TR－A－0163

24

眼球運動測定装置と動画像提示装置を用いた半盲視野実験・追従眼球運動実験システム

魚森 謙也　山田 光穂　本郷 仁志

1993. 3.23

ATR視聴覚機構研究所

〒619-02 京都府相楽郡精華町光台 2-2 ☎07749-5-1411

ATR Auditory and Visual Perception Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1411
Facsimile: +81-7749-5-1408

目　次

１．概要

本システムは，『半側視野測定実験』および『スムースパーシュート実験』をサポートするための以下の３つの機能で構成される。

提示画像生成機能（実験の前準備）
各実験での提示画像を生成する。

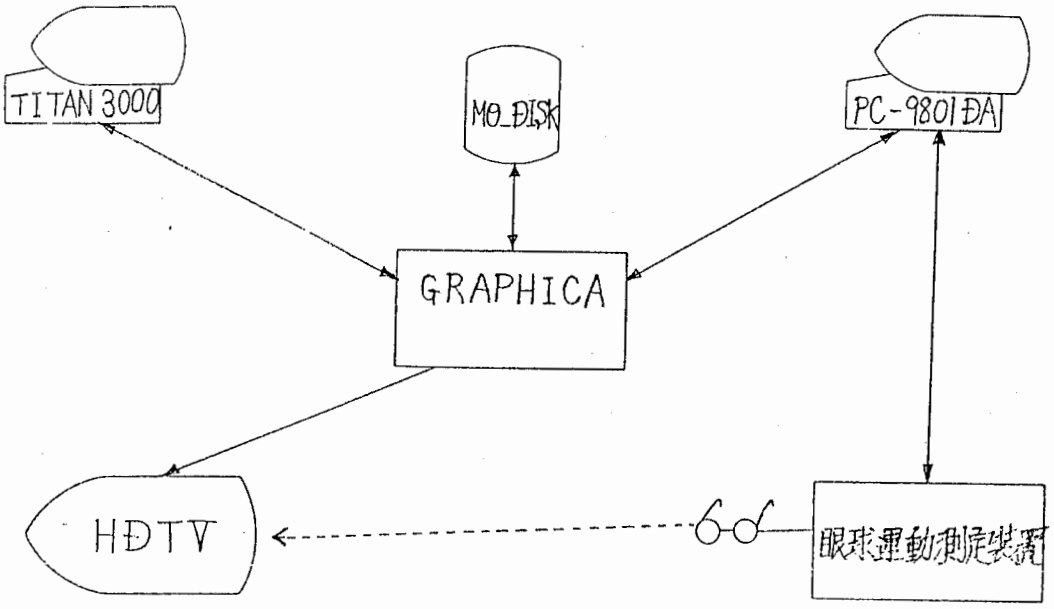
提示画像表示制御機能（実験）
実験中リアルタイムで送られてくる眼球データをもとに次に表示すべき提示画像を判別し実験画面上に表示する。

眼球運動解析機能（実験の後処理）
『半側視野測定実験』で得た眼球データを解析しグラフ表示等の編集処理を行う。

2．ハードウェア構成

2．1　ハードウェア構成図

本システムを実現するためのハードウェア構成図を以下に示す。



画像処理装置　　：GRAPHICA　（以降"GRAPHICA"とする）
眼球運動測定装置：　　（以降"測定装置"とする）
ﾜｰｸｽﾃｰｼｮﾝ　　：TITAN 3000（以降"TITAN"とする）
画像制御装置　　：PC-9801DA　（以降"制御装置"とする）

2．2　各機能における役割り

システムを構成する各機能における上記ハードウェアの役割について記述する。

提示画像生成機能

TITAN　　：提示画像を生成しGRAPHICAへ転送する。
GRAPHICA：TITANから受け取った画像データを光ディスクに保存する。

提示画像表示制御機能

測定装置：眼球運動を解析し制御装置へ送信する。
制御装置：測定装置からの眼球データをもとに次に表示すべき画像をGRAPHICAに指示すると同時にデータの保存も行う。
GRAPHICA：制御装置からの要求に応じてハイビジョンに提示画像を表示する。

眼球運動解析機能

TITAN　　：制御装置で保存した眼球データとGRAPHICAからの画像データをもとに編集処理を行う。
GRAPHICA：TITANで指定された画像データを光ディスクから読み出しTITANへ送信する。

3．ソフトウェア構成

本システムを実現するためのソフトウェア構成を各機能ごとに示す。

提示画像生成機能

・提示画像生成ソフト：『半側視野測定実験』および『スムースパーシュート実験』で使用する提示画像を生成しGRAPHICAの光ディスクへ保存する。
・ＧＬＳＰ　　：【補足】参照

提示画像表示制御機能

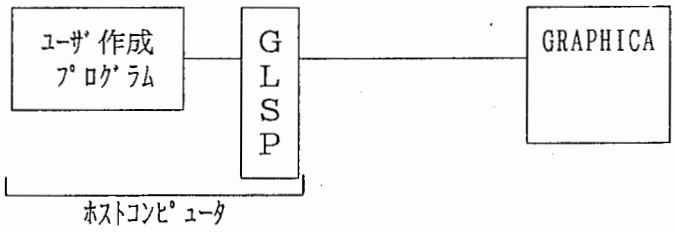
・実験制御ソフト　　：『半側視野測定実験』，『スムースパーシュート実験』を制御する。
・ＧＬＳＰ　　：【補足】参照
・眼球運動検出ソフト：【補足】参照

眼球運動解析機能

・眼球運動解析ソフト：『半側視野測定実験』で得られた結果を編集出力する。
・ＧＬＳＰ　　：【補足】参照

【補足】
『ＧＬＳＰ』，『眼球運動検出ソフト』共に既存のソフトであり詳細についてはそれぞれの『ソフトウェア仕様書』を参照していただきたい。

ＧＬＳＰとは・・・
GRAPHICAをホストコンピュータから使用するための基本ソフトウェアでＣ言語で定義されたサブルーチンパッケージである。



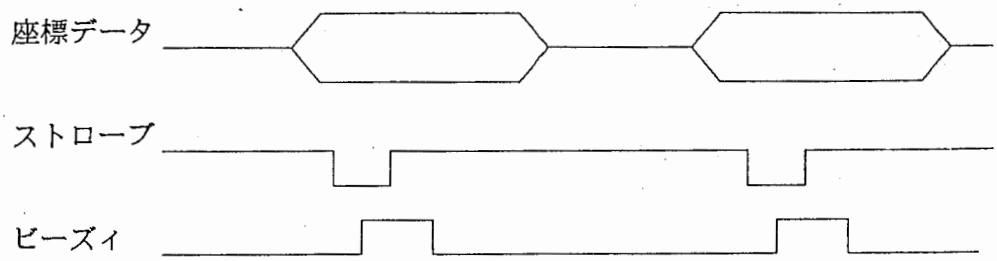
眼球運動検出ソフトとは・・・
眼球運動を測定しデータを制御装置へ送信するソフトで測定装置上に存在する。

4. インタフェース

制御装置と測定装置とのインタフェースについて記述する。

形式　　：パラレルＩ／Ｏ
I/Oボード：ＰＩＯ－４８Ｗ(98)Ｂ－０１

【タイミングチャート】
・データ入出力のタイミングチャートは以下のとおり。



【ＰＩＯボード接続ポート】
・ＰＩＯボードの接続は以下のとおり。

	D7	D6	D5	D4	D3	D2	D1	D0
+0	下位　Ａポート							
	B 02	B 03	B 04	B 05	B 06	B 07	B 08	B 09
+1	上位　Ａポート							
	A 02	A 03	A 04	A 05	A 06	A 07	A 08	A 09
+2	下位　Ｂポート							
	B 12	B 13	B 14	B 15	B 16	B 17	B 18	B 19
+3	上位　Ｂポート							
	A 12	A 13	A 14	A 15	A 16	A 17	A 18	A 19
+4	下位　Ｃポート							
	B 22	B 23	B 24	B 25	B 26	B 27	B 28	B 29
+5	上位　Ｃポート							
	A 22	A 23	A 24	A 25	A 26	A 27	A 28	A 29

ストローブ(B22)

ビーズィ　(A29)

