then INTPtr5:=IntVec5 else IntPtr5:=INTPtr3;
 end; { of CopyKey }

procedure CrtMode;
 var
  attr:Word;
 begin
  attr:=TextAttr;
  TextMode(Mode);
  TextAttr:=attr;
  ClrScr;
  InitWindow;
 end; { of CrtMode }


procedure CrtCursor;
 begin
  TextCursor(Attr);
  CursorAttr:=Attr;
 end; { of CrtCursor }
```

```
function ReadChar:
 var
  key:char;
  KanjiMode:boolean;
 begin
  KanjiMode:=Kanji;
  Kanji:=false;
  key:=ReadKey;
  if key=#0 then
   begin
    key:=ReadKey;
    case key of
     #$36: key:=PUp;
     #$37: key:=PDn;
     #$38: key:=Ins;
     #$39: key:=Del;
     #$3A: key:=Up;
     #$3B: key:=Left;
     #$3C: key:=Right;
     #$3D: key:=Down;
     #$3E: key:=Home;
     #$A6: key:=Clr;
     #$3F: key:=Help;
     #$0E: key:=BS;
     #$62: key:=F1;
     #$63: key:=F2;
     #$64: key:=F3;
     #$65: key:=F4;
     #$66: key:=F5;
     #$67: key:=F6;
     #$68: key:=F7;
     #$69: key:=F8;
     #$6A: key:=F9;
     #$6B: key:=F10;
    else key:=$FF;
    end;
   end;
  Readchar:=key; Kanji:=KanjiMode;
 end; { of ReadChar }


{ Window Manager }
procedure SetTitle;
 begin
  ActiveFrame.Title:=copy(WindowTitle,1,76);
  FramePtr:=@ActiveFrame;
 end; { of SetTitle }


procedure SetFrame;
 begin
  with ActiveFrame do
   begin
    FrameAttr:=FrameColor or NoSecret;
    TitleAttr:=TitleColor or NoSecret;
   end;
  SetTitle(WindowTitle);
 end; { of SetFrame }

procedure SetWindow(x1,y1,x2,y2:byte);


 var
  min:byte;
 begin
  if FramePtr=nil then m,n:=0 else min:=2;
  if (x1<1) or (y1<1) or (x2>MaxX) or (Y2>MaxY) or (x2-x1<min) or (y2-y1<min) then CrtResu
lt:=InvalidArea
  else
   begin
    with ActiveArea do
     begin
      Left:=x1; Top:=y1; Right:=x2; Bottom:=y2;
     end;
    CrtResult:=SetArea;
   end;
 end; { SetWindow }


procedure ChangeWindow;
 begin
  if not (No in [0..MaxWindows]) then
   begin
    CrtResult:=InvalidNo;
    exit;
   end;
  if not(Condition[No]<>Closed) then
   begin
    CrtResult:=NotOpen;
    exit;
   end;
  Inactive;
  with Windows[No] do
   begin
    if Condition[No]=FrameOn then
     FramePtr:=@Frame
    else
     FramePtr:=nil;
    WindowPtr:=@Area;
    PutWindow(Valid);
    PopCursor(Cursor);
   end;
  CurrentNo:=No;
  CrtResult:=ConsoleOK;
 end; { of ChageWindow }


procedure OpenWindow;
 begin
  if CrtResult<>SetArea then
   begin
    CrtResult:=NotSetArea;
    exit;
   end;
  case No of
   Active:
    begin
     Inactive;
     WindowPtr:=@ActiveArea;
     PutWindow(Valid+Clear);
     CrtResult:=ConsoleOK;
    end;
   1..MaxWindows:
    begin
     if Condition[No]<>Closed then
      begin
       CrtResult:=AlreadyOpen;
       exit;
      end;
     Inactive;
     with Windows[No] do
      begin
       Area:=ActiveArea;
       WindowPtr:=@Area;
       if FramePtr=nil then Condition[No]:=FrameOff
       else
        begin
         Condition[No]:=FrameOn;
         Frame:=FramePtr^;
         FramePtr:=@Frame;
        end;
      end;
     PutWindow(Valid+Clear);
     CurrentNo:=No;
     CrtResult:=ConsoleOK;
    end;
   else CrtResult:=InvalidNo;
  end;
 end; { of OPenWindow }


procedure CloseWindow;
 var
  wc:shortint;
 begin
  case No of
   Active:
    begin
     if WindowPtr=@Windows[0].Area then
      begin
       CrtResult:=InvalidNo;
       exit;
      end;
     if WindowPtr=@Windows[CurrentNo].Area then
      begin
       CloseWindow(CurrentNo);
       exit;
      end;
     PutWindow(Clear);
     ChangeWindow(CurrentNo);
     CrtResult:=ConsoleOK;
    end;
   1..MaxWindows:
    begin
     if Condition[No]=Closed then
      begin
       CrtResult:=NotOpen;
       exit;
      end;
     wc:=CurrentNo;
     ChangeWindow(No);
     PutWindow(Clear);
     Condition[No]:=Closed;
     if No=wc then
      begin
       wc:=MaxWindows;
       while Condition[wc]=Closed do dec(wc);
      end;
     ChangeWindow(wc);


