

FZ-1 external program manual ver 1.0

+++++

FZ-1 static parameter definition

+++++

LOCATION	OBJECT	CODE	LINE	SOURCE	LINE
			1	"70116"---	means as above.
			2	DATA	
address	3	ORG	0400H		
0400	4	keycount	DBS	1 ;	numbers of key_on
0401	5	lastresp	DBS	1 ;	the last touch response
vaule	1~127				
0402	6	keymap	DBS	8 ;	key on/off table
040A	7	sch	DBS	2 ;	big timer counter
040C	8	olddca	DBS	16 ;	generater data
041C	9	newdca	DBS	16 ;	---
042C	10	key	DBS	1 ;	console key code
042D	11	kkk	DBS	1 ;	---
042E	12	kls	DBS	1 ;	---
042F	13	sls	DBS	1 ;	---
0430	14	ki0	DBS	4 ;	---
0434	15	ki1	DBS	4 ;	---
0438	16	rpc	DBS	2 ;	repeat counter for console
key					
043A	17	adc1	DBS	8 ;	adc() static
0442	18	adcb1	DBS	2 ;	---
0444	19	env	DBS	1 ;	ad convert vaule of line or
mic in					
0445	20	vol	DBS	1 ;	ad convert value of entry
volume					
0446	21	old	DBS	8 ;	last ad convert value
044E	22	max	DBS	8 ;	max ad value
0456	23	min	DBS	8 ;	min ad value
045E	24	cenh	DBS	1 ;	center high limit for bender
045F	25	cenl	DBS	1 ;	center low limit for bender
0460	26	stat	DBS	3 ;	midi status byte
0463	27	par1	DBS	3 ;	midi first data byte
0466	28	nownote	DBS	2 ;	last MIDI key code and res-
ponse					
0468	29	genbit	DBS	2 ;	generater bit:num
046A	30	lastiy	DBS	2 ;	last generater pointer
046C	31	excn	DBS	1 ;	exclusive midi data counter
046D	32	nowled	DBS	1 ;	now led
046E	33	rand	DBS	2 ;	random number for lfo gene-
rate					
0470	34	jump0	DBS	2 ;	con()'s static
0472	35	lev	DBS	1 ;	---
0473	36	lv0	DBS	3 ;	---

0476	37	dm	DBS	1 ; ---
0477	38	dm0	DBS	3 ; ---
047A	39	sm	DBS	1 ; ---
047B	40	sm0	DBS	3 ; ---
047E	41	mm	DBS	1 ; ---
047F	42	mm0	DBS	3 ; ---
0482	43	lpos	DBS	2 ; para_change() parameter
0484	44	cpos	DBS	2 ; ---
0486	45	lmax	DBS	2 ; ---
0488	46	cmax	DBS	2 ; ---
048A	47	loff	DBS	2 ; ---
048C	48	ltop	DBS	2 ; ---
048E	49	vpos	DBS	2 ; ---
0490	50	posv	DBS	2 ; graph() parameter
0492	51	posp	DBS	2 ; ---
0494	52	vhi	DBS	2 ; ---
0496	53	wid	DBS	4 ; ---
049A	54	pos	DBS	16 ; ---
04AA	55	grast	DBS	4 ; ---
04AE	56	graed	DBS	4 ; ---
04B2	57	pplst	DBS	4 ; ---
04B6	58	ppled	DBS	4 ; ---
04BA	59	pp2st	DBS	4 ; ---
04BE	60	pp2ed	DBS	4 ; ---
04C2	61	mcount	DBS	2 ; meter() or jobbing() static
04C4	62	mlevel	DBS	2 ; ---
04C6	63	mpeek	DBS	2 ; ---
04C8	64	mtrig	DBS	2 ; ---
04CA	65	bb0	DBS	1 ; brink() static
04CB	66	bb1	DBS	1 ; ---
04CC	67	l_pos	DBS	4 ; d_graph() static
04D0	68	l_cur	DBS	2 ; ---
04D2	69	l_vhi	DBS	2 ; ---
04D4	70	l_brk	DBS	2 ; ---
04D6	71	trig	DBS	1 ; recording trig level (0~255)
04D7	72	rmod	DBS	1 ; recording mode
04D8	73	gain	DBS	1 ; recording gain 0=L 1=H
04D9	74	rate	DBS	1 ; sampling rate
(0:36kHz,1:18kHz,2:9kHz)				
04DA	75	length	DBS	2 ; recording length (10msec)
04DC	76	sintable	DBS	48 ; sin table for sin_synthesis
050C	77	add_v1	DBS	1 ; source 1 voice # (0~63)
050D	78	add_v2	DBS	1 ; source 2 voice # (0~63)
050E	79	add_l1	DBS	1 ; source 1 mix level (0~255)
050F	80	add_l2	DBS	1 ; source 2 mix level (0~255)
0510	81	add_t1	DBS	1 ; source 1 detune (-127~127)
0511	82	add_t2	DBS	1 ; source 2 detune (-127~127)
0512	83	add_dly	DBS	4 ; source 2 delay WORD address
0516	84	add_xs1	DBS	4 ; xmix start WORD address
051A	85	add_xs2	DBS	4 ; xmix end WORD address

051E	86	devnum	DBS	2 ; device number
(0:FDD,1:MIDI,2:PORT)				
0520	87	restat	DBS	2 ; remote() static
0522	88	remode	DBS	2 ; ---
0524	89	cat	DBS	160 ; cluster allocation table
05C4	90	dloc	DBS	2 ; disk location counter
05C6	91	xysheet	DBS	768 ; lcd graphic dot image
08C6	92	voice_num	DBS	2 ; disk subroutine's static
08C8	93	bank_num	DBS	2 ; ---
08CA	94	wave_num	DBS	2 ; ---
08CC	95	cnv_sta	DBS	2 ; ---
08CE	96	cnv_pos	DBS	2 ; ---
08D0	97	cnv_rp	DBS	4 ; ---
08D4	98	memsize	DBS	2 ; wave memory size (*64Kbyte)
08D6	99	mi	DBS	260 ; midi input ring buffer
09DA	100	mo	DBS	260 ; midi output ring buufer
0ADE	101	kb	DBS	260 ; keyboard ring buffer
0BE2	102	si	DBS	260 ; sequencer input ring buffer
0CE6	103	so	DBS	260 ; sequencer output ring buffer
0DEA	104	midirev	DBS	1 ; midi recieve channel
0DEB	105	midisnd	DBS	1 ; midi send channel
0DEC	106	midimsk	DBS	1 ; midi mask status
0DED	107	midiprg	DBS	1 ; midi program change resister
(-1:MASK)				
0DEE	108	seq	DBS	2 ; sequencer status
0DF0	109	godtime	DBS	4 ; god time for sequencer
0DF4	110	oldtime	DBS	4 ; old time for sequencer
0DF8	111	tempo	DBS	2 ; tempo counter
0DFA	112	mastertune	DBS	2 ; master tune (1/256 sub tone)
0DFC	113	pbn	DBS	2 ; play bank number (0-7)
0DFE	114	pb	DBS	4 ; &bank[pbn]
0E02	115	evn	DBS	2 ; edit voice number (0-63)
0E04	116	ev	DBS	4 ; &voic[evn]
0E08	117	bank	DBS	5248 ; struct bankdata bank[8]
2288	118	voic	DBS	12288 ; struct voicdata voic[64]
5288	119	pare	DBS	24 ; struct paradata para
52A0	120	nowe	DBS	384 ; run time paradata nowe[16]
5420	121	gene	DBS	464 ; run time generater data
55F0	122	psa	DBS	2 ; rom entry static
55F2	123	pca	DBS	2 ; ---
55F4	124	pwa	DBS	2 ; ---

.PA

+++++

Lowest console functions group

+++++

/***** FUNCTION No. 0 *****/
 name : entry

```

function      : all system initilize
usage         : entry();

```

Resets all FZ-1 system like when power on. All parameters and

wavedata may be erased. So does not return from BRK3 to an external

program.

```

/***** FUNCTION No. 6 *****/
name      : mgetc
func      : get char now
usage     : c=mgetc();
           int c; c=ERROR no key
/***** FUNCTION No. 7 *****/
name      : ungetc
func      : back get char now
usage     : ungetc( c );
           int c; return key code

```

The mgetc function gets a character of console switches. The character

code are define as follow. The ungetc function does push back a

character to keycode buffer.

```

/* ----- consol switches define ----- */
ZERO      0~9      /* tenkey 0-9      */
INC        10      /* increment value */
DEC        11      /* decrement      */

UP         16      /* move up cursor */
DOWN       17      /*      down      */
RIGHT      18      /*      right     */
LEFT       19      /*      left      */
ENT        20      /* entry menu     */
ESC        21      /* escape         */
DISP       22      /* display mode change */
PLAY       24      /* play mode      */
MOD        25      /* parameter modify */
SETMENU    26      /* set/menu switch */
TRANS      27      /* transpose      */
TUNE       28      /* tunning        */

```

```

/***** FUNCTION No. 8 *****/
name      : contsw

```

```

func      : check continue push switch
usage     : push = contsw( c );
            int push; OK(0):continus push
            int c;    check character

```

Checks a switch whether on or off.

```

/***** FUNCTION No. 10 *****/
name      : main volume
func      : read main entrya volume
usage     : v = mvol();
            int v;  if v = ERROR, not change

```

Reads a data_entry volume data. The ERROR (-1) value means that this

volume data is not changed after the last reading call of mvol().

```

/***** FUNCTION No. 11 *****/
name      : envelop value
func      : read envelop value
usage     : v = evol();
            int v;  0~127

```

Reads a analog-digital converter value of the line or mic signal for

a sampling recording.