座標値Ｘ　(A02～B09)
座標値Ｙ　(A12～B19)
　この座標値はディスプレイの中心に垂直に向かいＸ座標ならば視線を左に４０°傾けた位置を－４０９６，右に４０°傾けた位置を＋４０９６とし，Ｙ座標ならば視線を下に４０°傾けた位置を－４０９６，上に４０°傾けた位置を＋４０９６として各々の視点を正規化した値である。

データ番号(B27～B29)
　０：一人目の左，１：一人目の右　(150)
　２：二人目の左，３：二人目の右　(200)
　４：三人目の左，５：三人目の右　(300)
　６：四人目の左，７：四人目の右　(370)
　} ｻﾝﾌﾟﾘﾝｸﾞに要する時間（μｓ）

クリック　(A22)
　本システムでは通信中の割り込み処理を禁止しているため実験ソフトに対し実験の終了を通知する手段として制御装置～測定装置の接続ケーブルにハード的な処置を施しマウスで当ビットの操作を行っている。

５．提示画像生成機能

５．１　操作

【ソースファイル】
hitan2/usr3/osato/src/ mkpict.c
mycolor.c
【ｍａｋｅファイル】
dmake

以下の手順で起動する。

hitan2/usr3/osato/src/ で以下のコマンドを入力する。

```
%mkpict
```

５．２　関数

提示画像生成機能で使用した関数を示す。

1. chkpict

機能：

ピックスマップ上のデータをTITAN画面上に表示する。

Calling Sequence：

```
chkpict ()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

2. ctrl_pro

機能：

画像編集作業の主制御を行う。

Calling Sequence：

```
ctrl_pro ()
```

引数：

無し

下位モジュール：

```
initpix
sub1_pro
write_modisk
sub2_pro
write_modisk48
get_image
chkpict
```

戻り値：

無し

3. dcbcheck

機能：

ＧＬＳＰコマンドを使用した場合のリターン値をチェックし異常がある場合処理を中断する。

Calling Sequence：

ｄｃｂｃｈｅｃｋ ()

引数：

無し

下位モジュール：

無し

戻り値：

無し

4. dsp_menu

機能：

メニュー画面の表示を行う。

Calling Sequence：

```
dsp_menu()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

5. end_pro

機能：

ファイルのクローズ等の終了処理を行う。

Calling Sequence：

```
end_pro()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

6. get_image

機能：

ビックスマップをもとにイメージを作製する。

Calling Sequence：

```
get_image()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

7. init_pro

機能：

ピックスマップ作製等の初期処理を行う。

Calling Sequence：

```
init_pro ()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

8. initpix

機能：

ピックスマップのイニシャライズを行う。

Calling Sequence：

```
initpix(bcol)
```

引数：

int　bcol：ピックスマップの背景色。（I）
0＝白，0以外＝黒

下位モジュール：

無し

戻り値：

無し

9. main

機能：

提示画像生成機能のメイン関数。

Calling Sequence：

main ()

引数：

無し

下位モジュール：

init_pro
ctrl_pro
end_pro

戻り値：

無し

10. para_48fname

機能：

『半側視野測定実験』用の画像ファイル名をコンソールより取得する。

Calling Sequence：

```
para_48fname(fp, fname)
```

引数：

```
FILE    *fp     ：画像ファイルポインタ（O）
char    *fname  ：画像ファイル名       （O）
```

下位モジュール：

無し

戻り値：

無し

11. para_cifname

機能：

『スムースパーシュート実験』用の画像ファイル名をコンソールより取得する。

Calling Sequence：

para_cifname (fname)

引数：

char　*fname：画像ファイル名　(O)

下位モジュール：

無し

戻り値：

無し

12. para_size

機能：

『スムースパーシュート実験』のパラメータをコンソールより取得する。

Calling Sequence：

para_size(size)

引数：

int　*size：図形の直径（ドット数）　(O)

下位モジュール：

無し

戻り値：

無し

13. sub1_pro

機能：

『スムースパーシュート実験』用の画像編集処理を制御する。

Calling Sequence：

```
sub1_pro(fname)
```

引数：

char *fname：画像ファイル名。(O)

下位モジュール：

para_48fname

戻り値：

無し

14. sub2_pro

機能：

『半側視野測定実験』用の画像編集処理を制御する。

Calling Sequence：

sub2_pro ()

引数：

無し

下位モジュール：

para_size

戻り値：

無し

15. write_modisk

機能：

『スムースバーシュート実験』用の画像を光ディスクに書き込む。

Calling Sequence：

ｗｒｉｔｅ＿ｍｏｄｉｓｋ（）

引数：

無し

下位モジュール：

ｐａｒａ＿ｃｉｆｎａｍｅ

戻り値：

無し

16.write_modisk48

機能：

『半側視野測定実験』用の画像を光ディスクに書き込む。

Calling Sequence：

ｗｒｉｔｅ＿ｍｏｄｉｓｋ４８（ｆｎａｍｅ）

引数：

ｃｈａｒ　＊ｆｎａｍｅ：画像ファイル名。（Ｉ）

下位モジュール：

無し

戻り値：

無し

６．提示画像表示制御機能

６．１　操作

【ソースファイル】

```
b:¥scc¥scc.c
        scrcnt.c
        crtcnt.c
        menu.c
        dsplist.c
        jikken1.c
        jikken2.c
        kekka.c
        exp2sub.c
        eye.c
        errmsg.c
```

【ｍａｋｅファイル】

```
        mkscc
```

以下の手順で起動する。

b:¥scc¥ で以下のコマンドを入力する。

```
        exp1
```

６．２　関数

提示画像表示制御機能で使用した関数を示す。

1. angconv

機能：

測定装置座標データをスクリーン上のドット位置と座標位置に変換する。

Calling Sequence：

```
angconv(dist, x, y, dotx, doty, angx, angy)
```

引数：

int	dist	：視距離	(I)
int	x	：測定装置データx座標	(I)
int	y	：測定装置データy座標	(I)
int	*dotx	：ドットx位置	(O)
int	*doty	：ドットy位置	(O)
int	*angx	：x座標	(O)
int	*angy	：y座標	(O)

下位モジュール：

無し

戻り値：

無し

2. angle

機能：

眼球データの運動情報を保存する。

Calling Sequence：

ａｎｇｌｅ（ｆｐ１，ｆｐ２，ｆｐ３）

引数：

ＦＩＬＥ	＊ｆｐ１	：眼球データファイルポインタ	（Ｉ）
ＦＩＬＥ	＊ｆｐ２	：テンポラリファイル１ポインタ	（Ｉ）
ＦＩＬＥ	＊ｆｐ３	：テンポラリファイル２ポインタ	（Ｉ）

下位モジュール：

無し

戻り値：

無し

3. chgmode

機能：

『眼球情報２』画面の編集を行う。

Calling Sequence：

ｃｈｇｍｏｄｅ ()

引数：

無し

下位モジュール：

無し

戻り値：

無し

4.chpglc1

機能：

『眼球情報１』画面の指定ページを表示する。

Calling Sequence：

```
int chpglc1(page)
```

引数：

int　page：ページ番号（I）

下位モジュール：

無し

戻り値：

無し

5. clsdeg

機能：
画面のハードコピーを行う。

Calling Sequence：
hcopy (mode)

引数：
int　mode：表示中の画面モード（I）

下位モジュール：
無し

戻り値：
無し

6. crtpix

機能：

ビックスマップ及びイメージ域を生成する。

Calling Sequence：

```
crtpix ()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

7. cutnoiz

機能：

瞬き情報削除処理の制御関数。

Calling Sequence：

cutnoiz(page)

引数：

int　page：編集ページ

下位モジュール：

無し

戻り値：

無し

8. dirconv

機能：

座標１（ｘ１，ｙ１）から座標２（ｘ２，ｙ２）までの移動速度を計算する。

Calling Sequence：

```
int dirconv(x1,y1,x2,y2)
```

引数：

int	x1	：座標１ｘデータ	(I)
int	y1	：座標１ｙデータ	(I)
int	x2	：座標２ｘデータ	(I)
int	y2	：座標２ｙデータ	(I)
int	hz	：周波数	(I)

下位モジュール：

無し

戻り値：

速度

9. dspeyecon

機能：

アイコンの表示。

Calling Sequence：

ｄｓｐｅｙｅｃｏｎ ()

