CASIO DIGITAL SAMPLING KEYBOARD
MODEL FZ-1
DATA STRUCTURES

(For Software Developers)

PREPARED ON: MARCH 18, 1987

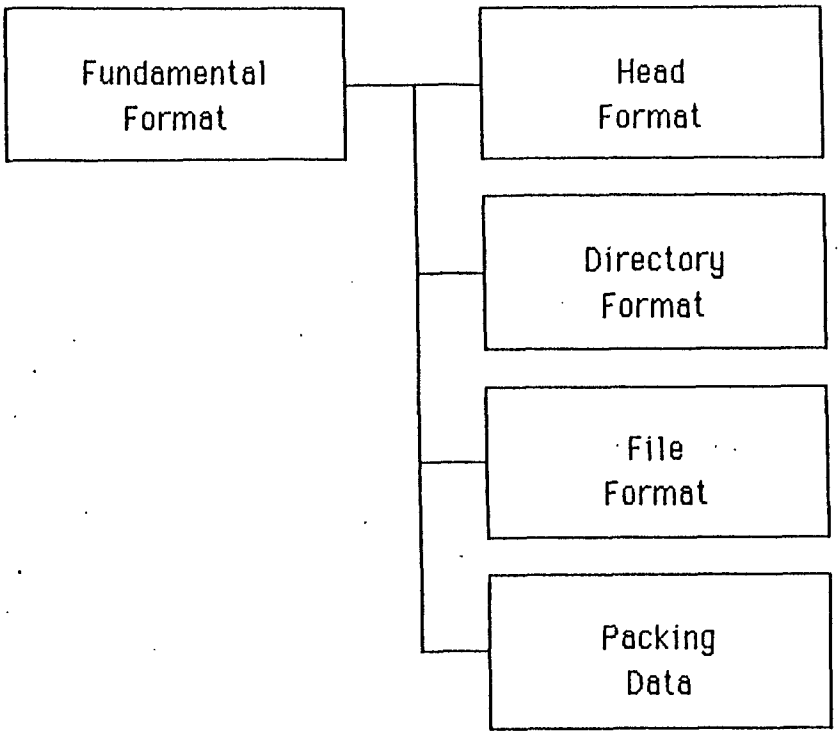
ORIGINALLY WRITTEN BY: T. SASAKI - R&D
TRANSLATED BY: O. ISHIYAMA - OVERSEAS DIVISION

CASIO TOKYO

Table of Contents

1. Outline of "Disk"

A floppy disk drive unit is built in the FZ-1 machine for use with high-density type 3.5' micro floppy disks. The disk drive is used for inputting/outputting data and parameters and also for loading expanded software programs.

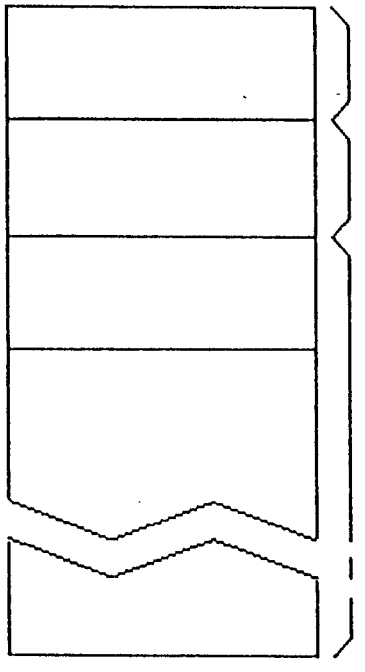


1-1. Fundamental Format

The system program which is installed in the FZ-1 machine has a format created on the basis of the IBM format and the FZ-1 formats a disk:

Double sided, 80 tracks, 8 sectors/track, 1,024 bytes/sector

The total capacity comes to 1,310,720 bytes.



loc

ø : head - a sector to be used for system

1 : dir - a sector to be used for file names

2 - 1279 : sectors to be used for data

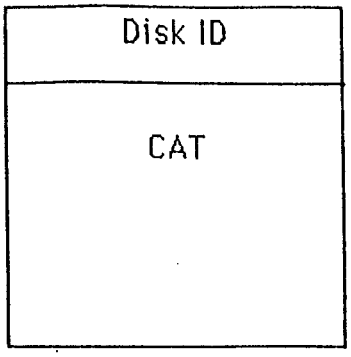
A logical sector *loc* whose formula is as shown below is occupied with headers, directories, and data.

loc = (16 x track) + (8 x head) + (sector - 1)

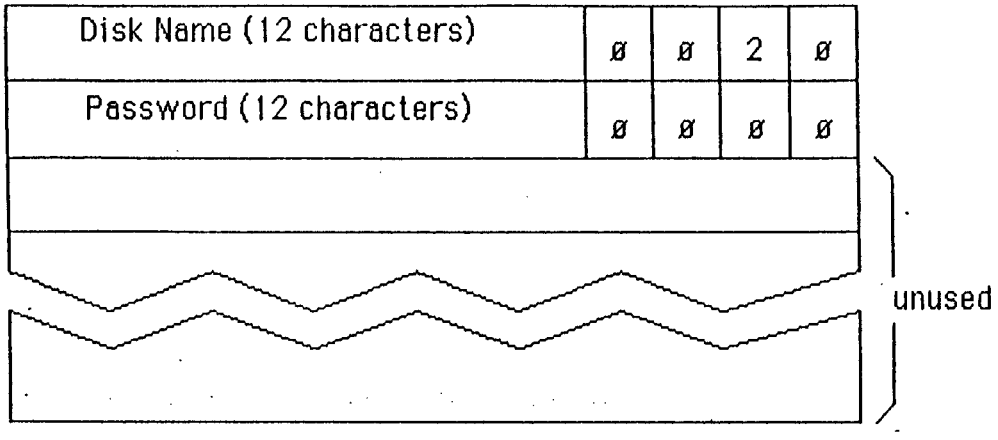
truck: ø - 79
head: ø - 1
sector: 1 - 8

1-2. Head Format

Head data, consisting of Disk Identification *Disk ID* and Cluster Allocation Table *CAT*, is placed exclusively at the logical sector ø.