.PA

```

+++++
+++++

```

Generater intialize functions group

```

+++++
+++++

```

```

/***** FUNCTION No. 20 *****/
name      : all note off
func      : note off by select dev
usage     : all_noteoff(i);
            int i;  (0=key, 1=MIDI, 2=seq)

```

Forces to stop the sound generation which may be generated by keyboard,

MIDI or sequencer,independaly.

```

/***** FUNCTION No. 21 *****/
name      : all midi chn
func      : midi off by select dev
usage     : all_midichn( chan );

```

```

int chan; new channel
    if new == old, send key off only

```

Changes the MIDI basic channel number of the FZ-1. This function may

be called from the function 'EFFECT/MIDI'. So it is not usual calling

for external programs.

```

/***** FUNCTION No. 22 *****/
name      : control_on
func      : send now control value to MIDI out
usage     : control_on();

```

Sends all control value to MIDI out. Bender, after touch, modulation

wheel, foot volume and sustain switch are send.

```

/***** FUNCTION No. 23 *****/
name      : control_off
func      : control initialize to MIDI out
usage     : control_off();

```

Sends all default control value to MIDI out for clear. This function

may be called for changing MIDI control mask condition. This default

values are defined as follows.

```

bender      ... center value
after       ... zero
modu W.     ... zero
foot vol    ... max
sustain sw  ... off

```

.pa

```

/***** FUNCTION No. 26 *****/
name      : note key code
func      : read entry volume
usage     : v = ngetc();
            int v; if v = ERROR, not change
                  else v = TOUCH:KEY_CODE

```

Gets the last MIDI note and touch code. The upper byte means the

initial touch value, and the lower one means the key code

which are

depend on MIDI format. A ERROR(-1) code means that no key code is

exist after the last call.

```

/***** FUNCTION No. 42 *****/
name      : gene_off
func      : generater sound off
usage     : gene_off( g )
              int g; generator number (0~7)

```

The FZ-1 has 8 generater channels. This function does initialize a

generater which is selected by parameter 'g'.

```

/***** FUNCTION No. 43 *****/
name      : gabinit
func      : gate array initialize do
usage     : gabint();

```

Initializes all generater channel. It is useful when all sound should

be cut off at once.

.PA

+++++

Basic PCM recording functions group

+++++

```

/***** FUNCTION No. 45 *****/
name      : rec_start
func      : send record start command to gaa
usage     : rec_start(vn,pre);
              struct voicedata *vn;    record voice
              unsigned int pre;        pre record

```

length

Sends some record_starting commands to the sound generater for pcm

recording. MUST set paramters as follows(see voice struct).

```

vn-wavst ... pcm recording wave data start address(WORD)
vn-waved ...                               end
rate      ... sampling frequncy (0:36KHz,1:18kHz,2:9kHz)

```

The pre_recording is stated after this call. The pre parameter ,which

means the pre_recording length, MUST be shorter than (ev->waved - ev-

>wavst).

```

/***** FUNCTION No. 46 *****/
name      : rec_trig
func      : start post recording by line 1
usage     : rec_trig();

```

Terminates the pre_recording, and starts the main_recording.

```

/***** FUNCTION No. 47 *****/
name      : rec_stop
func      : recording stop
usage     : rec_stop(vn,pre);
            struct voice data *vn;  record voice #
            unsigned int pre;      pre word length

```

Terminates the main_recording, and arranges the wave memory for

the pre_recorded data. The same parameter 'pre' should be set when

rec_start() was called.

```

/***** FUNCTION No. 48 *****/
name      : set_gain
func      : set recording gain
usage     : set_gain( g );
            g = 0 for 0dB, =1 for -20dB

```

Sets the recording gain for pcm recording.

.pa

```

/***** FUNCTION No. 49 *****/
name      : now status
func      : read now generator status byte
usage     : gstat = now_st( i );
            int gstat; generator status
            int i;      generator number (0-7)

```

Reads a generator status which is defined as follows.

```

gstat :  ** **MZ RELP
        || |||> play on ( zero means the end of
record)
        || ||> loop on

```



```

|| |> end on
|| |> reverse on
|> output zero on
+> multi loop

```

.pa

```

+++++
MIDI ringbuffer handling functions group
+++++

```

```

/***** FUNCTION No. 51 *****/

```

```

name      : multiplex

```

```

function: write key, midi, (seq) buffer

```

```

usage     : mpx(d1,d2,d3);

```

```

           int d1,d2,d3; send data

```

```

           if d2's MSB == 1 then no send

```

MIDI

```

           if d3==0xFF          then 2 byte code

```

The FZ1 has 3 ring_buffer for output, which are located at
ki,mo and

si. This function sets some MIDI data into these 3 ringbuffers.
The ki

means keyboard input buffer, mo does MIDI out ringbuffer and si
does

sequencer ringbuffer.

For instance, When push a musical keyboard like 'C4', this
function

will be called as 'mpx(90,40,7F)'. These data format are based on
MIDI

data. And when push a number key in play- mode, this function
will be

called as 'mpx(C0,01,FF)'. This program_change data has no second
data

byte. So set send data d3 as FF hex.

```

/***** FUNCTION No. 52 *****/

```

```

name      : key in

```

```

func      : read data from ringbuffer

```

```

usage     : data = keyin(kn);

```

```

                                int data;          read data, ERROR then no
data
                                int kn;             key number 0=KEY, 1=MIDI,
2=SEQ

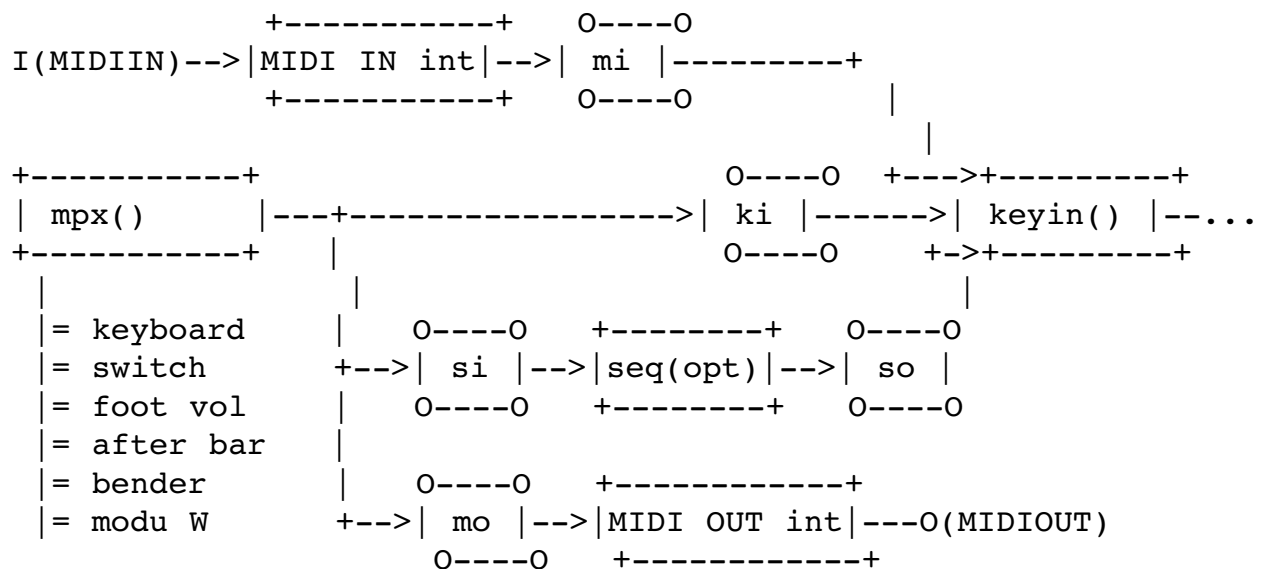
```

FZ-1 has 3 ring_buffer for input, which are ki,mi and so.
This

function gets some MIDI data from these 3 ringbuffers. A
ERROR(-1)

returns means that this ringbuffer is empty.

This is a chart for ringbuffers and functions as follows



.PA

```

+++++
+++++

```

Users console handling functions group

```

+++++
+++++

```

```

/***** FUNCTION No. 65 *****/

```

```

name      para_change
func      all parameter change operation
usage     : vl = para_change( ppp );
           int vl;mgetc() character
           ERROR-1(-2) for exit

```

```

struct pppdata ppp[];    para data

```

```

/* ----- para_change()'s static parameter ----- */
int lpos;          /* line      pos          (init: 0 ) */
int cpos;          /* column pos      (init: 0 ) */
int lmax;          /* line max limit  (init: ? ) */

```

```

int cmax;          /* coumn max limit      (init: ? ) */
int loff;          /* line offset        (init: ?<0)*/
int ltop;          /* line top limit     (init: 0 ) */
int vpos;          /* coumn parameter pos (init: 0 ) */

struct change {
    int max;        /* value max          */
    int min;        /* value min          */
    short mm;       /* mode ( see bellow ) */
    short cc;       /* char colmn pos (0~15)*/
    short cl;       /* char line pos  (1~?) */
    short cn;       /* char number      (0~15)*/
    char name[16];  /* char text for print */
                  /* '@' for cpos print */
    char sw[];      /* switch name        */
};

struct pppdata {
    int v;
    struct change *p;
};

----- change mode define -----
bit 7      : ( 1:horizun menu, 0:vertical menu)
bit 6      : ( 1:meter on      , 0:meter off      )
bit 5      : ( 1:volume off    , 0:volume on      )
bit 4      : ( 1:with voice name , 0:normal      )
bit 3      : undefined
-----
          : value switch          ex.
bit 2~0    : 0 : value menu        <2000          >
          1 : midi note code      <#C4          >
          2 : switch               <ON OFF        >
          3 : function menu        [copy from DCQ ]
          4 : value/switch         < ON ~ 01-99 ~ OFF >
          5 : bit set              < 0..00000 >
          6 : transpose            <C,#C ~ B >
          7 : title <skip> must be after (0-6)
-----
*/