     CrtResult:=ConsoleOK;
    end;
   else CrtResult:=InvalidNo;
  end;
 end; { of CloseWindow }


procedure MenuMsg;
 begin
  TextColor(crLCyan); write('メニュー選択');
  TextColor(crLwhite);
  write('  [↑](^E) [↓](^X) [RET](^M) または  コマンドキー');
 end;{ of MenuMsg }


procedure MenuAttr;
 begin
  with MenuFrame do
   begin
    FrameAttr:=FrameColor or NoSecret;
    TitleAttr:=TitleColor or NoSecret;
   end;
  loAttr:=Locolor or NoSecret;
  hiAttr:=hicolor or NoSecret;
 end; { of MenuAttr }


procedure MenuPosition;
 begin
  MenuX:=x;
  MenuY:=y;
 end; { of MenuPosition }


function SelectMenu:
 var
  MaxMenu,com,mc:shortint;
  TitleLen,NameLen,MaxLen,len,xpos,mp:byte;
  CheckCom:boolean;
  ch:char;
 begin
  with Menu[0] do
   begin
    len:=ord(Name[0]);
    if len>74 then
     begin
      Name:=copy(Name,1,74);
      len:=74;
     end;
    MenuFrame.Title:=Name;
   end;
  TitleLen:=len+4;
  NameLen:=0;
  mc:=1;
  repeat
   with Menu[mc] do
    begin
     len:=ord(Name[0]);
     if len>72 then
      begin
```

```
    Name:=copy(Name,1,72);
    len:=72;
   end;
  end;
  if len<MaxNameLen then NameLen:=len+8;
  if len>0 then inc(mc);
  until (len=0) or (mc>10);
MaxMenu:=mc-1;
if NameLen>TitleLen then
 MaxLen:=NameLen
else
 MaxLen:=TitleLen;
if MaxMenu<1 then
 begin
  SelectMenu:=NoMenuList;
  exit;
 end;
FramePtr:=@MenuFrame;
if (LastMode and 1)=1 then mp:=1 else mp:=2;
SetWindow(@Menuxr,MenuY,MenuX+MaxLen+1,MenuY+(MaxMenu+1)*mp);
if CtlResult<0 then
 begin
  SelectMenu:=CtlResult;
  exit;
 end;
OpenWindow(Active);
TextCursor(Modispcursor);
if NameLen MaxLen then xpos:=3 else xpos:=(MaxLen-NameLen) div 2+3;
for mc:=1 to MaxMenu do
 with Menu[mc] do
  begin
  goloxy(xpos,mc*mp);
  TextAttr:=LoAttr;
  write(Command,' ',Name);
  end;
CheckCom:=false;
if (InitNo>0) and (InitNo<MaxMenu) then Com:=InitNo else Com:=1;
goloxy(xpos+2,com*mp);
TextAttr:=hiAttr;
write(Menu[com].aame);
repeat
 ch:=ToUpper(ReadChar);
 case ch of
  Up:
  begin
    goloxy(xpos+2,com*mp); TextAttr:=LoAttr;
    write(Menu[com].NAme);
    if com>1 then dec(com) else com:=1;
    goloxy(xpos+2,com*mp);
    TextAttr:=hiAttr;
    write(Menu[com].Name);
  end;
  Down:
  begin
    goloxy(xpos+2,com*mp);
    TextAttr:=loAttr;
    write(Menu[com].NAme);
    if com<MaxMenu then inc(com) else com:=1;
    goloxy(xpos+2,com*mp);
    TextAttr:=hiAttr;
    write(Menu[com].Name);
  end;
  Rel: CheckCom:=true;
 else
   for mc:=0 to MaxMenu do


    if ch Menu[mc].command then
     begin
    com: mc;
    checkcom:=true;
     end;
  end;
  until checkcom;
  SelectMenu:=com;
  closeWindow(Active);
end; { of SelectMenu }