引数：

無し

下位モジュール：

無し

戻り値：

無し

10. dsphead

機能：

眼球ヘッダ情報の表示。

Calling Sequence：

ｄｓｐｈｅａｄ（ｍｏｄｅ）

引数：

ｉｎｔ　　ｍｏｄｅ：画面モード　（Ｉ）
『眼球情報１』＝１，『眼球情報２』＝２，『眼球情報３』＝３

下位モジュール：

無し

戻り値：

無し

11.dspimp

機能：
　　ピックスマップ上の画像情報をウィンドウ内に表示する。

Calling Sequence：
　　ｄｓｐｉｍｐ ()

引数：
　　無し

下位モジュール：
　　無し

戻り値：
　　無し

12. dsploc1

機能：

軌跡ワークファイルのデータを共通領域に展開後，『眼球情報１』画面を表示する。

Calling Sequence：

```
int dsploc1 (page)
```

引数：

int　page：ページ番号（I）

下位モジュール：

chpglc1

戻り値：

正常：０
ファイルオープン失敗：－１

13.dsploc2

機能：

軌跡ワークファイルのデータを共通領域に展開後，『眼球情報２』画面を表示する。

Calling Sequence：

```
int dsploc2()
```

引数：

無し

下位モジュール：

```
linelo2
hensyulo2
```

戻り値：

正常：０
ファイルオープン失敗：－１

14. dsptitan

機能：

編集作業の主制御を行う。

Calling Sequence：

```
dsptitan()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

15. endproc

機能：
　　終了処理を行う。

Calling Sequence：
　　endproc ()

引数：
　　無し

下位モジュール：
　　無し

戻り値：
　　無し

16. getimg

機能：

指定された画像ファイルを読出しGRAPHICAのイメージメモリ上にロードする。

Calling Sequence：

```
int getimg(fname)
```

引数：

char　*fname：画像ファイル名　(I)

下位モジュール：

saveimg

戻り値：

正常：　0
異常：－1

17. hcopy

機能：

画面のハードコピーを行う。

Calling Sequence：

ｈｃｏｐｙ（ｍｏｄｅ）

引数：

ｉｎｔ　ｍｏｄｅ：表示中の画面モード（Ｉ）

下位モジュール：

無し

戻り値：

無し

18. hensyu1o2

機能：

『眼球情報２』画面の編集を行う。

Calling Sequence：

ｈｅｎｓｙｕｌｏ２（）

引数：

無し

下位モジュール：

無し

戻り値：

無し

19. hensyulo22

機能：
『眼球情報２（移動方向編集処理）』の制御関数。

Calling Sequence：
hensyulo22 (min, max)

引数：
int　min：速度範囲指定（最小）
int　max：速度範囲指定（最大）

下位モジュール：
無し

戻り値：
無し

20. limset

機能：

しきい値ファイルの更新を行う。

Calling Sequence：

limset (deg)

引数：

int　deg：しきい値　(I)

下位モジュール：

pctoht

戻り値：

無し

21. linelo1

機能：

『眼球情報１』画面の初期設定を行う。

Calling Sequence：

```
linelo1(page)
```

引数：

int　page：ページ番号（I）

下位モジュール：

無し

戻り値：

無し

22. linelo2

機能：

『眼球情報２』画面の初期設定を行う。

Calling Sequence：

ｌｉｎｅｌｏ２ ()

引数：

無し

下位モジュール：

無し

戻り値：

無し

23. line1o22

機能：
『眼球情報２（移動方向編集画面）』の表示関数。

Calling Sequence：
hensyu1o22 ()

引数：
無し

下位モジュール：
無し

戻り値：
無し

24. loadimg

機能：

GRAPHICAのイメージメモリ上のデータをTITANのイメージ域に保存する。

Calling Sequence：

```
int loadimg()
```

引数：

無し

下位モジュール：

無し

戻り値：

正常：　0
異常：－1

25. locus1

機能：

『眼球情報１』画面表示初期設定。

Calling Sequence：

ｌｏｃｕｓ１ ()

引数：

無し

下位モジュール：

ｄｓｐｌｏｃ１
ｃｌｓｄｅｇ

戻り値：

無し

26. locus2

機能：

『眼球情報２』画面表示初期設定。

Calling Sequence：

ｌｏｃｕｓ２ ()

引数：

無し

下位モジュール：

ｄｓｐｌｏｃ２
ｃｌｓｄｅｇ

戻り値：

無し

27. locus3

機能：

『眼球情報３』画面表示初期設定。

Calling Sequence：

ｌｏｃｕｓ３ ()

引数：

無し

下位モジュール：

ｄｓｐｌｏｃ２
ｃｌｓｄｅｇ

戻り値：

無し

28. main

機能：

提示画像表示制御機能のメイン関数

Calling Sequence：

```
main(argc, argv)
```

引数：

```
int     argc     :引数の数  (I)
char    *argv[]  :引数      (I)
```

下位モジュール：

```
pctoht
getimg
dsptitan
```

戻り値：

無し

29. pctoht

機能：

眼球データファイルを読込み本ソフトウェアの処理形式に合う様に変換する。

Calling Sequence：

```
int pctoht(fname)
```

引数：

```
char  *fname  ：眼球データファイル名  (I)
```

下位モジュール：

seteyeinf

戻り値：

正常：　0
異常：－1

30. rtnmode

機能：

『眼球情報２（移動方向編集処理）』の指定範囲チェック。

Calling Sequence：

```
int rtnmode()
```

引数：

無し

下位モジュール：

無し

戻り値：

正常：　０
異常：－１

31. scrgrf

機能：

『眼球情報１』画面の改ページ制御を行う。

Calling Sequence：

```
scrgrf (page, dirc)
```

引数：

```
int   page：現ページ     (I)
int   dirc：改ページ方向 (I)
```

下位モジュール：

```
chpglc1
```

戻り値：

無し

32. setcolor

機能：

カラー情報設定。

Calling Sequence：

```
setcolor()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

33. setcursor

機能：

カーソル情報設定。

Calling Sequence：

ｓｅｔｃｕｒｓｏｒ（）

引数：

無し

下位モジュール：

無し

戻り値：

無し

34. seteyeinf

機能：

眼球データを保存する。

Calling Sequence：

```
seteyeinf (fp1, fp2, fp3)
```

引数：

FILE	*fp1	：眼球データファイルポインタ	(I)
FILE	*fp2	：眼球データファイルポインタ	(I)
FILE	*fp3	：眼球データファイルポインタ	(I)

下位モジュール：

sethinf
seteyes

戻り値：

無し

35. seteyes

機能：

眼球データの運動情報を保存する。

Calling Sequence：

```
seteyes(fp1, fp2, fp3)
```

引数：

```
FILE   *fp1   :眼球データファイルポインタ   (I)
FILE   *fp2   :テンポラリファイル1ポインタ (I)
FILE   *fp3   :テンポラリファイル2ポインタ (I)
```

下位モジュール：

```
angle
angconv
velconv
dirconv
```

戻り値：

無し

36. setgc

機能：

ＧＣ情報設定。

Calling Sequence：

ｓｅｔｇｃ ()

引数：

無し

下位モジュール：

無し

戻り値：

無し

37. sethead

機能：

眼球ヘッダ情報およびアイコンの表示。

Calling Sequence：

```
sethead (mode)
```

引数：

```
int    mode：画面モード　(I)
```

『眼球情報１』＝１，『眼球情報２』＝２，『眼球情報３』＝３

下位モジュール：

```
dsphead
dspeyecon
```

戻り値：

無し

38. sethinf

機能：

眼球データのヘッダ情報を保存する。

Calling Sequence：

ｓｅｔｈｉｎｆ（ｆｐ１）

引数：

ＦＩＬＥ　＊ｆｐ１　：眼球データファイルポインタ　（Ｉ）

下位モジュール：

無し

戻り値：

無し

39. setmask

機能：

マスク情報設定。

Calling Sequence：

setmask ()

引数：

無し

下位モジュール：

無し

戻り値：

無し

40. setwindow

機能：

ウィンドウ情報設定。

Calling Sequence：

```
setcolor()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

41. spilocus

機能：

ピックスマップ上の画像に軌跡ワークファイルのデータを重ね合わせて表示する。

Calling Sequence：

```
spilocus(char *fname)
```

引数：

char　*fname：テンポラリファイル名　(I)

下位モジュール：

無し

戻り値：

無し

42. updeyes

機能：

しきい値変更にともない眼球ワークファイルを更新する。

Calling Sequence：

```
updeyes ()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

43. velconv

機能：

座標１（ｘ１，ｙ１）から座標２（ｘ２，ｙ２）までの移動速度を計算する。

Calling Sequence：

```
int velconv(x1,y1,x2,y2,hz)
```

引数：

int	x1	：座標１ｘデータ	(I)
int	y1	：座標１ｙデータ	(I)
int	x2	：座標２ｘデータ	(I)
int	y2	：座標２ｙデータ	(I)
int	hz	：周波数	(I) .