```

This function is very useful but complicated one. All parameter

editor calls it. When first calling, posv,loff,lpos,cpos and vpos must

be set as zero. And the lmax means the number of parameter, and the

ltop means the offset line number for lcd screen. For instance,

para_chenge() dose not use line 0~5 when ltop is 6. Set a minus value

of ltop when first calling for intialize. Show the most simple example

as follows.

```
simple()
{
    struct paradata p[1];

    /* initalize for para_change */
    posv = loff = lpos = cpos = vpos = 0;
    ltop = -1; lmax = 1;

    /* struct paradata set (\ means 'back slash' in Japan) */
    p[0].p = "\177\000\201\377\000\015\002\003DCA LEVEL =+***";
    p[0].v = parameter;

    /* main loop */
    for(;;) {
        if( para_change( p )==(ERROR-1) ) break;
        parameter = p[0].v;
    }
}
```

And show the example for paradata initalizing as bellow, which is

used as 'BANK EDIT/CREAT BANK'.

```
p[0].p = "\100\000\001\000\020\016\002\002VOICE No. = **";
p[1].p = "\177\000\000\000\001\014\004\003ORIGINAL =*****";
p[2].p = "\177\000\000\000\001\014\005\003HIGHEST =*****";
p[3].p = "\177\000\000\000\001\014\006\003LOWEST =*****";
p[4].p = "\177\000\001\000\000\015\007\003MAX TOUCH = ***";
p[5].p = "\177\000\001\000\000\015\010\003MIN TOUCH = ***";
p[6].p = "\177\000\000\000\000\015\011\003AREA LEVEL= ***";
p[7].p = "\020\000\001\000\000\016\012\002MIDI CH = **";
p[8].p = "\377\000\000\000\045\010\013\000OUTPUT OOOOOOOO";
.pa
```

```
/****** FUNCTION No. 66 *****/
name      ; meter
func      ; sumilate audio level meter
usage     ; meter( v );
                int v; (ERROR=open meter, else value )
```

Displays the level meter, which is used in the pcm recording fun

ctions. At the first time, set parameter v as ERROR(-1) for

initializing meter. And set parameter as meter value. So the most

simple example is as bellow.

```
meter(ERROR);  
for(;;) meter( evol() );
```

```
/****** FUNCTION No. 69 *****/  
name      : tenkey  
func      : tenkey data input subroutine  
usage     : v = tenkey(v,para);  
           int v;           old & new value  
           struct change   tenkey paramter paket;  
remark    : para may be ROM data ! DONOT write
```

It operates for parameter editting. It is usually called by the

para_change(). All tenkey and enter volume are used for this function.

```
/****** FUNCTION No. 70 *****/  
name      : bitkey  
func      : bitport set 000*000*  
usage     : bitkey( l, v );  
           int l; line position  
           short *v; old value  
remark    : column posistion is 8.
```

This function operates for the bit parameter like the output-bit in

'BANK EDIT/CREAT BANK'. Called by the para_change().

```
/****** FUNCTION No. 71 *****/  
name      : ascii  
func      : ascii char data input subroutine  
usage     : ascii(c,l,s,n);  
           int c;           colmn position  
           int l;           line position  
           int s;           character length  
           char *n;         character name
```

This function operates for character parameter like voice name or bank

name.

.pa

/***** FUNCTION No. 73 *****/

```
name      : d_change
func      : display change parameter with value,key or
switch
usage     : d_change( mode,v,para );
              int    mode; (0:normal, 1:reverse)
              int    v;    display value
              struct change *para;
```

Displays a struct para data. It may be used for rewrite a line.
The

most simple example is showd as folows. All para_change()'s para-
meters

must be defined before this calling.

```
diplay_parameter()
{
  char *p;

  p = "\100\000\001\000\020\016\002\002VOICE No. =  **";
  d_change( 0,10,p );
}
```

/***** FUNCTION No. 74 *****/

```
name      : d_change_all
func      : display change with offset dl
usage     : d_change_all( mode,ppp );
              int mode;
              OK (0)    = all character
              ERROR(-1)= only value
              struct pppdata ppp[];
```

Displays all parameters by struct pppdata. It may be used for
rewri

ting the all display.

/***** FUNCTION No. 75 *****/

```
name : brink
func : brink timer counter
usage: b = brink();
              int b;
              ERROR -- no change mode
              0     -- normal write
              1     -- reverse write;
```

Gets brink parameter. It is controled timer interapt operation.
For

instance, The small program is showed ,which prints a character
"*" at

line0 and col0 with brinkning.

```

for(;;){
    if( (b=brink())!=ERROR )
        print ( 0,0,b,"*" );
}
.pa
/***** FUNCTION No. 76 *****/
name      : graph
func      : graphic edit wave point
usage     :  c = graph( mode,ppp1,ppp2 );
              int c;                mgetc character
                                      ERROR-1 for
exit
              int mode;             0 = only display, nocur-
sol
                                      1 = ppp1 set
                                      2 = ppp1 & ppp2
              long *ppp1;           cursol pos ,v1 offset
(WORD.)
              long *ppp2;           cursol pos ,v2 offset
(WORD.)
    use static :
/* ----- graph()'s static parameters ----- */
int  posv;           /* graphic items number
*/
/*                  b15   : 0=new wave
*/
/*                  b14   : 0=new disp
*/
/*                  b0    : 0=1st,1=2nd
*/
int  posp;           /* graphic level
*/
int  vhi;            /* verchcal shift
*/
long wid;            /* graphic width
*/
long pos[4];         /* graphic postions
*/
long grast,graed;    /* graphic window limit
*/
long pplst,ppled;    /* ppp1 value      limit
*/

```

```
long pp2st,pp2ed;                                /* ppp2 value      limit
*/
```

This function has very useful but complicated for the wave graphic

display. All FZ-1's parameter editing function like as truncate, loop

address positioning, uses this function .

Must set parameters for the first call as follow.

```
posv  .. set zero for initialize graph().
grast .. set display wave data start address (WORD)
graed ..                                     end
pplst .. cursol address low limit for ppp1   (WORD)
ppled ..                                     high
pp2st .. cursol address low limit for ppp2   (WORD)
pp2ed ..                                     high
```

For instance :

```
graph_simple()
```

```
{
```

```
    long ppp1,ppp2;
```

```
    posv  = 0;
```

```
    ppp1   = grast = pplst = ev->wavst;
```

```
    ppp2   = graed = pp2ed = ev->waved;
```

```
    ppled  = ev->gened;
```

```
    pp2st  = ev->genst;
```

```
/* main loop */
```

```
    for(;;) {
```

```
        if( graph(2,&ppp1,&ppp2 )==(ERROR-1) ) break;
```

```
    }
```

```
}
```

The parameter 'mode' means the number of the setting address. If this

mode is 2, the function graph() can edit two address parameter like

the generator start and end. All user operation is the same one like

the menu 'VOICE EDIT/CREATE VOICE/TRUNCATE'. If this mode is 1, It can

edit one address like the delay time in the menu 'DELAY TIME'.


```

/***** FUNCTION No. 77 *****/

```

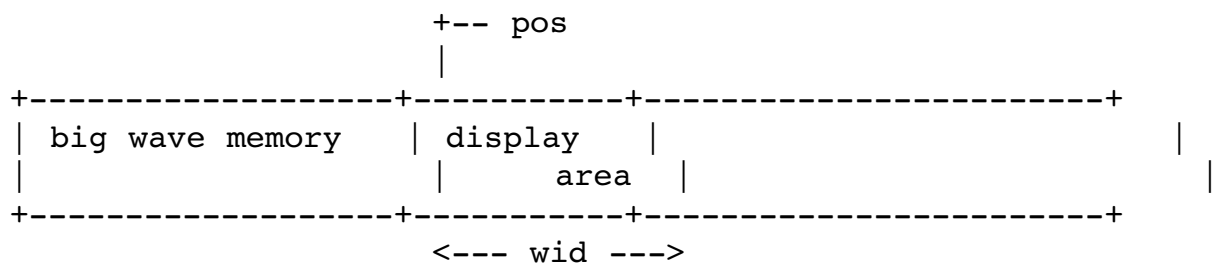
```

    name      : draw for graph subroutine
    usage     : c = d_graph( mode,pos,wid,cur,vhi );
                  int c;          cursor changed=ERROR,
else OK
                  int mode;      OK=full, ERROR=if chan-
ged.
                  long pos;      wave start address
(WORD.)
                  long wid;      display width
(WORD.)
                  int cur;       cursor pos
(0~95)
                  int vhi;       vertical close up
(0~7)

```

This function displays the wavedata with cursol. This function is

usefull for only displaying pcm wavedata into lcd unit.



```

/***** FUNCTION No. 79 *****/

```

```

    name      : egraph
    func      : envelop data graphic subroutine
    usage     : c = egraph( p,e );
                  struct paradata p[];
                  struct envelop *e;
                  int c;          return char

```

Calls by envelop() which is function no.91.