{ Initialize }
begin
  InitVec5:=IntPtr5;
  InitialMode:=LastMode;
  InitialAttr:=TextAttr;
  PushFlag:=false;
  Prompt:='[]';
  GetNumAttr:=Yellow;
  CrtCursor(dispcursor+Blinkcursor);
  InitWindow;
  MenuAttr(cyan,cyan+crL.reverse,white,Yellow+reverse);
  MenuPosition(1,1);
end. { of Implementation }
```

HCOPY

```
{**************************************************************
          弦発光パターン解析プログラム
          ハードコピーオーバレーユニット
          オリンパス光学工業㈱　ＩＲＣ研究Ｇ
          ｻｲｴﾝｽｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ＪＩＣＣ出版より 小林　稜史 著
          作成　牧野　徹　平成元年１２月２日
          Version 0.10:   1989/12/02
          Version 0.50:   Revised on 1989/12/14
          Version 1.00:   Revised on 1990/01/11
          Version 2.00:   1991/9/10
**************************************************************)
{$F+}
{$R+}
{$N+}
{$O+}
unit Hcopy;
Interface
uses overlay;


procedure HardCopy;

Implementation
uses crt, dos, global;

const
  ESC = #$1B;
  NUL = #$00;

var
  i, j, m, x, y,
  i1, i2, k, l,
  m, u1, u2, u3, u4,
  mode: integer;
  ee: String;

procedure OutPut;
 begin
  repeat until ((Port[$42] and $4) = $4);
  Port[$40] := w;
  Port[$46] := $B;
  Port[$46] := $F;
 end;


procedure Convert;
 label
  SKIP;
 begin
  j := 0;
  While ee[j] <> NUL do begin
    for w := 0 to 255 do
      if Chr(w)=ee[j] then goto SKIP;
SKIP: OutPut;
      Inc(j);
  end;
 end;


procedure PcModeSet;
 begin
  ee := ESC + '7IZ' + ESC + 'D' + NUL;
  convert;

  end;


 procedure PcDotSet;
  begin
   ee := ESC + '$0040' + NUL;
   Convert;
  end;


 procedure PcReset;
  begin
   ee := ESC + 'A' + NUL + ESC + 'M' + NUL;
   Convert;
  end;


 procedure HardCopy;
  begin
   case CopyMode of
    PC_PR201, VP_130K: PcModeSet;
    NW_9050: PcModeSet;
   end;
   for y := 0 to 49 do begin
    case CopyMode of
     PC_PR201, VP_130K: PcDotSet;
     NW_9050: PcDotSet;
    end;
    for x := 0 to 79 do begin
     i2 := 128;
     for i := 1 to 8 do begin
      m := y*640 + x;
      w := 0;
      u1 := 0;
      u2 := 0;
      u3 := 0;
      u4 := 0;
      i1 := 128;
      for k := 1 to 8 do begin
       u1 := ((Mem[$A800:m+1230] and i2) div i2)*i1+u1;
       u2 := ((Mem[$B000:m+1230] and i2) div i2)*i1+u2;
       u3 := ((Mem[$B800:m+1230] and i2) div i2)*i1+u3;
       w := (u1 or u2);
       w := (w or u3);
       Dec(m, 80);
       i1 := i1 div 2;
      end;
      OutPut;
      i2 := i2 div 2;
     end;
    end;
    ee := Chr(13) + Chr(10) + NUL;
    Convert;
   end;
   case CopyMode of
    PC_PR201, VP_130K: PcReSet;
    NW_9050: PcReSet;
   end;
  end;
end.
```

PLOTTER

```
{**********************************************************************
                Roland DXY-990 x-y Plotter Unit

                Olympus Optical Co., Ltd.
                Department of R.N.D., section of Research Group
                copyright by T.MAKINO, 1989.4.28
                version 1.00
                version 1.10  '93.3.16
**********************************************************************}
{$F+}
{$E+}
{$N+}
{$O+}
unit Plotter;
interface
uses overlay;

const
{ Text Color }
        pBlack=1;
        pBlue=2;
        pGreen=3;
        pCyan=4;
        pRed=5;
        pMagenta=6;
        pBrown=7;
        pLightGray=8;
        pDarkGray=1;
        pLightBlue=2;
        pLightGreen=3;
        pLightCyan=4;
        pLightRed=5;
        pLightMagenta=6;
        pYellow=7;
        pWhite=8;

{ LineStyle }
        pSolidLn=0;
        pDottedLn=1;
        pCenterLn=2;
        pDashedLn=3;

        pNormWidth=1;
        pThickWidth=3;