下位モジュール：

無し

戻り値：

速度

44. winit

機能：

Ｘウィンドウ生成。

Calling Sequence：

ｗｉｎｉｔ ()

引数：

無し

下位モジュール：

無し

戻り値：

無し

７．眼球運動解析機能

７．１　操作

７．１．１　起動

【ソースファイル】
hitan2/usr3/osato/loc/ locus.h
locus.c
xctrl.c
ldimg.c
mycolor.c
【ｍａｋｅファイル】
kmake

以下の手順で起動する。

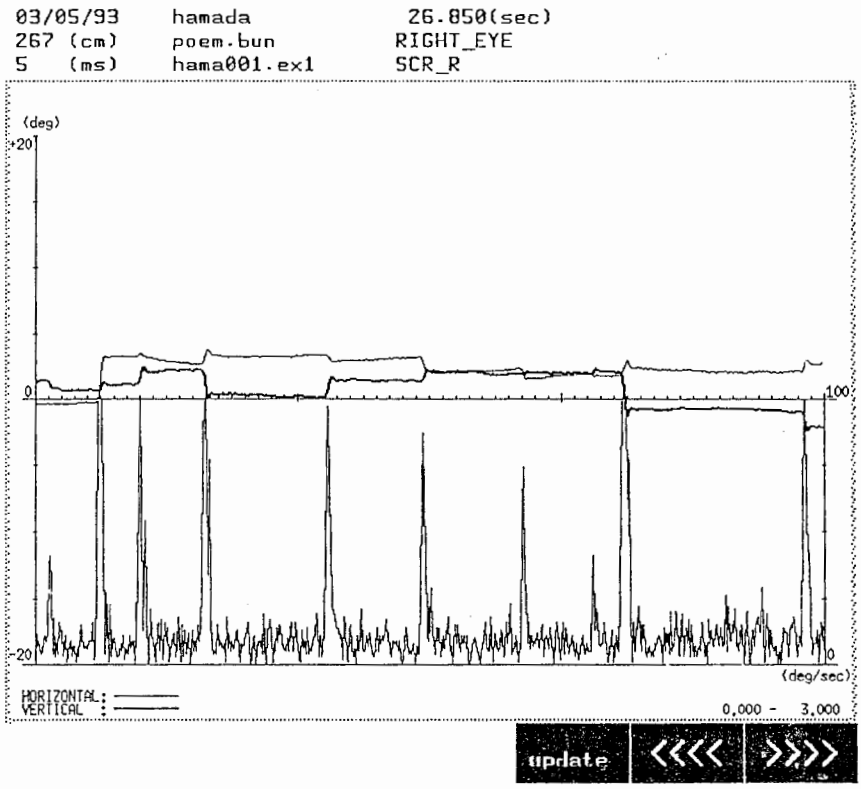
① hitan2/usr3/osato/loc/ で以下のコマンドを入力する。

```
%locus [ngra]
    ngra : GRAPHICAと接続されていない場合またはGRAPHICAからの光ディスク
           の読出しが必要ない場合に設定。
```

② コマンド入力後以下のメッセージが表示され眼球データファイル入力待ちとなるのでファイル名を入力する。

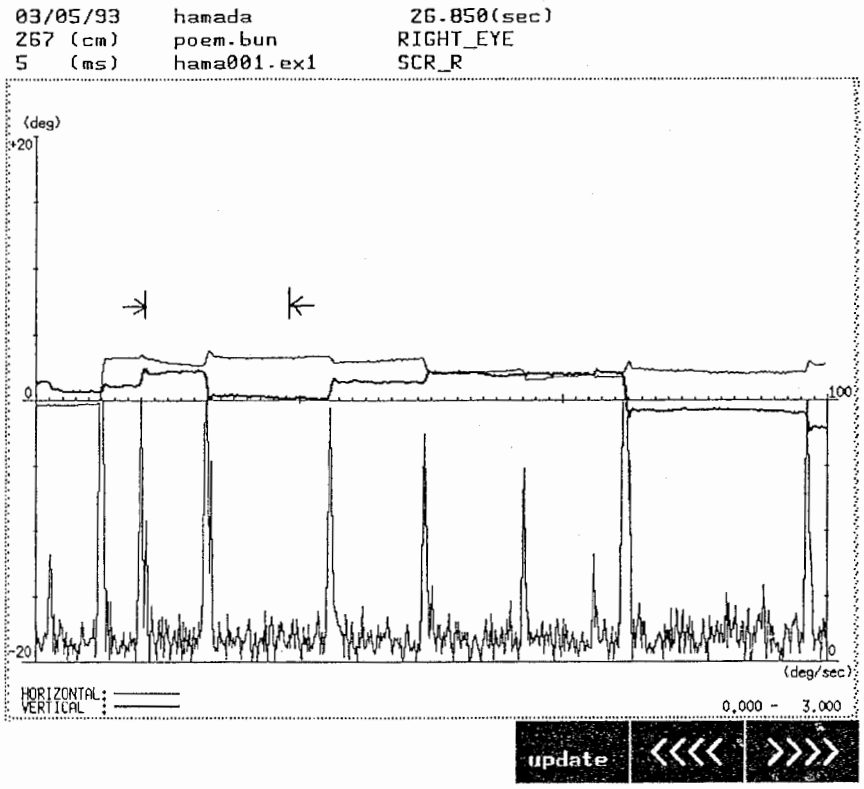
```
%***** << LOCUS      Ver 1.0 >> *****
%Eye Data File ?
%=
```

③ GRAPHICAとの接続がある場合３０秒程接続がない場合には１０秒程で以下の画面が立ち上がる。



7．1．2　操作方法

① [1] の説明



[<<<<] [>>>>] ：改ページ

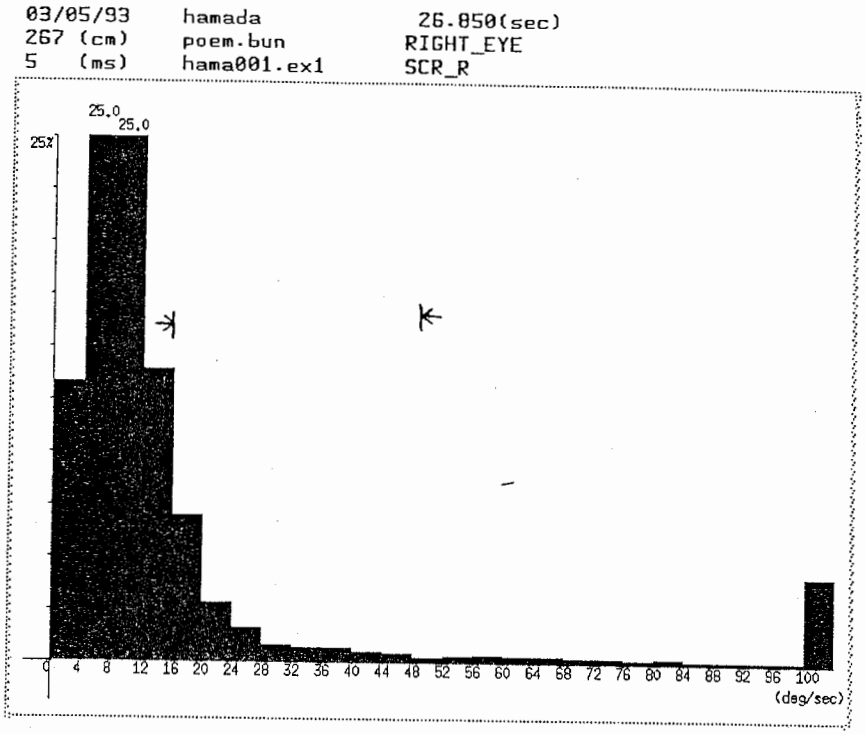
[update] ：瞬きデータの範囲を指定後，当キー押下で眼球データファイルが更新される。但し当キー押下前に他画面へ遷移した場合は範囲指定は取り消される。