.pa

```

/***** FUNCTION No. 80 *****/

```

```

    name      : print voice number with name & status
    func      ;
    usage     : printvn();

```

Prints the ev voice number and name. This ev is defined as the work

static table. For instance, as follows.

```

1  -- VOICE No.2 --
2  { PIANO TONE    }
3  -- RECORDED     --

```

```

/***** FUNCTION No. 81 *****/

```

```

name      : print voice number with status
func      ;
usage     : printst();

```

Prints the ev voice status like "--RECORDED--", "--SYNTHESIZED--" and

--NO SOUND--" at line 3.

```

/***** FUNCTION No. 82 *****/

```

```

name      : print bank number
func      ;
usage     : printbn();

```

Prints the pb bank number and name. This pb is defined as the work

static table. For instance, as follows.

```

1  -- BANK No.2 --
2  { PIANO BANK   }

```

```

/***** FUNCTION No. 83 *****/

```

```

name      : print_name
func      : print name char {*****}
usage     : print_name( l,m,c,name );
                      int l,m;    line,

```

mode(0:normal,1:reverse)

```

                      int c;      left brace < , { or [
                      char n[];   source name string

```

Prints 12 byte characters like voice , bank or file name. The left

brace may be used properly in FZ-1 as follows.

```

'<'... disk name
'{'... file name like bank or voice.
'['... operation name like external porgrams

```

```

/***** FUNCTION No. 84 *****/

```

```

name      : print_num
func      : print out number
usage     : print_num( c,l,m,o,v );
                      int c,l,m;      colmn,line,mode in

```

print

```

                                int o;          print width+1
                                unsigned v;      print value
Displays the number by 10 digit. For instance,
print_num(0,0,0,4,99)

```

may print "099" at line0 ,column0 .

.pa

```

/***** FUNCTION No. 85 *****/

```

```

name : yes_no
func : operation yes no question
usage: answer = yes_no( l,s );
       int answer; (YES=10,NO=else)
       int l;      question message print line.
       char *s;    some message < 15;

```

Asks yes or no for the long-time operation like a disk loading or

wave synthesizing. The string s means some question message. IF NULL ,

then print the default one. When push [YES] for allowing operation,

all sound generations and all MIDI line data are cleared, and all

music keyboard and console switches become void. So it may be use with

end_job() no.88 for releasing these functions.

```

/***** FUNCTION No. 86 *****/

```

```

name      : jobbing
func      : animation display in jobbing.
usage     : jobbing();

```

Displays working animation "0000".

```

/***** FUNCTION No. 87 *****/

```

```

name      : end_job
func      : message end of job and wait new keyin
usage     : end_job(e,l);
           int e; error flag (fdd or copy)
           int l; line position

```

A error code 'e' is defined as follow.

```

#define NEXTDATA      (OK+1)          /* next data exist
*/

```

```

#define OK                0
#define ERROR             -1
#define BADDISK           (ERROR-0)      /* disk fatal error
occar          */
#define SAMEFILE         (ERROR-1)      /* find same file in
mopen()        */
#define NOSPAC           (ERROR-2)      /* no disk space
*/
#define NOFILE           (ERROR-3)      /* not file file in
mopen()        */
#define PROTECTED        (ERROR-4)      /* protected disk
*/
#define SINGLE           (ERROR-5)      /* single side diskket
*/
#define NOREADY          (ERROR-6)      /* not ready disk
*/
#define NOSAME           (ERROR-7)      /* verify error
*/
#define ENDFILE          (ERROR-8)      /* end of file      (cance-
led)          */
#define TIMEOUT          (ERROR-9)      /* port or midi dump
time out      */
#define PORTERROR        (ERROR-10)     /* port or midi data
format        */
#define NOMEMORY         (ERROR-11)     /* memory full error
*/
#define NODIR            (ERROR-12)     /* dir area full >64
*/
#define VNERROR          (ERROR-13)     /* Voice number error
*/
#define BNERROR          (ERROR-14)     /* Bank  number error
*/

```

.pa

For instance of function 85~87,189,190.

```

        if( yes_no( 6,"" )==YES ){
dido()    /*freeze interapt */
while( your_long_operation()==DOING )
        jobbing();
eido();  /* release interpat */
        end_job( OK,6 );
}

```

.PA

```

+++++
+++++

```

```

        FZ-1 voice,bank and effect parameters edit functions group
+++++
+++++

```

/****** FUNCTION No. 88 *****/

```

name      : generator
func      : generator operation
usage     : generator();

```

Operates as 'VOICE EDIT/CREATE VOICE/TRUNCATE'. Edit voice truncate

address which is selected by the ev.

```

/***** FUNCTION No. 91 *****/
name      : envelop
func      : envelop operation
usage     : envelop( vv );
              int vv;          (0:DCA 1:DCF)

```

Operates as VOICE EDIT/CREATE VOICE/DCA ENVELOPE' or 'VOICE EDIT/CREATE

VOICE/DCF ENVELOPE'.

```

/***** FUNCTION No. 96 *****/
name      : loop_set
func      : loop_set operation
usage     : loop_set();

```

Operates as 'VOICE EDIT/CREATE VOICE/LOOP SET'

```

/***** FUNCTION No. 97 *****/
name      : lfo_set
func      : lfo_set operation
usage     : lfo_set();

```

Operates as 'VOICE EDIT/CREATE VOICE/LFO SET'

```

/***** FUNCTION No. 98 *****/
name      : velocity_sence
func      : velocity_sence operation
usage     : velocity_sence();

```

Operates as 'VOICE EDIT/CREATE VOICE/VELOCITY SENSE'

```

/***** FUNCTION No. 99 *****/
name      : tuning
func      : key set opearation
usage     : tunnig();

```

Operates as 'VOICE EDIT/CREAT VOICE/TUNE MEMORY READ'

```

/***** FUNCTION No. 100 *****/
name      : delete_voice

```

```
func      : delete_voice operation
usage     : delete_voice();
```

Operates as 'VOICE EDIT/DELETE VOICE'

```
/****** FUNCTION No. 101 *****/
name      : define_bank
func      : define_bank operation
usage     : define_bank();
```

Operates as 'BANK EDIT/DEFINE BANK'

```
/****** FUNCTION No. 102 *****/
name      : create_bank
func      : create_bank operation
usage     : create_bank();
```

Operates as 'BANK EDIT/CREATE BANK'

```
/****** FUNCTION No. 103 *****/
name      : delete_bank
func      : delete_bank operation
usage     : delete_bank();
```

Operates as 'BANK EDIT/DELETE BANK'

```
/****** FUNCTION No. 104 *****/
name      : delete_area
func      : delete_bank_area
usage     : dearea_bank();
```

Operates as 'BANK EDIT/DELETE AREA'

```
/****** FUNCTION No. 105 *****/
name      : bender_range
func      : bender_range operation
usage     : bender_range();
```

Operates as 'EFFECT/BEND RANGE'

```
/****** FUNCTION No. 106 *****/
name      : vol_dev
func      : volume device editor (moduW,after,foot)
usage     : vol_dev( vv );
            short vv[];      parameter pointer like
            &pare.mod_lfp    ... MODU. W
            &pare.aft_lfp    ... AFTER
            &pare.fot_lfp    ... FOOT
```

Operates as 'EFFECT/MODULATION' , 'EFFECT/AFTER TOUCH' or
'EFFECT/FOOT'

VR' which is pointed by vv.

```

/***** FUNCTION No. 107 *****/
name      : midi_function
func      : midi_function operation
usage     : midi_function();

```

Operates as 'EFFECT/MIDI FUNCTION'

.pa

```

/***** FUNCTION No. 108 *****/
name      : set_copy
func      : set_copy voice or bank number
usage     : set_copy( mode );
              int mode;
              mode = VOICE or BANKE
              VOICE_COPY      0
              VOICE_REPLACE   1
              BANK_COPY       2
              BANK_REPLACE    3

```

Operates as 'VOICE EDIT/COPY VOICE' , 'VOICE EDIT/REPLACE VOICE' ,

'BANK EDIT/COPY BANK' or 'BANK EDIT/REPLACE BANK', which is selected by

mode.

```

/***** FUNCTION No. 109 *****/
name      ; send_excl
func      : send exclusive effect midi
usage     : send_excl( en,ev );
              int en; effect number
              int ev; effect value

```

Send exclusive MIDI data formatted as follow.

F0 - 44 - 02 - 00 - 7n 78 en - ev - F7

7n : n means MIDI basic channel number.

ev : effect value (00~7F)

en : effect number defined as follows

0 = bender depth ,1,2,18~7F undefined

	lfodcp	lfodca	lfodcf	lfodcq	dca	dcf	dcq
modoW	03	04	05	06	07	08	09
footV	0A	0B	0C	0D	0E	0F	10
after	11	12	13	14	15	16	17

```

/***** FUNCTION No. 110 *****/
name      : define_voice
func      :
usage     : define_voice();

```

Operates as 'DEFINE VOICE'.

```

/***** FUNCTION No. 111 *****/
name      : keyboard_set
func      :
usage     : keyboard_set();

```

Operates as 'KEYBOARD SET'.