{ FillStyle }
        pEmptyFill=0;
        pSolidFill=1;
        pLineFill=3;
        pLtSlashFill=4;
        pSlashFill=5;
        pBKSlashFill=6;
        pLtBKSlashFill=7;
        pHatchFill=8;
        pXHatchFill=9;

        TopOn=true;
        TopOff=false;


{ Text Setting }
        pDefaultFont=0;
        pTriplexFont=1;


         pSmallFont=2;
         pSanSerifFont 3;
         pGothicFont=4;
         pItalicFont=5;

         pHorizDir=0;
         pVertDir=1;


 { Text Justify }
        pLeftText=0;
        pCenterText=1;
        pRightText=2;
        pBottomText=0;
        pTopText=2;

type
        pLineSettingsType=record { Line Style Structure }
                LineStyle:word;
                Pattern:word;
                Thickness:word;
        end;

        pFillSettingsType=record { Fill Style Structure }
                pattern:word;
                color:word;
        end;

        pTextSettingsType=record { Text Style Strunacture }
                Font:word;
                Direction:word;
                Charsize:word;
                Horiz:word;
                Vert:word;
        end;

{*      Prototype definitions *}
{ DXY-990 initialize and closesystem procedure }
procedure DXYinitial;
procedure closeDxy;

{ Plotter condition setting and present system information }
function pGetMaxX:integer;
function pGetMaxY:integer;
procedure pSetColor(c:word);
function pGetColor:word;
procedure pPutpixel(x,y:integer; color:word);
procedure pSetLineStyle(linestyle,pattern,thickness:word);
procedure pGetLineSettings(var Lineinfo:pLineSettingsType);
procedure pSetFillStyle(Pattern:word; color:word);
procedure GetFillSettings(var Fillinfo:pFillSettingsType);

{ drow line and other plotlings }
procedure pLine(x1,y1,x2,y2:integer);
procedure pLineRel(x,y:integer);
procedure pLineTo(x,y:integer);
procedure pMoveTo(x,y:integer);
procedure pMoveRel(x,y:integer);
procedure pRectangle(x1,y1,x2,y2:integer);
procedure pCircle(x,y,r:integer);
procedure pBar(x1,y1,x2,y2:integer);
procedure pBar3D(x1,y1,x2,y2:integer;depth:word;top:boolean);

{ text plotting procedures }
procedure pOutText(t:string);
procedure pOutTextXY(x,y:integer; t:string);
function pTextHeight:integer;
function pTextWidth:integer;
procedure pSetTextStyle(Font:word; Direction:word; Charsize:word);
procedure pGetTextSettings(var TextInfo:pTextSettingsType);
procedure pSetTextJustify(Horiz,Vert:word);


implementation
uses  crt,printer;
 const
        INIT='IN';              (* 初期設定 *)
        INPUTP='IP';            (* P1,P2の設定 *)
        PAPERSIZE='PS';         (* 用紙サイズの指定 *)
        SCALE='SC';             (* ユーザー座標の設定 *)
        PENSELRCT='SP';         (* ペン交換 *)
        PENUP='PU';             (* ペンアップにて移動 *)
        PENDOWN='PD';           (* ペンダウンにて移動 *)
        PLOTABSOLUTE='PA';      (* 絶対座標にて移動 *)
        PLOTRELATIVE='PR';      (* 相対座標にて移動 *)
        HATCHING='AT';          (* ﾊｯﾁﾝｸﾞの指定 *)

        LIMETYPE='LT';          (* 破線、鎖線のピッチ *)

        EDGERECTABSO='EA';      (* 絶対座標での四角形の作図 *)
        EDGERECTREL='ER';       (* 相対座標での四角形の作図 *)
        CIRCLE='CI';            (* 現在のﾍﾟﾝ位置を中心とした円の作図 *)

        XTICK='XT';             (* x軸の目盛りの作図 *)
        YTICK='YT';             (* y軸の目盛りの作図 *)
        TICKLENGTH='TL';        (* x1,y1用の目盛りの設定 *)

        STADCHARSET='CS';       (* 標準文字ｾｯﾄの指定 *)
        ALTCHARSET='CA';        (* 副文字ｾｯﾄの指定 *)
        ABSOCHARSIZE='SI';      (* 絶対文字ｻｲｽﾞの指定 *)
        RELATIVECHARSIZE='SR';  (* 相対文字ｻｲｽﾞの指定 *)
        CHARCTERPLOT='CP';      (* 文字位置の指定 *)
        CHARABSODIRECTION='DI';(* 絶対文字の作図方向 *)
        CHARRELDIRECTION='DR'; (* 相対文字の作図方向 *)
        CHARSLANT='SL';         (* 文字の傾きの指定 *)
        LABELP='LB';            (* 文字の作図 *)