【瞬き範囲指定方法】

① →|カーソルで瞬きデータの先頭をクリックする。カーソルの形状が|←に変わったら瞬きデータの終わりを指定する。

② [update] を押下し眼球データファイルを更新する。

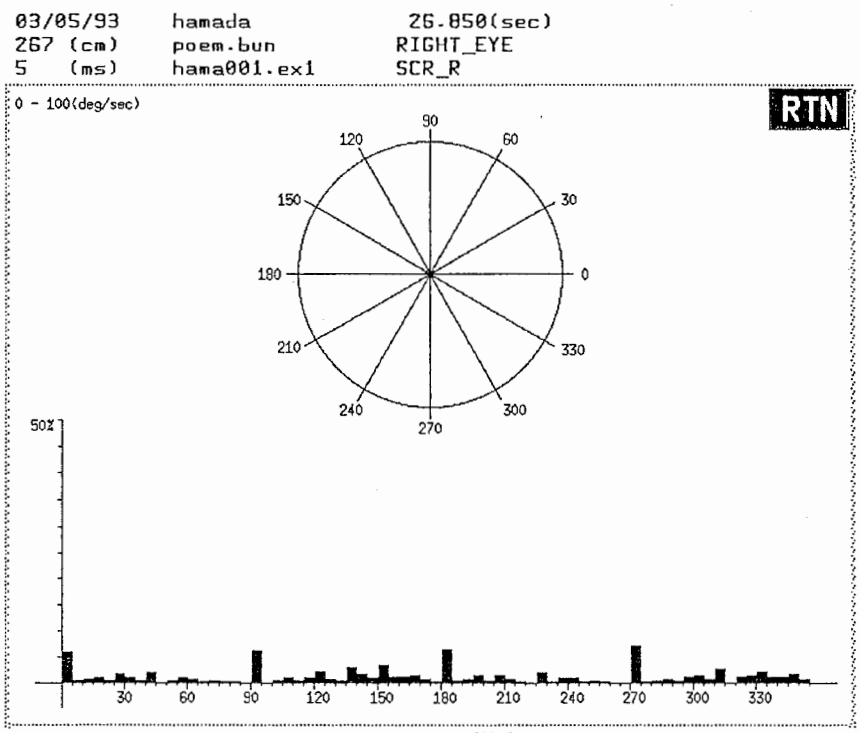
② 2 の説明



【サッカード表示範囲指定方法】

① ⊣カーソルでデータの先頭をクリックする。カーソルの形状が ⊢ に変わったらデータの終わりを指定する。

② サッカード情報表示画面に切り替わる。



③ 3 の説明

locus ：軌跡情報をスーパーインポーズ表示する。

para set ：当キー押下によりしきい値，軌跡の先頭，軌跡の終了を指定可能になる。

７．２　ファイル

７．２．１　眼球データファイル

眼球運動測定装置より得たデータである。（ＡＳＣＩＩデータ）
フォーマットは以下の通り。

（例）

```
04/01/93   ------------------- 実験日
367        ------------------- 視距離
5          ------------------- サンプリング時間
a0000.zuk  ------------------- 画像ファイル名
tetsuzi    ------------------- 被験者名
8          ------------------- 返答値
8          ------------------- 正解値
12000      ------------------- 眼球データ数
0          ------------------- スクロール方向
1          ------------------- 検査対象
  100,  140  ---------------- 眼球データ
 1000,  140
       .
       .
       .
  900,  300
  800,  400
```

7．3　関数

眼球運動解析機能で使用した関数を示す。

1. ana_answer

機能：

画像ファイル名より正解値を解析する。

Calling Sequence：

```
int ana_answer(pict)
```

引数：

char　*pict：画像ファイル名（I）

下位モジュール：

無し

戻り値：

無し

2. ana_memory

機能：

入力データから画像保存場所を解析する。

Calling Sequence：

```
void ana_memory(keepno,
                mem,
                page)
```

引数：

```
int   keepno：保存番号
int   *mem   ：メモリ番号
int   *page  ：ページ番号
```

下位モジュール：

無し

戻り値：

無し

3.com_init

機能：

通信イニシャライズ処理。

Calling Sequence：

com_init ()

引数：

無し

下位モジュール：

無し

戻り値：

無し

4. control_list

機能：

実験画像のリスト表示制御を行う。

Calling Sequence：

```
int control_list(key)
```

引数：

char *key：実験モード　(I)
"_A"：『半側視野測定実験』
"_Y"：『スムースパーシュート実験』

下位モジュール：

dspkmark
menu_list
dsp_list
inp_ctlky
rect_erase
dsp_image_j1
dsp_image_j2

戻り値：

正常：　0
異常：－1

5.date_input

機能：

実験日をコンソールより入力する。

Calling Sequence：

```
void date_input(date)
```

引数：

char　*date：実験日　(O)

下位モジュール：

date_input2

戻り値：

無し

6.date_input2

機能：

実験日入力。

Calling Sequence：

```
void date_input2 (pox,
                  poy,
                  date)
```

引数：

```
short    pox：入力メッセージ表示位置（I）
short    poy：入力メッセージ表示位置（I）
char    *date：実験日              （O）
```

下位モジュール：

無し

戻り値：

無し

7. date_input_year

機能：

実験日（年）入力。

Calling Sequence：

```
void date_input_year (short,
                      short,
                      char *)
```

引数：

```
short     pox：入力メッセージ表示位置（I）
short     poy：入力メッセージ表示位置（I）
char    *date：実験日（年）          （O）
```

下位モジュール：

無し

戻り値：

無し

8. date_input_month

機能：

実験日（月）入力。

Calling Sequence：

```
void date_input_month(short,
                      short,
                      char *)
```

引数：

```
short    pox：入力メッセージ表示位置（I）
short    poy：入力メッセージ表示位置（I）
char   *date：実験日（月）          （O）
```

下位モジュール：

無し

戻り値：

無し

9. date_input_day

機能：

実験日（日）入力。

Calling Sequence：

```
void date_input_day (short,
                     short,
                     char *)
```

引数：

```
short    pox：入力メッセージ表示位置（I）
short    poy：入力メッセージ表示位置（I）
char    *date：実験日（日）          （C）
```

下位モジュール：

無し

戻り値：

無し

10.direct_input

機能：

画像移動方向入力。

Calling Sequence：

```
void direct_input(data)
```

引数：

```
int    *data：移動方向　(O)
                 0：垂直方向
                 1：水平方向
```

下位モジュール：

無し

戻り値：

無し

11.dist_input

機能：
視距離をコンソールより入力する。

Calling Sequence：
```
void dist_input(dist)
```

引数：
int　*dist：視距離　(O)

下位モジュール：
無し

戻り値：
無し

12. dsp_ident

機能：

画像ロード確認メッセージ表示を行う。

Calling Sequence：

```
void dsp_ident (type,
                gno,
                gmode,
                keepno)
```

引数：

int	type	：実験モード	(I)
int	gno	：画像番号	(I)
int	gmode	：スクロール方向	(I)
int	keepno	：保存番号	(I)

下位モジュール：

無し

戻り値：

無し

13. dsp_image_j1

機能：

『半側視野測定実験』の画像をロードする。

Calling Sequence：

```
dsp_image_j1()
```

引数：

無し

下位モジュール：

```
inp_imginf
dsp_kmark
load_pict
pshadow
dsp_keepdt
swindow
```

戻り値：

無し

14.dsp_image_j2

機能：

『スムースパーシュート実験』の画像をロードする。

Calling Sequence：

```
dsp_image_j2()
```

引数：

無し

下位モジュール：

```
inp_imginf
dsp_kmark
load_pict
dsp_keepdt
```

戻り値：

無し

15. dsp_keepdt

機能：
保存画像名表示。

Calling Sequence：
ｄｓｐ＿ｋｅｅｐｄｔ（ｔｙｐｅ）

引数：
ｉｎｔ　　ｔｙｐｅ：実験モード

下位モジュール：
無し

戻り値：
無し

16.dsp_list

機能：

実験画像のリストを表示する。

Calling Sequence：

```
dsp_list()
```

引数：

無し

下位モジュール：

```
xycprintf
rect_erase
```

戻り値：

無し

17. ers_ident

機能：
ロード実行確認メッセージ非表示。

Calling Sequence：

```
void ers_ident (type,
                keepno)
```

引数：

```
int  type   ：実験モード（I）
int  keepno ：保存番号  （I）
```

下位モジュール：
無し

戻り値：
無し

18. expmode_input

機能：

対象眼球の設定。

Calling Sequence：

```
void expmode_input(data)
```

引数：

```
int    *data：実験対象眼球   (O)
              両眼：1
              左眼：2
              右眼：3
```