.PA

```

+++++

```

FZ-1 source select functions group

```

+++++

```

```

/***** FUNCTION No. 112 *****/
name      : level_fix
func      :
usage     : level_fix();

```

Operates as 'SOURCE SELECT/SAMPLING/LEVEL SET'.

```

/***** FUNCTION No. 113 *****/
name      : length_set
func      : set length and sampling freq
usage     : length_set();

```

Operates as 'SOURCE SELECT/SAMPLING/LENGTH SET'.

```

/***** FUNCTION No. 114 *****/
name      : length limit
func      : memory limit select
usage     : time = length_limit();
           int time;          rest memory time (10ms)
remark    : use external : length,memsize

```

Gets the sampling time in the rest of pcm memory. The zero means that

the full memory is used by the recording or synthesizing.

```

/***** FUNCTION No. 115 *****/
name      : rec_do

```



```

func      : recording operation
usage     : rec_do( mode );
              int mode;      OK(0)      for auto
                              ERROR(-1) for manual

```

Operates as 'SOURCE SELECT/SAMPLING/AUTO SAMPLING' or 'SOURCE SELEC

T/SAMPLING/MANUAL SAMPLUNG' which is selected by mode.

.pa

/***** FUNCTION No. 117 *****/

```

name      ; init_voice
func      : initialize voice data before a recording
usage     : err = init_voice( vp,ll );
              int err;      OK(0)      :normal
                              NOMEMORY  :error

```

```

                                struct voicedata *vp;   voicedata
pointer
                                long ll;                voice
length(WORD);
                                if ll is zero, all address set
as zero.

```

```

remark    : vp->loop is null, then all parameter initiali-
ze
            isn't null, address only.

```

Initializes a voice parameters except vp->loop before
pcm_recording

or synthesizing operations. The NOMEMORY return means the
out of

memory for this initializing. When this vp>loop is null, all
voice

parameters are preseted by the default value. When not, they
are

initialized the address parameter like the vp>genst or
vp>loop[],

only.

/***** FUNCTION No. 118 *****/

```

name      : check_delete
func      : check voice delete for user
usage     : err = check_delete( vv );
              int err; OK for delete, ERROR for

```

return

struct voicedata *vv; voice pointer

Print a message 'VOICE EXSIST, DELETE (YES/NO) ?' at line 4 and 5.

And check whether YES or NO key, When YES, call the rec_delete()

function.

/***** FUNCTION No. 119 *****/

name : rec delete

func : delete wave without key pos,name

usage : rec_wdelete(vv);

struct voicedata *vv; voice pointer

Delete wavedata and voice_parameters except the key_position and the

voice name.

/***** FUNCTION No. 120 *****/

name : preset_wave()

func : set preset wave data

usage : preset_wave()

Operates as 'SOURCE SELECT/WAVE SYNTHESIS/PRESET WAVE'.

.pa

/***** FUNCTION No. 121 *****/

name : sin_add()

func : set sin add synthesizer parameters.

usage : sin_add() ;

static :

unsinged short sintable[MAXSIN]; sin add

table(0~255)

Operates as 'SOURCE SELECT/WAVE SYNTHESIS/SIN SYNTHESIS'.

/***** FUNCTION No. 123 *****/

name : cut_sample()

func : cut from pcm data

usage : cut_sample()

Operates as 'SOURCE SELECT/WAVE SYNTHESIS/CUT SAMPLE'.

/***** FUNCTION No. 124 *****/

name : hand_drawing()

func : draw wave data by hand (or cursor)

usage : hand_drawing()

Operates as 'SOURCE SELECT/WAVE SYNTHESIS/HAND DRAWING'.

```

/***** FUNCTION No. 126 *****/
name      :  add_select()
func      :
usage     :  add_select( mode )
                int mode;          1 : for rev select
                                   2 : for mix,x_mix select.

static :
        unsigned short add_v1,add_v2;  add voice number1,2
(0~63)
```

Operates as 'SOURCE SELECT/MIX WRITE/VOICE SELECT' , 'SOURCE SELECT/

XMIX WRITE/VOICE SELECT' or 'SOURCE SELECT/REV WRITE/VOICE SELECT'

which is selected by mode.

```

/***** FUNCTION No. 127 *****/
name      :  add_level()
func      :
usage     :  add_level()
static :
        unsigned short add_l1,add_l2; add level 1,2 (0~255)
```

Operates as 'SOURCE SELECT/MIX WRITE/LEVEL SET' or 'SOURCE SELECT/

XMIX WRITE/LEVEL SET'. The add_l1 and add_l2 are used as the same

meanings.

.pa

```

/***** FUNCTION No. 128 *****/
name      :  add_delay()
func      :
usage     :  add_delay()
static :
long add_dly;  add delay address offset */
```

Operates as 'SOURCE SELECT/MIX WRITE/LEVEL SET' or 'SOURCE SELECT/

XMIX WRITE/LEVEL SET'.

```

/***** FUNCTION No. 129 *****/
    name      :  add_detune()
    func       :
    usage      :  add_detune()
    static     :
short add_t1,add_t2;    detune tune (-127 ~ 127) */

```

Operates as 'SOURCE SELECT/MIX WRITE/DETUNE' or 'SOURCE SELECT/

XMIX WRITE/DETUNE'.

```

/***** FUNCTION No. 130 *****/
    name      :  add_cross()
    func       :
    usage      :  add_cross()
    static     :
long add_xs1;  start cross address (WORD address)
long add_xs2;  end   cross address (WORD address)

```

Operates as 'SOURCE SELECT/XMIX WRITE/CROSS ZONE'.

```

/***** FUNCTION No. 131 *****/
    name      :  mix_exe
    func       :  mix excute (0=MIX, 1=CROSS, 2=REVERSE)
    usage      :  mix_exe( mode );
                  int mode;
    remark     :  dido() eido() in use.

```

Operates as 'SOURCE SELECT/MIX WRITE/EXECUTE' , 'SOURCE SELECT/XMIX

WRITE/EXECUTE' or 'SOURCE SELECT/REV WRITE/EXECUTE' which is selected

by mode.

.pa

```

/***** FUNCTION No. 132 *****/
    name      :  check mix excute
    func       :  check add_v1 & add_v2 before mixexecute.
    usage      :  err = chk_mix( mode );
                  int err; (OK(0) normal,ERROR(-) NG)
                  int mode;
                  (0=MIX, 1=CROSS, 2=REVERSE)

```

Check the add_v1 and the add_v2 for mixing operating. The

ERROR

return means that the voice selected by add_v1 or add_v2 is a

synthesized voice, or ,the evn and add_v1 or add_v2 selectes the same

voice when mix or x-mix mode. So these voice number ,add_v1 or add_v2,

are not able for calculating.

```

/***** FUNCTION No. 133 *****/
name      : init synthesizer
func      : initialize for one wave synthesizer
usage     : init_synthe( st );
           int st; OK      : first set
           ERROR : second set

```

Intialize the voice parameters for wave_synthesized voice.

```

/***** FUNCTION No. 134 *****/
name      : preset_write
func      : preset_write calculaterfor PCM data
usage     : preset_write( p );
           int p;          preset wave number
           1 = saw-Tooth
           2 = Square
           3 = Pulse
           4 = Double sin
           5 = Saw-pulse
           6 = random

```

Write a preset wave synthesized data into the voice selected by evn.

MUST be called after init_synthe(ERROR).

```

/***** FUNCTION No. 135 *****/
name      : sin_write
func      : sin_write calculaterfor PCM data
usage     : sin_write();

```

Write a sin wave synthesized data into the voice selected by evn.

MUST be called after init_synthe(ERROR).

```

/***** FUNCTION No. 136 *****/

```

```

name      : cut_write
func      : cut_write calculaterfor PCM data
usage     : cut_write( st,ed );
            long st,ed; cut data address (WORD.)

```

Write a cut wave synthesized data into the voice selected by evn.

MUST be called after init_synthe(ERROR).

```

/***** FUNCTION No. 137 *****/

```

```

name      : mix
func      : mix calculater for PCM data
usage     : mix( mode );
            int mode; OK for mix, ERROR for xmix

```

Write a mix or x-mix calculated wave data selected by add_v1 and

add_v2 into the voice selected by evn. All static parameter for mixing

must be stored before calling as follows.

```

add_v1    DBS      1 ; source 1 voice # (0~63)
add_v2    DBS      1 ; source 2 voice # (0~63)
add_l1    DBS      1 ; source 1 mix level (0~255)
add_l2    DBS      1 ; source 2 mix level (0~255)
add_t1    DBS      1 ; source 1 detune (-127~127)
add_t2    DBS      1 ; source 2 detune (-127~127)
add_dly   DBS      4 ; source 2 delay WORD address
add_xs1   DBS      4 ; xmix start WORD address
add_xs2   DBS      4 ; xmix end WORD address

```

```

/***** FUNCTION No. 138 *****/

```

```

name      : rev
func      : rev calculaterfor PCM data
usage     : rev();

```

Reverse the pcm wave data selected by add_v1 into the voice selected

by evn. It is able to calculate when the add_v1 voice is equal to evn

voice.

.PA

```

+++++
+++++

```

FZ-1 disk functions group

```

+++++

```

++++

common parameter defined as follows.

mode:	SAVE	0	... transfer data to device
	LOAD	1	... transfer data from device
	MERGE	2	... merge data with device
	VERIFY	3	... verify data with device
sta :	DATA_STAT	0	... full data
	VOICE_STAT	1	... voice data selected by
	BANK_STAT	2	... bank data selected by pbn
	EFFECT_STAT	3	... effect data
dev :	DISK_DEV	0	... FZ-1 floppy disk
	PORT_DEV	1	... external port i/o dump
	MIDI_DEV	2	... MIDI exclusive dump

/****** FUNCTION No. 139 *****/

```
name      : load_all
func      : data transfer waveram form DISK,PORT,MIDI
usage     : load_all( mode,sta );
           mode : LOAD(1) ,MERGE(2), VERIFY(3)
           sta  : DATA_STAT(0),BANK_(1),VOICE_(2)
```

Operates as 'DATA DUMP/LOAD VOICE' , 'DATA DUMP/MERGE VOICE' or 'DATA

DUMP/VERIFY VOICE.