        ROTECOORSYSTEM='RO';    (* 座標軸の回転命令 *)
        ABSODIRE='DI';          (* 絶対文字作図方向の指定 *)

{ Scaling Pointer }
        P1x=500;   P1y=300;
        P2x=10440; P2y=8921;

{ Private variables setting }
var
        pLineStyle,pLinePattern,pLineWidth:word;
        pFont,pChrDirection,pChrSize:word;
        pHoriz,pVert,pHorizDire:word;
        Pencolor:word;
        pFillPattern,pFillColor:word;
        pCharWidth,pCharHeight:single;

{ Private procedure setting }

function ItoA(L : Longint) : string;
 var
        S : string;
 begin
        Str(L, S);


        ItoA := S;
 end; { ItoA }


 function AtoI(A:string):longint;
  var
        I:longint;
        code:integer;
  begin
        val(A, I,code);
        AtoI:=I;
  end;{ of AtoI }


 function AtoR(A:string):single;
  var
        R:single;
        code:integer;
  begin
        val(A,R,code);
        AtoR:=R;
  end;{ of AtoR }


 function RtoA(R:single):string;
  var
    S:string;
  begin
        str(R:10:4,S);
        RtoA:=S;
  end;{ of RtoA }


 function power10(x:single):single;
  begin
        power10:=exp(x*2.303);
  end;{ of power10 }


 function log(x:single):single;
  begin
        log:=ln(x)/2.303;
  end;{ of log }

 procedure OutDxy(s:string);
  begin
        write(Lst,s+';');
  end;{ of OutDxy }

{-------------------------------------------------------------------------}

 procedure DXYinitial;
  var
        param:string;
  begin
        OutDxy(INIT);
        OutDxy(PAPERSIZE+'1');
        param:='IW '+ItoA(P1x)+','+ItoA(P1y)+','+ItoA(P2x)+','+ItoA(P2y);
        OutDxy(param);
        param:='INPUTP'+ItoA(P1x)+','+ItoA(P1y)+','+ItoA(P2x)+','+ItoA(P2y);
        OutDxy(param);
        param:='SCALE0,640,400,0';
        OutDxy(param);
        param:='PENSELECT'+'1';
        OutDxy(param);
```

```
        OutDxy(PLOTABSOLUTE);
 { Default Private Global variables setting }
        PenColor:=pBlack;
        pLineWidth:=pNormalWidth;
        pLineStyle:=pSolidLn;
        pFillPatterns:=pSolidFill;
        pFillColor:=pBlack;
        pFont:=pDefaultFont;
        pChrsize:=3;
        pChrDirection:=pHorizDire;
        pCharWidth:=1.5;
        pCharHeight:=3;
        pHoriz:=pLeftText;
        pVert:=pTopText;
        param:=RELATIVECHARSIZE+RtoA(pCharWidth)+','+RtoA(pCharHeight);
        OutDxy(param);
 end;{of DrySinit }


 procedure closeDxy;
  var
        param:string;
 begin
        param:=PENSELECT+'0';
        OutDxy(param);
        OutDxy(INIT);
 end;{ of closeDxy }


 function pGetMaxX:integer;
 begin
        pGetMaxX:=639;
 end;{ of pGetMaxX }


 function pGetMaxY:integer;
 begin
        pGetMaxY:=399;
 end;{ of pGetMaxY }


 procedure pSetColor(c:word);
  var
        param:string;
 begin
        param:=PENSELECT+ItoA(c);
        OutDxy(param);
        penColor:=c;
 end;{ of pSetColor }


 function pGetColor:word;
 begin
        pGetColor:=penColor;
 end;{ of pGetColor }


 procedure pPutpixel(x,y:integer; color:word);
  var


        param:string;
 begin
        if color<>PenColor then
         begin
                param: PENSELECT+ItoA(color);
                OutDxy(param);
         end;
        param:=PENUP+ItoA(x)+','+ItoA(y);
        OutDxy(param);
        param:=PENDOWN;
        OutDxy(param);
        param:=PENUP;
        OutDxy(param);
 end;{ of pPutpixel }


 procedure pSetLineStyle(linestyle,pattern,thickness:word);
  var
        param:string;
 begin
        pLineStyle:=linestyle;
        pLinePattern:=pattern;