下位モジュール：

無し

戻り値：

無し

19. eye_move

機能：
眼球データ取得。

Calling Sequence：

```
eye_move (lxd,
          lyd,
          rxd,
          ryd,
          mhit)
```

引数：

```
int   *lxd：左目Xデータ
int   *lyd：左目Yデータ
int   *rxd：右目Xデータ
int   *ryd：右目Yデータ
int   *mhit：マウス押下
```

下位モジュール：
無し

戻り値：
無し

20. file_check

機能：

実験結果ファイル名チェック。

Calling Sequence：

```
int file_check(fname)
```

引数：

char　*fname：実験結果ファイル名

下位モジュール：

無し

戻り値：

正常：　0
異常：－1

21. fname_input

機能：

実験結果ファイル名入力。

Calling Sequence：

```
int fname_input(fname)
```

引数：

```
char    *fname：実験結果ファイル名（O）
```

下位モジュール：

無し

戻り値：

無し

22.freq_input

機能：

周波数入力。

Calling Sequence：

```
freq_input(freq)
```

引数：

```
float *freq：周波数 (O)
```

下位モジュール：

無し

戻り値：

無し

23. init_dflt

機能：

実験データ初期値設定。

Calling Sequence：

```
init_dflt ()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

24. inp_dispno

機能：

表示画像番号の入力をコンソールより行う。

Calling Sequence：

```
int inp_dispno (max,
                xpos,
                ypos)
```

引数：

```
int   max  ：画像総数                (I)
int   xpos：入力要求メッセージ表示位置 (I)
int   ypos：入力要求メッセージ表示位置 (I)
```

下位モジュール：

無し

戻り値：

表示画像番号

25. init_j1_pro

機能：

『半側視野測定実験』パラメータ設定の制御関数。

Calling Sequence：

```
void init_j1_pro()
```

引数：

無し

下位モジュール：

```
menu_j1_sub1
expmode_input
pgsel_input
```

戻り値：

無し

26. init_j2_pro

機能：
『スムースパーシュート実験』パラメータ設定の制御関数。

Calling Sequence：
```
void init_j2_pro()
```

引数：
無し

下位モジュール：
```
menu_j2_sub1
freq_input
width_input
direct_input
select_input
time_input
point_input
plan_input
keisu_input
```

戻り値：
無し

27. init_pro

機能：

実験環境入力制御。

Calling Sequence：

```
void init_pro()
```

引数：

無し

下位モジュール：

```
menu_init
date_input
name_input
dist_input
sample_input
```

戻り値：

無し

28. init_rect

機能：

初期表示画面の編集を行う。

Calling Sequence：

```
init_rect()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

29. inp_ctlky

機能：

リスト表示画面でのキー入力を制御する。

Calling Sequence：

```
int inp_ctlky()
```

引数：

無し

下位モジュール：

無し

戻り値：

入力キー

30. inp_imginf

機能：

表示画像情報の入力制御関数。

Calling Sequence：

```
int inp_imginf (type,
                max,
                gno,
                gmode,
                kepno)
```

引数：

```
int   type   :実験モード (I)
int   max    :画像総数   (I)
int   *gno   :画像番号   (O)
int   *gmode :画面モード (O)
int   *kepno :保存番号   (O)
```

下位モジュール：

無し

戻り値：

無し

31. inp_j1_kekka

機能：
『半側視野測定実験』返答値入力。

Calling Sequence：
inp_j1_kekka(ans)

引数：
int　*ans：返答値　(O)

下位モジュール：
無し

戻り値：
無し

32. inp_memono

機能：

画像保存メモリ指定の入力をコンソールより行う。

Calling Sequence：

```
int inp_memono (xpos,
                ypos)
```

引数：

int　xpos：入力要求メッセージ表示位置（I）
int　ypos：入力要求メッセージ表示位置（I）

下位モジュール：

無し

戻り値：

無し

33. inp_scrdist

機能：

スクロール方向の入力をコンソールより行う。

Calling Sequence：

```
int inp_scrdist (xpos,
                 ypos)
```

引数：

```
int   xpos：入力要求メッセージ表示位置（I）
int   ypos：入力要求メッセージ表示位置（I）
```

下位モジュール：

無し

戻り値：

0：左方向スクロール
1：右方向スクロール

34. j1init

機能：

『半側視野測定実験』のイニシャライズ処理を行う。

Calling Sequence：

```
void j1init()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

35. j2init

機能：

『スムースパーシュート実験』のイニシャライズ処理を行う。

Calling Sequence：

```
void j2init()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

36. jikken_main

機能：

実験処理主制御。

Calling Sequence：

jikken_main()

引数：

無し

下位モジュール：

無し

戻り値：

無し

37. jikken1_pro

機能：

『半側視野測定実験』の制御関数。

Calling Sequence：

```
void jikken1_pro()
```

引数：

無し

下位モジュール：

```
menu_jikken1
control_list
init_j1_pro
menu_j1_sub2
start_jikken1
inp_j1_kekka
kekka_jikken
```

戻り値：

無し

38. jikken2_pro

機能：

『スムースパーシュート実験』の制御関数。

Calling Sequence：

```
void jikken2_pro()
```

引数：

無し

下位モジュール：

```
menu_jikken2
control_list
init_j2_pro
menu_j2_sub2
start_exp2
kekka_jikken
```

戻り値：

無し

39. keisu_input

機能：
係数入力。

Calling Sequence：
keisu_input (data)

引数：
int　*data：実験係数

下位モジュール：
無し

戻り値：
無し

40.kekka_jikken

機能：

実験結果出力制御。

Calling Sequence：

kekka_jikken(type)

引数：

int　type：実験モード　(I)

下位モジュール：

無し

戻り値：

無し

41. list_static

機能：

結果ファイルヘッダ部作成。

Calling Sequence：

ｌｉｓｔ＿ｓｔａｔｉｃ（ｆｐ）

引数：

ＦＩＬＥ　＊ｆｐ：結果ファイルポインタ

下位モジュール：

無し

戻り値：

無し

42. load_pict

機能：

提示画像をロードする。

Calling Sequence：

```
void load_pict(type,
               fname,
               keepno)
```

引数：

```
int  type   :実験モード     (I)
char *fname :画像ファイル名 (I)
int  keepno :保存メモリ番号 (I)
```

下位モジュール：

無し

戻り値：

無し

43. main

機能：

提示画像表示制御機能のメイン関数。

Calling Sequence：

main ()

引数：

無し

下位モジュール：

無し

戻り値：

無し

44. menu_init

機能：
環境値設定メニュー表示。

Calling Sequence：
```
int menu_init()
```

引数：
無し

下位モジュール：
無し

戻り値：
処理番号

45. menu_j1_sub1

機能：

『半側視野測定実験』パラメータ設定画面表示。

Calling Sequence：

```
int menu_j1_sub1(err)
```

引数：

int　err：エラー番号（I）

下位モジュール：

無し

戻り値：

無し

46. menu_j1_sub2

機能：

『半側視野測定実験』起動画面表示。

Calling Sequence：

```
int menu_j1_sub2()
```

引数：

無し

下位モジュール：

無し

戻り値：

実験開始：'Y'
実験中止：'N'

47. menu_j2_sub1

機能：

『スムースパーシュート実験』パラメータ設定画面表示。

Calling Sequence：

```
int menu_j2_sub1(err)
```

引数：

int　err：エラー番号

下位モジュール：

無し

戻り値：

処理番号

48. menu_j2_sub2

機能：

『スムースパーシュート実験』起動画面表示。

Calling Sequence：

```
int menu_j2_sub2 ()
```

引数：

無し

下位モジュール：

無し

戻り値：

実験開始：'Y'
実験中止：'N'

49.menu_jikken1

機能：

『半側視野測定実験』メニュー画面表示。

Calling Sequence：

```
int menu_jikken1(err)
```

引数：

int　err：エラー番号　(I)