/****** FUNCTION No. 140 *****/

```
name      : del_asbefor
func      : data transfer from device to wavemem
usage     : del_asbefor( mode,dev,sta,name );
           int mode;      load mode
(Load(1),MERGE(2),VERIFY(3))
           int dev;       device number
(DISK(0),PORT(1),MIDI(2))
           int sta;       files
status(FULL,BANK,VOICE,EFFECT,PROG)
           char *name;    file name (in use dev==DISK )
```

Delete the wave parameter and memory before loading and load this

data.

/****** FUNCTION No. 141 *****/

```
name      : save_all
```

```

func      : data transmit waveram to DISK,PORT,MIDI
usage     : save_all( mode,sta );

```

Operates as 'DATA DUMP/SAVE VOICE'

```

/***** FUNCTION No. 142 *****/
name      : erase_all
func      : erase disk file
usage     : erase_all( sta );
              int sta;   file status
(DATA,BANK,VOICE,EFFECT)

```

Operates as 'DATA DUMP/ERASE VOICE'

```

/***** FUNCTION No. 143 *****/
name      : format_all
func      : listing voice bank files in DISK
usage     : format_all();

```

Operates as 'DATA DUMP/FORMAT DISK'

```

/***** FUNCTION No. 144 *****/
name      : print_device name
func      : print device name
usage     : print_dev( sta );
              int sta;           data status

```

Print the selected device number information at line 1~3.

```

/***** FUNCTION No. 145 *****/
name      : select dev
func      : selct device number
usage     : select_dev();

```

Operates as 'DATA DUMP/SELECT DEVICE'.

.PA

```

+++++
+++++

```

Level 2 disk iocs functions group

```

+++++
+++++

```

```

/***** FUNCTION No. 147 *****/
name      : mopen
function: mopen file name
usage     : err = mopen(name,sta,fbuf,dbuf);
              int err;           ERROR=open error
              char *name;       file name
              int sta;          file status
              struct fcb *fbuf;

```



```

                                char dbuf[SYSSIZ]; 1024 byte buffer

struct fcb {
    struct {
        unsigned int sc;    /* start cluster number
000 */
        unsigned int ec;    /* end
*/
    } dbp[MAXDBP];
    unsigned int mp;        /* map cluster number
100 */
    unsigned int np;        /* now
102 */
    unsigned int dp;        /* dbp pointer
104 */
};

    sta :      DATA_STAT      0      ... full data
            VOICE_STAT      1      ... voice data selected by
evn
            BANK_STAT      2      ... bank data seleted by pbn
            EFFECT_STAT      3      ... effect data

```

Opens a FZ-1 file for read. A detaied explanation of err is showd in

end_job(). The OK return means the success in opening file.

```

/***** FUNCTION No. 148 *****/
name      : mcreat
function   : mcreate file name / open file
usage      : err = mcreat(name,sta,fbuf,dbuf);
                                int err;          OK:mcreat done
                                char *name;        mcreate name
                                int sta;          file status
                                struct fcb *fbuf;
                                char dbuf[SYSSIZ];

```

Opens a FZ-1 file for write.

```

/***** FUNCTION No. 149 *****/
name      : mclose
function   : mclose open file
usage      : err = mclose( fbuf,dbuf );
                                int err;          mclose error
                                struct fcb *fbuf;
                                char dbuf[SYSSIZ];

```

Closes a FZ-1 file after data writing.

.pa

```

/***** FUNCTION No. 150 *****/

```

```

name          : mread
function       : mread ls cluster ( ls * 1024 byte)
usage         : err = mread(f,ls,data);
               int err;          mread error
               struct fcb *f;
               int ls;          mread cluster
#
               char data[];     data buffer
remark        : if datasize less than ls*1024, system
down

```

Read blocked data from FZ-1 file. A detaied explanation of err is

showd in end_job().

```

/***** FUNCTION No. 151 *****/
name          : mwrite
function       : mwrite ls cluster ( ls * 256 byte)
usage         : err = mwrite(f,ls,data);
               int err;          mwrite error
               struct fcb *f;
               int ls;          mwrite cluster
#
               char data[];     data buffer
remark        : if endof dbp, ls!=1 fatal error.

```

Writes blocked data into FZ-1 file. A detaied explanation of err is

showd in end_job().

```

/***** FUNCTION No. 152 *****/
name          : delete
function       : delete file
usage         : err = delete( name,sta );
               int err;          delete error
               char *name;       delete file
               int sta;          file status

```

Delete a FZ-1 file.

For the instance for the function 147~152 as follows.

```

some_disk_operation( name )
char *name;
{
    struct fcb fbuf;
    char      dbuf[SYSSIZ];

```

```

if( mcreat( name,VOICE_STAT,&fbuf,dbuf ) == OK ) {
    mwrite(fbuf,1,dbuf);
    mclose( fbuf,dbuf );
}
}
.pa
/***** FUNCTION No. 153 *****/
name      : serch
function: serch file name in dir
usage     : dpn = serch( name,sta,d ):
            int dpn;          find file posion di
                                <0      fdc err occur

            char *name;      filename;
            int sta;  file extension status
            struct syspar *d:

Initialize fdc call before disk operation
name==0 : serch null dir for write
name==1 : only read dir area with disk_name

            if sta's msb is 1 ,High byte of sta is egnored.

/***** FUNCTION No. 154 *****/
name      : catserch
function: blank cat serch
usage     : catnum = catserch( nth );
            int catnum;      nth serched cat num
                                ERROR =, no file space

Serch a blank sector from cat table.

/***** FUNCTION No. 155 *****/
name      : catset
func      : set cat bit
usage     : catset( i );
            int i;  claster position

Set a bit at cat table.

/***** FUNCTION No. 156 *****/
name      : catreset
func      : reset cat bit
usage     : catreset( i );
            int i;  claster position

Reset a bit at cat table.
.pa
+++++
+++++

```

FZ-1 wave data load/save functions group

+++++

/***** FUNCTION No. 157 *****/

```

name      : save
func      : data transfer from wavemem to device
usage     : save( mode,dev,sta,name );
              int mode;          save mode (SAVE(0)
only)
                                SAVE      ... 0
                                int dev;    device number
                                int sta;    files status
                                char *name; file name (12 charac-
ter)
    remark : use dido & eido

```

Save the wave data selected by sta into the device file selected dev

and name. The sta and dev are defined as follows.

```

sta :      DATA_STAT      0      ... full data
          VOICE_STAT       1      ... voice data selected by
evn
          BANK_STAT        2      ... bank data selected by pbn
          EFFECT_STAT       3      ... effect data

dev :      DISK_DEV        0      ... FZ-1 floppy disk
          PORT_DEV         1      ... external port i/o dump
          MIDI_DEV         2      ... MIDI exclusive dump

```

/***** FUNCTION No. 158 *****/

```

name      : load
func      : data transfer from device to wavemem
usage     : load( mode,dev,sta,name );
              int mode;          load mode
(Load,MERGE,VERIFY)
                                LOAD      ... 1
                                MERGE     ... 2
                                VERIFY    ... 3
                                int dev;    device number
                                int sta;    files status
                                char *name; file name (12 charac-
ter)
    remark : use dido & eido

```

Load , merge load or verify load the wave data selected sta from the

device file selected dev and name. Each meanings are explai-

ned in

save() function.

```

/***** FUNCTION No. 159 *****/
name      : format_do
func      : format disk & initialize cat,disk_top,dir
usage     : format_do(disk,pass );
           char *disk;      disk name
           char *pass;      pass word
remark    : use dido & eido

```

Formats diskette for FZ-1 use. Both hard and soft format will be done.

The pass_word is not use ,now. It may be usefull for some external

programs.

```

/***** FUNCTION No. 160 *****/
name      : set_files
func      : set file name listing voice bank files in DISK
usage     : err = set_files( sta,name );
           int err;          OK      : set name
                               ERROR-1 : ESC or PLAY
           int  sta;          file
status(DATA,PROG.etc)
           char name[12];    file name area MUST be
12
           remark    : if stat<0 ,High byte of sta is egnored.

```

Display file name, and select one. This function are used when every

load operations. For instance as follows.

```

sample_set_files()
{
  char name[12];

  if( set_files( DATA_STAT,name )==OK )
    some_disk_operation( name );
}
.PA
+++++
                                     Lcd handling functions group
+++++

```

```

/***** FUNCTION No. 161 *****/
name      : lcd initialize
func      : send dispoff/mac/time/dirpon/"ZZ-5000 HEAR"
usage     : lcdinit();

```

Initializes lcd unit and display power on messages.

```

/***** FUNCTION No. 162 *****/
name      : print
function: print string at line,column
usage     : print(c,l,b,s);

int c;    column pos (0<= c <16)
int l;    line   pos (0<= c <8 )
int b;    back color
           0 = normal
           1 = reverse &
           2 =
reverse,only

char *s; null terminated string

```

Print a string character terminated NULL code at line l , column c by

mode b. All character printing function are used this function.

```

/***** FUNCTION No. 163 *****/
name      : cls
function   : clear screen
usage     : cls(xs,ys,xs,ys,c);

int xs,ys; left top graphic
pos

int xe,ye; right bottom pos
int c;      0=clear reset dot
            1=black set dot
            2=exclusiv dot
            else=send xysheet[] to

lcd

MSB = 1, no send

lcd();
remark      : xs<=xe && ys<=ye must be !!
            if !(0<=x<=95 && 0<=y<=63) broke
memory

```

Clear lcd screen limited by xs,ys,xs and ye. When the msb of the

parameter "c" is set, this cls function does only clear the graphic

memory named by xyseet, does not send the graphic data to lcd unit.