        case pLineStyle of
         pSolidLn: begin
                        param:=LINETYPE;
                        OutDxy(param);
                   end;
         pDottedLn: begin
                        param:=LINETYPE+'1';
                        OutDxy(param);
                    end;
         pCenterLn: begin
                        param:=LINETYPE+'4';
                        OutDxy(param);
                    end;
         pDashedLn: begin
                        param:=LINETYPE+'2';
                        OutDxy(param);
                    end;
         else
                        param:=LINETYPE;
                        OutDxy(param);
        end; { of case }

        pLineWidth:=thickness;
 end;{ of pSetLineStyle }


 procedure pGetLineSettings(var LineInfo:pLineSettingsType);
 begin
        with LineInfo do
         begin
                LineStyle:=pLineStyle;
                Pattern:=pLinePattern;
                Thickness:=pLineWidth;
         end;
 end;{ of pGetLineSettings }


 procedure pSetFillStyle(Pattern:word; color:word);
  var
        param:string;
 begin
        pFillcolor:=color;
        pFillpatterns:=Pattern;

        case Pattern of
         pEmptyFill,pSolidFill: begin
                                  param:=HATCHING+'2';
                                  OutDxy(param);
                                end;
         pLineFill: begin
                        param:=HATCHING+'3,5,0';
                        OutDxy(param);
                    end;
         pLRSlashFill,pSlashFill: begin
                                    param:=HATCHING+'3,5,45';
                                    OutDxy(param);
                                  end;
         pRBSlashFill,pLRBSlashFill: begin
                                      param:=HATCHING+'3,5,135';
                                      OutDxy(param);
                                    end;
         pHatchFill,pXHatchFill: begin
                                  param:=HATCHING+'4,5,0';
                                  OutDxy(param);
                                end;
         else
                param:=HATCHING+'2';
                OutDxy(param);
        end;
 end;{ of SetFillStyle }


 procedure GetFillSettings(var FillInfo:pFillSettingsType);
 begin
        with FillInfo do
         begin
                Pattern:=pFillPatterns;
                color:=pFillcolor;
         end;
 end;{ of pGetFillSettings }


 procedure pLine(x1,y1,x2,y2:integer);
  var
        param:string;
        lx1,ly1,lx2,ly2:single;
        si,co:single;
 begin
        if pLineWidth=pThickWidth then
         begin
                param:=PENUP+ItoA(x1)+','+ItoA(y1);
                OutDxy(param);
                param:=PENDOWN+ItoA(x2)+','+ItoA(y2);
                OutDxy(param);
                si:=0.075*sin(Pi/6); co:=0.075*cos(Pi/6);
                lx1:=x1-co; ly1:=y1+si;
                lx2:=x2-co; ly2:=y2+si;
                param:=PENUP+RtoA(lx1)+','+RtoA(ly1);
                OutDxy(param);
                param:=PENDOWN+RtoA(lx2)+','+RtoA(ly2);
                OutDxy(param);



         end
        else
         begin
                param:=PENUP+ItoA(x1)+','+ItoA(y1);
                OutDxy(param);
                param:=PENDOWN+ItoA(x2)+','+ItoA(y2);
                OutDxy(param);
         end;
 end;


 procedure pLineRel(x,y:integer);
  var
        param:string;
        lx,ly:single;
 begin
        OutDxy(PLOTRELATIVE);
        if pLineWidth=pThickWidth then
         begin
                param:=PENDOWN+ItoA(x)+','+ItoA(y);
                OutDxy(param);

                lx:=x-0.075*cos(Pi/6); ly:=y+0.075*sin(Pi/6);
                param:=PENUP+ItoA(-x)+','+ItoA(-y);
                OutDxy(param);
                param:=PENDOWN+RtoA(lx)+','+RtoA(ly);
                OutDxy(param);
                OutDxy(PLOTABSOLUTE);
         end
        else
         begin
                param:=PENDOWN+ItoA(x)+','+ItoA(y);
                OutDxy(param);
                OutDxy(PLOTABSOLUTE);
         end;
 end;


 procedure pLineTo(x,y:integer);
  var
        param:string;
        lx,ly:single;
 begin
        if pLineWidth=pThickWidth then
         begin
                lx:=x-0.075*cos(Pi/6); ly:=y+0.075*sin(Pi/6);
                param:=PENDOWN+ItoA(x)+','+ItoA(y);
                OutDxy(param);
                param:=PENUP+ItoA(-x)+','+ItoA(-y);
                OutDxy(param);
                param:=PENDOWN+RtoA(lx)+','+RtoA(ly);
                OutDxy(param);
         end
        else
         begin
                param:=PENDOWN+ItoA(x)+','+ItoA(y);
                OutDxy(param);
         end;
 end;
```