下位モジュール：

無し

戻り値：

処理番号

50.menu_jikken2

機能：

『スムースパーシュート実験』メニュー画面表示。

Calling Sequence：

```
int menu_jikken2(err)
```

引数：

int　err：エラー番号　(I)

下位モジュール：

無し

戻り値：

無し

51.menu_list

機能：
リスト画面制御部表示。

Calling Sequence：
menu_list(type)

引数：
int　type：実験モード

下位モジュール：
無し

戻り値：
無し

52. menu_main

機能：

メインメニュー表示。

Calling Sequence：

```
int menu_main()
```

引数：

無し

下位モジュール：

無し

戻り値：

処理番号

53. mk_j1_result

機能：
『半側視野測定実験』結果ファイル作成。

Calling Sequence：
mk_j1_result()

引数：
無し

下位モジュール：
無し

戻り値：
無し

54. mk_j2_result

機能：
『スムースパーシュート実験』結果ファイル作成。

Calling Sequence：
mk_j2_result()

引数：
無し

下位モジュール：
無し

戻り値：
無し

55. mode_input

機能：

実験日入力。

Calling Sequence：

```
void mode_input(int)
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

56. name_input

機能：

被験者名をコンソールより入力する。

Calling Sequence：

```
void name_input(name)
```

引数：

char　*name：被験者名　(O)

下位モジュール：

無し

戻り値：

無し

57. nonscr

機能：

ノンスクロール画面表示。

Calling Sequence：

nonscr (keepno)

引数：

int　keepno：保存番号 (I)

下位モジュール：

無し

戻り値：

無し

58. pgsel_input

機能：

保存されている実験画像の中から提示画像を選択する。

Calling Sequence：

```
int pgsel_input (gname,
                 gmode,
                 keepno,
                 subject)
```

引数：

char	*gname：画像ファイル名	(O)
int	*gmode：画像モード	(O)
int	*keepno：保存番号	(O)
int	subject：対象眼球	(O)

下位モジュール：

無し

戻り値：

無し

59. plan_input

機能：

実験手順の入力。

Calling Sequence：

plan_input (data)

引数：

int　*data：実験手順　(O)

O：ｱﾀﾞﾌﾟﾃｰｼｮﾝのみ

1：測定のみ

2：ｱﾀﾞﾌﾟﾃｰｼｮﾝ&測定

下位モジュール：

無し

戻り値：

無し

60. point_input

機能：

画像表示位置入力。

Calling Sequence：

```
point_input (xpos,
             ypos)
```

引数：

```
int    *xpos：入力要求メッセージ表示位置（I）
int    *ypos：入力要求メッセージ表示位置（I）
```

下位モジュール：

無し

戻り値：

無し

61.psadow

機能：

『半側視野測定実験』画像の編集を行う。

Calling Sequence：

```
pshadow()
```

引数：

無し

下位モジュール：

```
ana_memory
init_rect
```

戻り値：

無し

62. req_screen2

機能：

画像番号入力。

Calling Sequence：

```
int req_screen(data)
```

引数：

int　*data：画像総数

下位モジュール：

無し

戻り値：

無し

63. sample_input

機能：
サンプリング時間をコンソールより入力する。

Calling Sequence：
```
void sample_input(sample)
```

引数：
```
int    *sample：サンプリング時間（O）
```

下位モジュール：
無し

戻り値：
無し

64. scrinit

機能：

助既画面表示。

Calling Sequence：

```
scrinit(keepno)
```

引数：

int　keepno：保存番号（O）

下位モジュール：

無し

戻り値：

無し

65. scrscr

機能：

スクロール制御。

Calling Sequence：

```
scrscr(scno,
       page)
```

引数：

int　scno：画面番号
int　page：メモリページ番号

下位モジュール：

無し

戻り値：

無し

66. start_exp2

機能：

『スムースパーシュート実験』の開始制御関数。

Calling Sequence：

```
start_exp2 (flg,
            ptx,
            pty,
            dtx,
            dty)
```

引数：

int	flg	：眼球データ反映フラグ
int	*ptx	：眼球データX座標
int	*pty	：眼球データY座標
int	*dtx	：画像表示位置
int	*dty	：画像表示位置

下位モジュール：

無し

戻り値：

無し

67. start_jikken1

機能：

『半側視野測定実験』制御関数。

Calling Sequence：

```
void start_jikken1 ()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

68. swindow

機能：

提示画像表示スクリーンに６４０×４８０のマスクをかける。

Calling Sequence：

```
void swindow()
```

引数：

無し

下位モジュール：

無し

戻り値：

無し

69. time_input

機能：
実験時間入力。

Calling Sequence：
time_input (data)

引数：
int　*data：実験時間　(O)

下位モジュール：
無し

戻り値：
無し

70.width_input

機能：

振幅入力。

Calling Sequence：

```
width_input (data)
```

引数：

```
int    *data：振幅   (O)
```

下位モジュール：

無し

戻り値：

無し

8．システム構成補足

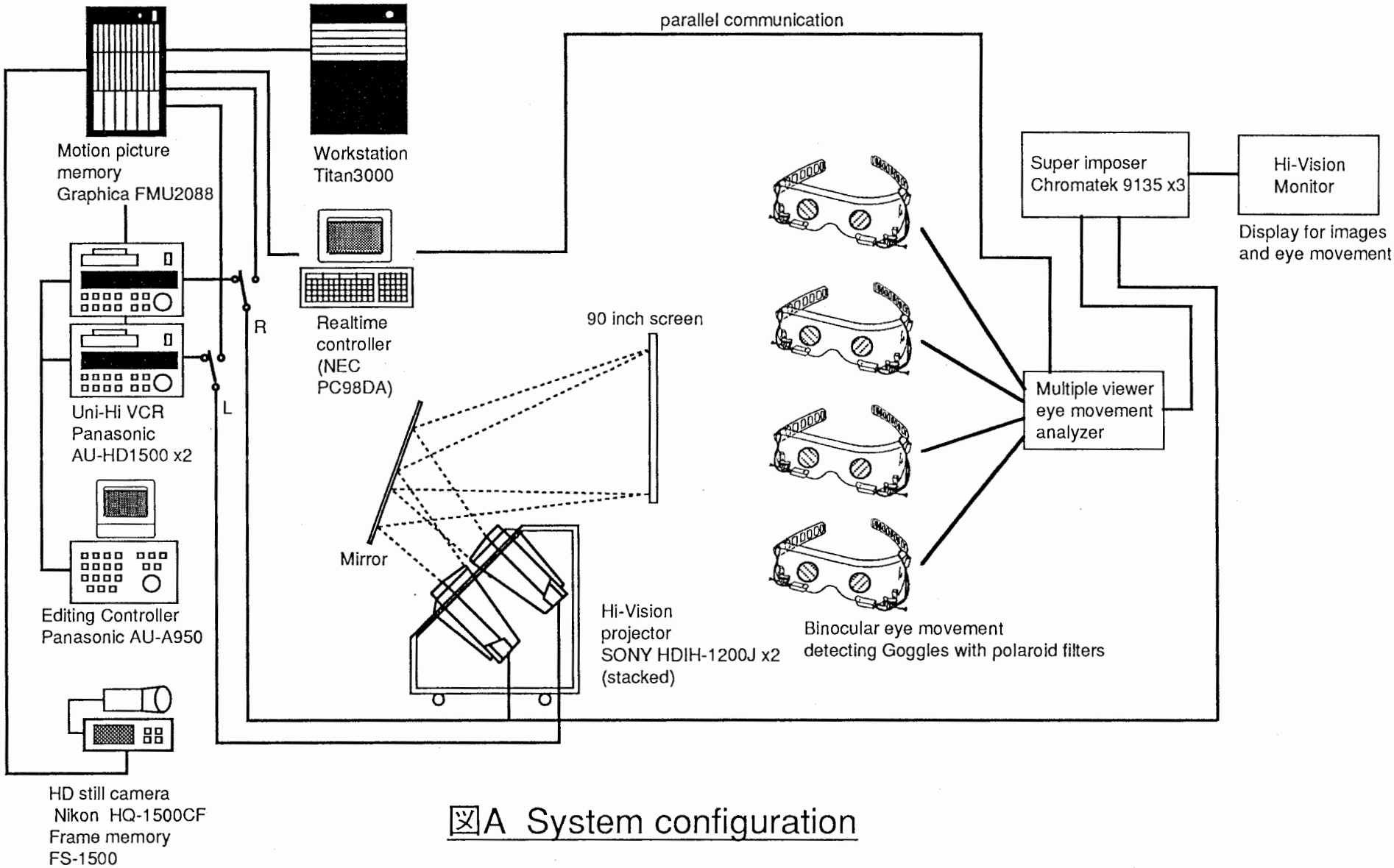
図Aに構成を示す。

Hi-Vision立体画像のソースは、Workstation(Titan3000)、Uni-Hi VCR(Panasonic AU-HD1500x2)+Editing Controller(Panasonic AU-A950)、HD Still Camera(Nikon HQ-1500CF)、動画像メモリ(Graphica FMU2088)により生成される。一般的には、WorkStationではＣＧによる静止画の生成、動画像メモリでは短時間（ステレオ画像モードで２４フレーム）の動画の生成、ＶＣＲでは長時間の動画の生成に用いる。