And when the parameter "c" is 3, then send the graphic data in the

memory named by xysheet to lcd unit. So it is able to write all

graphic data at a sitting, after setting into the xysheet memory. The

graphic dot arrangment of the xysheet is showed as follows.

```
dot set      : xysheet[ x/8 + y*(96/8) ] |= (1<<(x&7);
              (0<=x<=95 && 0<=y<=63)
```

.pa

```
/***** FUNCTION No. 164 *****/
```

```
name          : pset
function       : dot set/reset
usage         : pset(x,y,c);
               int x,y; graphic position
               int c    0=white, 1=black,
```

2=ex

MSB=1 ,no send to

lcd();

```
if !(0<=x<=95 && 0<=y<=63) broke
```

memory

Set a dot pointed by x and y. The parameter "c" means the same one in

the cls function.

```
/***** FUNCTION No. 165 *****/
```

```
name          : line
function       : draw line
usage         : line(xs,ys,xe,ye,c);
               int xs,ys; left top graphic
```

pos

```
int xe,ye; right bottom pos
```

```
int c;        0=white, else=black
```

```
if !(0<=x<=95 && 0<=y<=63) broke
```

memory

Draws a line pointed by xs,ys,xe and ye. The parameter "c" means the

same one in the cls function.

```

/***** FUNCTION No. 166 *****/
name      : boxline
function   : draw boxline
usage      : boxline(xs,ys,xs,ys,c);
            int xs,ys; left top graphic
pos
            int xe,ye; right bottom pos
            int c;      0=white, else=black
            if !(0<=x<=95 && 0<=y<=63) broke
memory
```

Draws a box pointed by xs,ys,xs and ye. The parameter "c" means the

same one in the cls function.

```

/***** FUNCTION No. 167 *****/
name      : lcd_vol
func       : lcd volume set, move constrast
usage      : lcd_vol();
```

Regulates the lcd contrast by the data entry volume while pushing

[disp] switch.

.PA

+++++

Wave memory handling functions group

+++++

```

/***** FUNCTION No. 168 *****/
name      : bdelete
func       : delete bankdata & voicedata using in bank
usage      : bdelete( bstep,bn );
            int bstep;      delete from bstep;
            struct bankdata *bn;
```

Delete area parameters and wave data which are used in this bank.

If the voice in the bn bank is used by other bank, then does not

delete this voice. The bstep is a offset. It should delete the voice

selected between bstep and bn->bstep.

```

/***** FUNCTION No. 169 *****/

```

```

    name      : wdelete
    func      : wave data delete
    usage     : wdelete( wn );
                struct voicedata *vn; voice data number

```

Deletes the voice and wave data which is pointed by wn. If the wn

voice has the common wave data which is used by other voices, then

does not delete wave data, does the only voice parameter.

```

/***** FUNCTION No. 170 *****/

```

```

    name      : wunuse
    func      : delete wave unused part
    usage     : wunuse( vn );
                struct voicedata *vn;          voice
number

```

Truncates the wave data limited vn->wavst - vn->genst and vn->waved -

vn->gened. The function "delete unused part" calls this subroutine.

And preseted vn->genst as vn->wavst and vn->gened as vn->waved, after

calling this function.

```

/***** FUNCTION No. 171 *****/

```

```

    name      : wend
    func      : check end of wave data
    usage     : wend( end );
                long *end;

```

remark : Return end of wave memory. Not check with mem-size.

Gets the end word address of wave memory into the long value

pointed by parameter "end".

.pa

```

/***** FUNCTION No. 172 *****/

```

```

    name      : wsame

```

```

func      : check same wave point data
usage     : match = wsame( vn );
            int match;          macth voice with vn;
                                ERROR : vn is only
one.
                                OK      : voice use
twice.

```

Check whether the vn voice has the same pcm wave data with other

voice or not. For instance, two voice parameter share the same wave data

after the copy voice function (VOICE EDIT /COPY VOICE).

```

/***** FUNCTION No. 173 *****/
name      : ad_adjust
func      : adjust all address by offset
usage     : ad_adjust( l,vn );
            long l;              adjust offset(WORD.)
            struct voicedata *vn;

```

Arranges all address parameters selected by vn at the address l. This

function is used when move all address pointers like gened or loopst

after deleting wave data.

```

/***** FUNCTION No. 174 *****/
name      : ad_chk
func      : address check and repair by memsize
usage     : err = ad_chk( v );
            struct voicedata *v;

```

Checks all address parameters whether some address parameters are out

of memory ,or not. The NOMEMORY return means this out of memory, and

OK return for normal.

```

/***** FUNCTION No. 178 *****/
name      : merge_bank
func      : merge source bankdata into dest bankdata
usage     : err = merge_bank( d,s );
            int err; (OK(0) merge done,ERROR(-1))

```

```

struct bankdata *d;
struct bankdata *s;      merge source

```

Merges the bankdata pointed by "s" into "d".The ERROR return means

wrong bank parameter. For instance, the result of bank
step(bstep)

number is over maxnumber 64.

```

.pa
/***** FUNCTION No. 181 *****/
name      : use wave
func      : check voice no.vp use number.
usage     : err = use_wave( vp );
           int vp;      serch voice number
           int err;
           0      : vp is null voice
           1      : first use in voice
           2~64   : second use
remark    : check common use voice. return n th. common

```

voice

Checks the use number of the voice selected by vp. If it is the
first

voice of the 64 voices, the return should be 1. When some
voices

shares one wave data, this function check what number it is.

```

/***** FUNCTION No. 182 *****/
name      : use voice
func      : check voice vp's use number.
usage     : err = use_voice( vp,bb );
           int vp;      serch voice number
           int bb;      serch bank number
           int err;
           0      : no use voice in bank
bb.
           1      : first use in voice
           2~64   : second use

```

Checks the use number in the bb bank for the vp voice . It is
able to

define areas by the same voices in the FZ-1 system. So, this
function

counts what number this vp voice is in the bb bank.

```

/***** FUNCTION No. 183 *****/
name      : in_bank
func      : check voice vp ,whether use in bank bb  or not
usage     : err = in_bank( vp,bb );
              int vp;          serch voice number
              int bb;          serch bank  number
              int err;
                      ERROR      : no use voice in
bank bb.
                      OK          ; use in bank

```

Check whether the bb bank uses the vp voice or not. The ERROR return

means that this vp voice is not used in the bank bb.

.pa

```

/***** FUNCTION No. 184 *****/
name      : set_wbvnum
func      : preset saving wave,bank,voice number
usage     : length = set_wbvnum( sta );
              int length;      data blocknumber
(1024byte)
              int sta;         data
status(DATA,BANK,VOICE)

```

Counts the block number of the dump format data which is selected by

sta. The return length means the total size of the block data. For

instance, the function set_wbvnum(DATA_STAT) means the full data

size of FZ-1 system, now. When the sta is BANK_STAT, it means the size

of the bank block which is selected by pbn, and when the sta is

VOICE_STAT, it means the size of the voice block which is selected by

evn. Each block sizes are stored as follows .

```

voice_num   DBS      2 ; --- voice number without null voice
bank_num    DBS      2 ; --- bank  number without null bank
wave_num    DBS      2 ; --- wave  block number used by voices
.PA

```

```

+++++
++++
                        Very low level functions group
+++++
LASTWORK
/***** FUNCTION No. 188 *****/
    name      : lcd(s)
    func      : send data to lcd module
    usage     : err = lcd(s);
                  int err;          err=OK, err=ERROR time-
out
                  char *s;          0xFF terminated string
    remark    : Use outp,inp
    Sends the bit image data to the lcd units. It may be not
    use for
external program.

/***** FUNCTION No. 189 *****/
    name      : dido
    func      : disable all interapt, close time,midi,key
    usage     : dido();

    Freezes the all interapt operation.

/***** FUNCTION No. 190 *****/
    name      : eido
    func      : disable all interapt, open time,midi,key
    usage     : eido();

    Releases the all interapt operaion.

/***** FUNCTION No. 191 *****/
    name      : cread
    func      : physical claster read
    usage     : err = cread(pcl,data);
                  int err;          read
err
                  int pcl;          cla-
ster#
                  char *data;      data
buf
    remark    : datasize must be greater than FDDSIzbyte

    Reads the next set of byte from a file. The detailed error
    examina
tions are showed in the function end_job(). OK(0) is for normal
return.

```

```

/***** FUNCTION No. 192 *****/
name      : cwrite
func      : physical cluster write/
usage     : err = cwrite(pcl,data);
                                int err;           write
err
                                int pcl;           cla-
ster#
                                char *data;        data
buf
remark    : datasize must be greater than
FDDSIZEbyte

```

Writes the next set of bytes to a files. The detailed error
examina

tions are showed in the function end_job(). OK(0) is for
normal

return.