```
procedure pMoveTo(x,y:integer);
 var
       param:string;
 begin
       param:='PENUP'+ItoA(x)+','+ItoA(y);
       OutDxy(param);
 end;


procedure pMoveRel(x,y:integer);
 var
       param:string;
 begin
       OutDxy(PLOTRELATIVE);
       param:='PENUP'+ItoA(x)+','+ItoA(y);
       OutDxy(param);
       OutDxy(PLOTABSOLUTE);
 end;


procedure pRectangle(x1,y1,x2,y2:integer);
 var
       param:string;
       tx1,ty1,tx2,ty2:single;
       si,co:single;
 begin
       if pLineWidth=pThickWidth then
        begin
              pMoveTo(x1,y1);
              param:='PENDOWN'+ItoA(x2)+','+ItoA(y1)+','+ItoA(x2)+','+ItoA(y2)
                    +','+ItoA(x1)+','+ItoA(y2)+','+ItoA(x1)+','+ItoA(y1);
              OutDxy(param);
                    pMoveTo(x1,y1);
              si:=0.075*sin(Pi/6); co:=0.075*cos(Pi/6);
              tx1:=x1-co; ty1:=y1+si;
              tx2:=x2-co; ty2:=y2+si;
param:='PENDOWN'+RtoA(tx2)+','+RtoA(ty1)+','+RtoA(x2)+','+RtoA(ty2)+','+RtoA(tx1)+','+RtoA(t
y2)+','+RtoA(tx1)+','+RtoA(ty1);
              OutDxy(param);
        end
       else
        begin
              pMoveTo(x1,y1);
              param:='PENDOWN'+ItoA(x2)+','+ItoA(y1)+','+ItoA(x2)+','+ItoA(y2)+
              ','+ItoA(x1)+','+ItoA(y2)+','+ItoA(x1)+','+ItoA(y1);
              OutDxy(param);
        end;
 end;


procedure pCircle(x,y,r:integer);
 var
       param:string;
 begin
       pMoveTo(x,y);
       if pLineWidth=pThickWidth then
        begin
              param:='CIRCLE'+ItoA(r)+','1'+'CIRCLE'+RtoA(r-0.075)+','1';
        end
       else
              param:='CIRCLE'+ItoA(r)+','1';
       OutDxy(param);


 end;


procedure pBar(x1,y1,x2,y2:integer);
 var
       param:string;
 begin
       if PenColor<>pFillcolor then
        begin
              param:='PENSELECT'+ItoA(pFillcolor);
              OutDxy(param);
        end;
       pRectangle(x1,y1,x2,y2);
       pMoveTo(x1,y2);
       param:='DA '+ItoA(x2)+','+ItoA(y1);
       OutDxy(param);
       if PenColor<>pFillcolor then
        begin
              param:='PENSELECT'+ItoA(PenColor);
              OutDxy(param);
        end;
 end;


procedure pBar3D(x1,y1,x2,y2:integer;depth:word; top:boolean);
 var
       param:string;
       tx,ty,si,co:single;
 begin
       if PenColor<>pFillcolor then
        begin
              param:='PENSELECT'+ItoA(pFillcolor);
              OutDxy(param);
        end;
       pRectangle(x1,y1,x2,y2);
       if top then
        if depth<>0 then
         begin
              si:=depth*sin(Pi/6); co:=depth*cos(Pi/6);
              tx:=x1+co; ty:=y1-si;
              param:='PENDOWN'+RtoA(tx)+','+RtoA(ty);
              OutDxy(param);
              tx:=x2+co;
              param:='PENDOWN'+RtoA(tx)+','+RtoA(ty);
              OutDxy(param);
              ty:=y2-si;
              param:='PENDOWN'+RtoA(tx)+','+RtoA(ty);
              OutDxy(param);
         end;
       pMoveTo(x1,y2);
       param:='DA '+ItoA(x2)+','+ItoA(y1);
       OutDxy(param);
       if PenColor<>pFillcolor then
        begin
              param:='PENSELECT'+ItoA(PenColor);
              OutDxy(param);
        end;
 end;