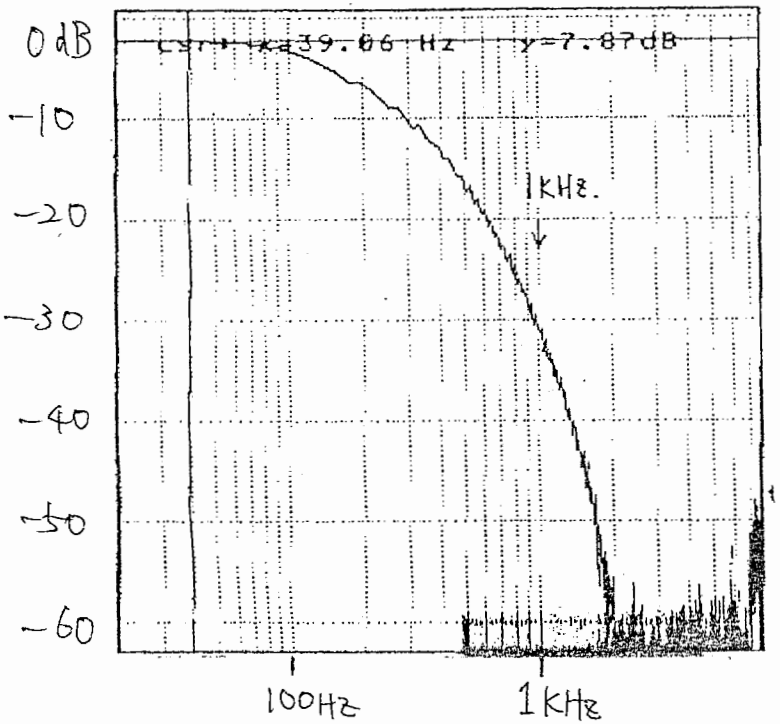
Hi-Vision立体画像提示は９０インチ背面投射型のＨＤプロジェクタ（SONY HD1H-1200J）を用い、プロジェクタ前面に設置された直線偏光フィルタと、眼球運動測定ゴーグルに取付けられた直線偏光フィルタにより実現される。通常の２Ｄ画像提示の場合は、片方のチャンネルのみ表示する。

眼球運動測定装置（竹井機器製）は、同時に４人まで両眼眼球運動を測定可能である。サンプリング周波数は最大２００Ｈｚであり、アナログ帯域は15,30,100,1kHzを選択できる（アナログLPFは、ＳＣＦ(スイッチト・キャパシタ・フィルタ)を用いた構成になっており４次バタワース特性(-24dB / oct)になっている。また、総合的な伝達特性としては、切り替え可能なこのＬＰＦ(最終段)の前段の信号処理系の伝達特性が関与する。検出用のＰＤから切り替え可能なこのＬＰＦの入力までの振幅・位相特性を図Ｂに示す。全体の伝達特性は、この特性＋４次バタワース特性となる。）。眼球運動測定装置は、測定モードにおいてparallel 通信によりリアルタイム実験制御用のパーソナルコンピュータ（ＰＣ９８０１DA）に校正後の眼球運動データを送信する。これを受けて、リアルタイム実験制御用のパーソナルコンピュータは実験を制御する。データの送受信は１人当たり数百μsecで終了する。このため、眼球運動測定結果を提示画像にリアルタイムで反映することが出来る。但し、現在はGraphicaの動画像メモリの垂直SYNCをリアルタイム実験制御用のパーソナルコンピュータに入力していないので、両者は非同期で動作している。リアルタイム実験制御用のパーソナルコンピュータからの命令はGraphica側ではFIFOに入る。その後、Graphica内のCPUにより順次命令が実行されるが、実際に表示画面が変化するのは、垂直SYNCのタイミングからである。垂直SYNCまでに実行された命令の内、実行が終了した最も新しい命令が有効となる。

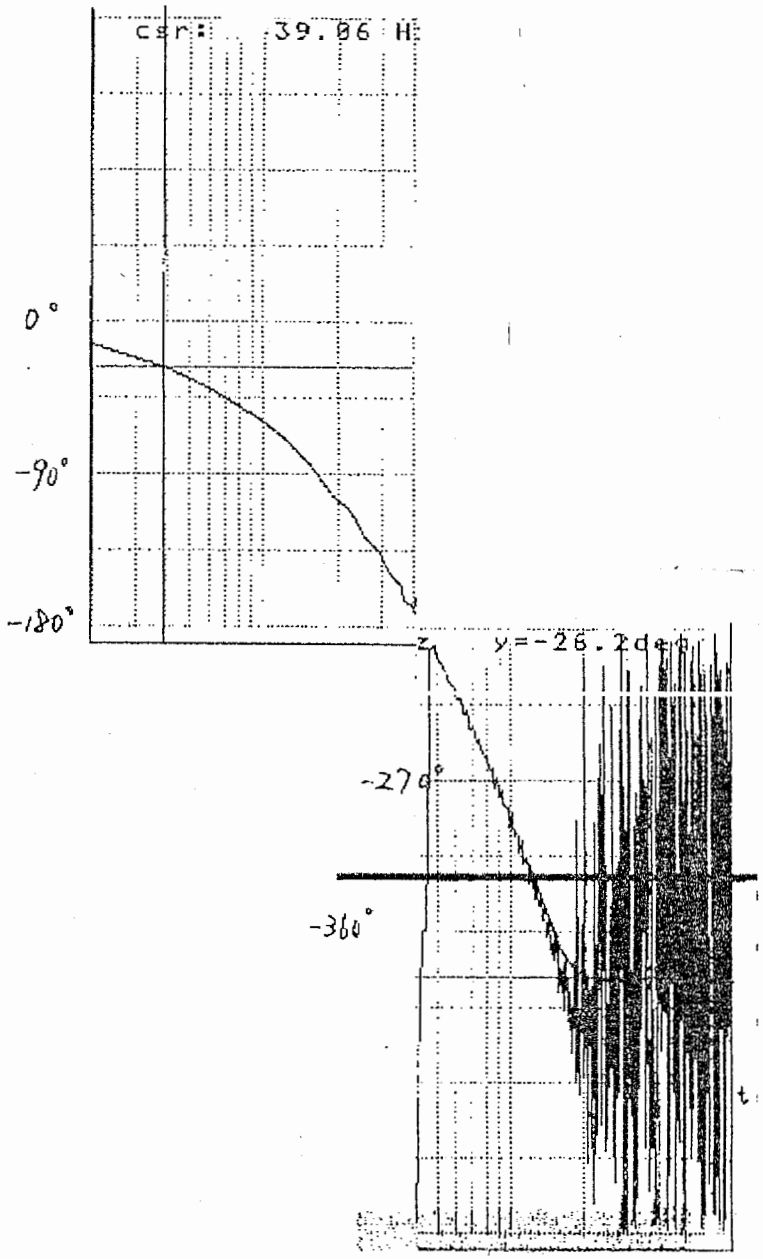
測定された眼球運動は、各被験者の視線の位置として２次元的に表示される。これと、提示画像はHi-Visionスーパーインポーザにより合成され、Hi-Vision モニタに表示される。この時、Chromatec製の信号処理装置により、スーパーインポーズする画像はＲch, Ｌch, Ｒ＋Ｌch, Ｒ-Ｌchを選択できる。



図A System configuration



(a) 振幅特性



(b) 位相特性

図B. PD(フォトダイオード)入力からアナログLPF入力までの伝達特性