.pa

```

/***** FUNCTION No. 194 *****/
name      : fdc_format()
func      : fdd first fdc_format
usage     : err = fdc_format();
                                int err;           error flag

```

Formats diskket as IBM format, which is defined for 1024
byte /

sector ,8 sector / hed , 2 hed / track , 80 track / disk-
ket. The

format data patern is 5Ah. Only hard format will be done.

```

/***** FUNCTION No. 195 *****/
name      : fdc_init
func      : intialize fdd contlo
usage     : fdc_init();

```

Intializes fdc unit. Call this function before any fdd operati-
ons.

```

/***** FUNCTION No. 196 *****/
name      : seek
func      : seek hed at c cylinder
usage     : seek( c );
                                int c;   cylinder number(0~79)

```

FZ-1's diskette consists of 80 tracks. The cylinder number means this

track number.

```

/***** FUNCTION No. 199 *****/
name      : fdc_check
func      : check fdc status
usage     : st3 = fdc_check();
              int st3 ; 0 = OK no error for read &
write
              -5 = write protected
              -6 = not double side disk
              -7 = not ready, or motor broken
ken
              -1 = device fault, FDD broken
ken
```

Checks FDC condition which is defined as the parameter st3.

```

/***** FUNCTION No. 200 *****/
name      : movmem
func      : move memory by BKM
usage     : movmem(s,d,n)
              char *s;      source pointer
              char *d;      destination pointer
              unsigned n;    byte number (even)
```

Moves the memory block from one location to another. The n, which

means a byte number for moving, MUST be even number. It may be bug.

.pa

```

/***** FUNCTION No. 201 *****/
name      : cmpmem
func      : compare memory by BKM
usage     : err = cmpmem(s,d,n)
int err;   -1:ERROR not same)
              char *s;      source pointer
              char *d;      destination pointer
              unsigned n;    byte number (even)
```

Compares the memory block which is pointed by "s" and "d". The n MUST

be even number, too.

```

/***** FUNCTION No. 202 *****/
name      : setmem
```

```

func      : set memory by data
usage     : setmem(s,n,d);
           char *s;          set destination data
poiner
           int  d;          set data value
           unsigned n;      set number( even )

```

Sets the nth number of bytes of the data value "d" into the memory

block which is pointed by "s" .

/***** FUNCTION No. 203 *****/

```

name      : wpeek
func      : wave data phisical read
usage     : wpeek( ad,data,len )
           long ad;          wave ram addreaa(WORD)
           char *data;       read buffer point
           unsigned int l;   data length (BYTE even)

```

Puts a block data of the pcm sample into the wave_ram. The address ad

must be word address, BUT the byte count l must be the even byte

count. It may be the poor discord in FZ-1 system.

/***** FUNCTION No. 204 *****/

```

name      : wpoke
func      : wave data phisical read
usage     : wpoke( ad,data,len )
           long ad;          wave ram addreaa
(WORD.)
           char *data;       read buffer point
           unsigned int l;   data length (BYTE even)
)

```

Gets a block data of sample from the wave_ram. The address ad must be

word address, BUT byte count l must be byte count.

/***** FUNCTION No. 205 *****/

```

name      : wput
func      : wave data phisical read
usage     : wput( ad,data )
           long ad;          wave ram addreaa
(WORD.)
           int  data;        write data

```


Puts a word of the pcm sample into the wave_ram. The address must be

word address.

```
/****** FUNCTION No. 206 *****/
```

```
name      : wget
func      : wave data phisical read
usage     : data = wget( ad )
              long ad;          wave ram addreaa
(WORD.)
              int  data;        write data
```

Get a sample in data from wave_ram. The address ad must be word address.

```
/****** FUNCTION No. 207 *****/
```

```
name      : wcomp
func      : wave data phisical verify compare
usage     : wcomp( ad,data,len )
              long ad;          wave ram addreaa(WORD)
              char *data;       read buffer point
              unsigned int l;   data length (BYTE even)
```

This function compares the wave_data with the local buffer data. If

any difference exists, wcomp()'s return is ERROR(-1).

```
/****** FUNCTION No. 209 *****/
```

```
name      : cnvtdc
func      : convert unsigned binary into decimal character
usage     : cnvtbd( d,c,s );
              unsigned int d;   convert data
              int c;           character count (2-6)
              char *s;         char .
```

Converts the unsigned number d into the ascii decimal character

number which is pointed by s. And the c is defined the size of the

buffer.

```
/****** FUNCTION No. 210 *****/
```

```
name      : set_wid
func      : search wave data max & min between pos0 & pos1
usage     : set_wid( gmax,gmin,pos0,pos1 );
```

```
int    *gmax,*gmin;
long pos0,pos1; (WORD address)
```

Serches the max or min value of pcm data between the address limited

by pos0 and pos1. Thease values should be return by pointer, gmax and

gmin.

```
/***** FUNCTION No. 211 *****/
name      : iocreat
func      : io port open for write
usage     : iocreat(mode,sta,&fbuf,&work );
```

Sends a port open code to the slave FZ-1.The full and detailed

explanation for this dump format is described in "MIDI DATA FOR-MAT for

FZ-1".

```
.pa
/***** FUNCTION No. 212 *****/
name      : ioopen
func      : io port open for write
usage     : ioopen(mode,sta,&fbuf,&work );
```

Waits and recives a port open code from the master FZ-1.

```
/***** FUNCTION No. 214 *****/
name      : ininit
func      : load mode initialize
usage     : ininit();
```

Initializes a port unit for the slave mode.

```
/***** FUNCTION No. 215 *****/
name      : outinit
func      : save mode initialize
usage     : outinit();
```

Initializes a port unit for the master mode.

```
/***** FUNCTION No. 216 *****/
name      : iowrite
func      : io output byte
usage     : iowrite( work,byte );      00004,00008
                                     char *work; 1024 byte data buffer
```

int byte; data counter to send

Sends a blocked dump data to the slave FZ-1

```

/***** FUNCTION No. 217 *****/
name      : ioread
func      : io input byte
usage     : ioread( work,byte );    00004 00008
                                char *work;    1024 byte data buffer
                                int byte;      data counter to send

```

Receives a blocked dump data from the master FZ-1 .

```

/***** FUNCTION No. 218 *****/
name      : midiopen
func      : midi io input initialize
usage     : midiopen( mode,stat,buf,work );
                                int mode;      LOAD,MERGE,SAVE,VERIFY
                                int stat;      DATA,BANK,VOICE,EFFECT
                                char *buf;     256 byte buffer
                                char *work;    1024byte buffer

```

Receives the open MIDI dump code. The full and detailed explanation

for this dump format is described in "MIDI DATA FORMAT for FZ-1".
.pa

```

/***** FUNCTION No. 219 *****/
name      : midicreat
func      : midi io output initialize
usage     : midicreat( mode,stat,buf,work );
                                int mode;      LOAD,MERGE,SAVE,VERIFY
                                int stat;      DATA,BANK,VOICE,EFFECT
                                char *buf;     256 byte buffer
                                char *work;    1024byte buffer

```

Sends the open MIDI dump code.

```

/***** FUNCTION No. 220 *****/
name      : midiread
func      : master io input 1024byte
usage     : midiread( buf,work )
                                char *buf;     256 byte buffer
                                char *work;    1024byte buffer

```

Receives a block dump data from MIDI in.

```

/***** FUNCTION No. 221 *****/

```

```

name      : midiwrite
func      : master io output 1024byte
usage     : midiwrite( buf,work );
                char *buf;          256 byte buffer
                char *work;        1024byte buffer

```

Sends a block dump data to MIDI out.

```

/***** FUNCTION No. 222 *****/
name      : midipeek
func      : data into buf X byte
usage     : dataread( buf,byte )
                char *buf;          256byte buffer

```

```

/***** FUNCTION No. 223 *****/
name      : midipoke
func      : midi exclusive data for mo
usage     : midipoke( buf,byte)
                char *buf; send data pointer

```

int byte data count (byte)

Sends MIDI data

```

/***** FUNCTION No. 224 *****/
name      : midiclose
func      : master io output 1024byte
usage     : midiclose( buf );
                char *buf;          256 byte buffer

```

Sends 'MIDI CLOSE exclusive data'.

.pa

```

/***** FUNCTION No. 225 *****/
name      : mportout
func      : output port data with check_summ
usage     : mportout( buf,byte );
                char *buf; send data pointer

```

it ta count (byte)

Puts a block of data to external port with the check sum byte.

```

/***** FUNCTION No. 226 *****/
name      : sportin
func      : input port data with check_summ
usage     : sportin( buf,byte );
                char *buf; send data pointer

```

it ta count (byte)

Gets a block of data from external port. A block has the check sum

byte, so 1025 byte may be recieved when a block size is 1024byte.

.pa

```

+++++
++++

```

Mode functions group

```

+++++
++++

```

```

/***** FUNCTION No. 228 *****/

```

```

name      : play mode
func      : play mode define
usage     : play_mode();

```

This function operates as the PLAY mode in FZ-1.

```

/***** FUNCTION No. 230 *****/

```

```

name      : tune mode
func      : tune mode define
usage     : tune_mode( mode );
            int mode; TMOD : tuning mode
            RMOD : transpose mode

```

This function operates as when push TUNE or TRANSPOSE key on FZ-1.

```

/***** FUNCTION No. 232 *****/

```

```

name      : lcut
func      : limiter v between max and min
usage     : lcut( v,min,max );
            long *v;
            long min,max;

```

This is a small function which limits the value v between max and

min.

```

/***** FUNCTION No. 233 *****/

```

```

name      : lswap
func      : swap long data
usage     : lswap( a,b );
            long *a,*b;

```

This is a small function which swap long value a as b.

```

/***** FUNCTION No. 234 *****/

```

```

name      : ck_lcd
func      : full set dot
usage     : ck_lcd();

```

This function use for testing lcd system. First, set all lcd dot on.

Push any console key. Second, display all character code.

```

/***** FUNCTION No. 235 *****/
name      : disk checker
func      : check disk sector read/write
usage     : ckdisk();

```

This function may be used for some testing fdd system. It may erase the user disk in FZ-1. So MUST remove the sound or program disk, and insert a formatted disk in FZ-1 before call it.