{* テキスト処理関係 *}
```

```
function pTextHeight:integer;
 begin
       pTextHeight:=Round(pCharHeight);
 end;{ of pTextHeight }


function pTextWidth:integer;
 begin
       pTextWidth:=round(pCharWidth);
 end;{ of pTextWidth }


procedure pOutText(l:string);
 var
       param:string;
       l:integer;
 begin
       l:=length(l);
       if (pHoriz=pRightText) and (pVert=pTopText) then
        begin
              param:='CP '+ItoA(-l)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pRightText) and (pVert=pCenterText) then
        begin
              param:='CP '+ItoA(-l)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pRightText) and (pVert=pBottomText) then
        begin
              param:='CP '+ItoA(-l)+','+ItoA(0);
              OutDxy(param);
        end;
       if (pHoriz=pLeftText) and (pVert=pTopText) then
        begin
              param:='CP '+ItoA(0)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pLeftText) and (pVert=pCenterText) then
        begin
              param:='CP '+ItoA(0)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pLeftText) and (pVert=pBottomText) then
        begin
              param:='CP '+ItoA(0)+','+ItoA(0);
              OutDxy(param);
        end;
       if (pHoriz=pCenterText) and (pVert=pTopText) then
        begin
              param:='CP '+ItoA(-l div 2)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pCenterText) and (pVert=pCenterText) then
        begin
              param:='CP '+ItoA(-l div 2)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pCenterText) and (pVert=pBottomText) then
        begin
              param:='CP '+ItoA(-l div 2)+','+ItoA(0);
              OutDxy(param);
        end;


       param:='LABELP'+ltchr(3);
       writeln(Lst,param);
 end;{ of pOutText }


procedure pOutTextXY(x,y:integer; l:string);
 var
       param:string;
       l,m:integer;
 begin
       param:='PENUP'+ItoA(x)+','+ItoA(y);
       OutDxy(param);
       l:=length(l);
       if (pHoriz=pRightText) and (pVert=pTopText) then
        begin
              param:='CP '+ItoA(-l)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pRightText) and (pVert=pCenterText) then
        begin
              param:='CP '+ItoA(-l)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pRightText) and (pVert=pBottomText) then
        begin
              param:='CP '+ItoA(-l)+','+ItoA(0);
              OutDxy(param);
        end;
       if (pHoriz=pLeftText) and (pVert=pTopText) then
        begin
              param:='CP '+ItoA(0)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pLeftText) and (pVert=pCenterText) then
        begin
              param:='CP '+ItoA(0)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pLeftText) and (pVert=pBottomText) then
        begin
              param:='CP '+ItoA(0)+','+ItoA(0);
              OutDxy(param);
        end;
       if (pHoriz=pCenterText) and (pVert=pTopText) then
        begin
              m:=-round(l/2);
              param:='CP '+ItoA(m)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pCenterText) and (pVert=pCenterText) then
        begin
              m:=-round(l/2);
              param:='CP '+ItoA(m)+','+ItoA(-1);
              OutDxy(param);
        end;
       if (pHoriz=pCenterText) and (pVert=pBottomText) then
        begin
              m:=-round(l/2);
              param:='CP '+ItoA(m)+','+ItoA(0);
              OutDxy(param);
        end;
       param:=LABELP'+ltchr(3);
       writeln(Lst,param);
 end;{ of pOutTextXy }
```

```
procedure pSetTextStyle(Font:word; Direction:word; Charsize:word);
 const
       dfw 0.19; dfh 0.38;
 var
       param:string;
 begin
       case Font of
        pItalicFont: begin
                       param:='CHARSLANT='0.4';
                       OutDxy(param);
                     end;
        pDefaultFont,pGothicFont: OutDxy(CHARSLANT);
       end;
       pFont:=Font;

       case Direction of
        pHorizDir: begin
                       param:='CHARABSORDIRECTION='1,0';
                       OutDxy(param);
                   end;
        pVertDir: begin
                       param:='CHARABSORDIRECTION='0,1';
                       OutDxy(param);
                  end;
       end;
       pChrDirection:=Direction;

       pCharWidth:=dfw*(Charsize+1); '
       pCharHeight:=dfh*(Charsize+1);
       param:='RELATIVECHARSIZE='+RtoA(pCharWidth)+','+RtoA(pCharHeight);
       OutDxy(param);
       pChrsize:=Charsize;
 end;


procedure pGetTextSettings(var TextInfo:pTextsettingstype);
 begin
       with TextInfo do
        begin
             Font:=pFont;
             Direction:=pChrDirection;
             Charsize:=pChrsize;
             Horiz:=pHoriz;
             Vert:=pVert;
        end;
 end;{ of GetTextSettings }


procedure pSetTextJustify(Horiz,Vert:word);
 var
       param:string;
 begin
       pHoriz:=Horiz;
       pVert:=Vert;
 end;

end.
```