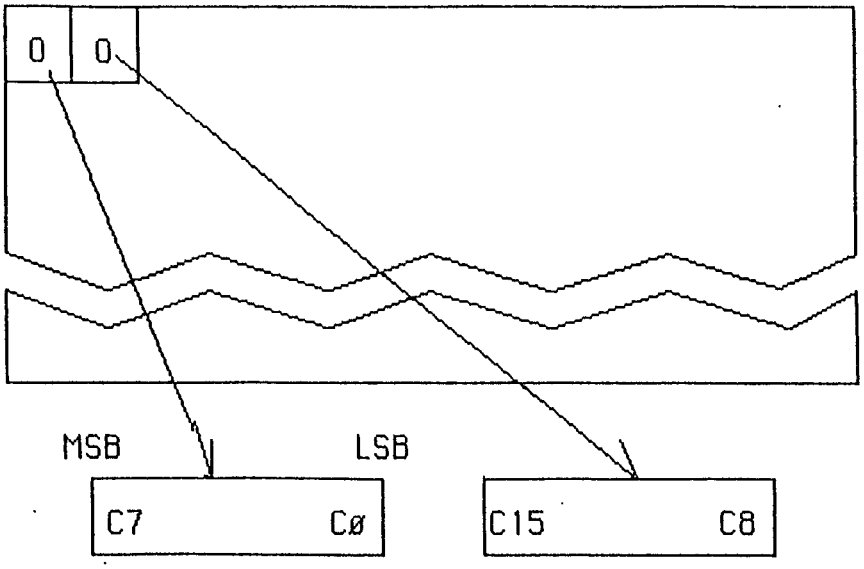


a) Disk ID (128 bytes) - Disk Identification



The Disk ID identifies a disk with the record of a disk name and a password.

b) CAT (768 bytes) - Cluster Allocation Table



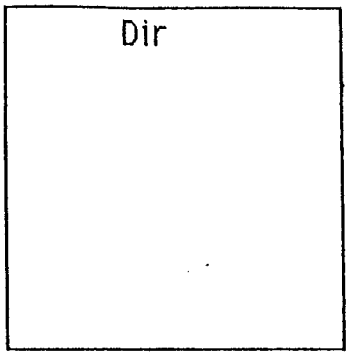
C *loc*: "1" for Used, "ø" for Unused.

The CAT indicates a status for entire sectors; The figure "1" denotes entire sectors are already used and "ø" denotes not yet used. The correspondence with sectors is the sequence from LSB to MSB, and from Low address to High address. The logical sector *loc* verifies these by the formula: $0 \leq loc \leq 1279$

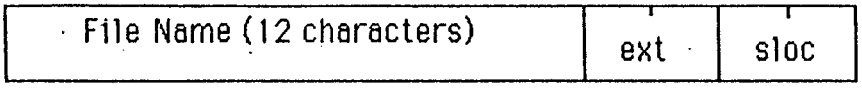
If a figure ø is obtained from the formula CAT[loc/8]&(1<<(loc%8)), then the entire sectors are not yet used. If it is not ø, then they are already used.

1-3. Directory Format

The Directory data *dir* is a sector where records file names, identifications and start logical clusters are recorded and placed exclusively at the logical sector 1.



a) Dir data



The Dir data consists of 16 bytes per file and a piece of floppy disk can store up to 64 files. The portion of a file is shown above.

b) File Name: Consists of 12 characters in the ASCII standard. The blank(s) is filled with SPACE (2øh). If the first 1 byte is NULL (øøh), then the directory is regarded as blank.

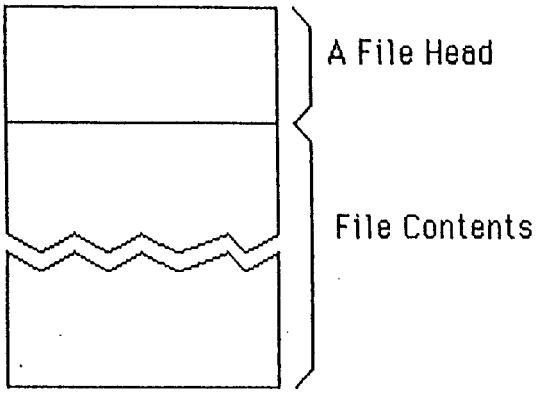
c) ext: A figure for 2 bytes indicating the data contents in the file. The lower byte denotes data content in each file, and the higher bytes denotes a file address. The FZ-1 allows you to save/load waveform data uninterruptedly on 2 pieces of floppy disk. The higher byte for *ext* takes ø as value for the first floppy disk and does 1 for the second disk.

Lower Byte for *ext*	Data Content
ø	Full Dump Data
1	Voice Data
2	Bank Data
3	Effect Data
4	Sequence Data
5	Expanded Program Data

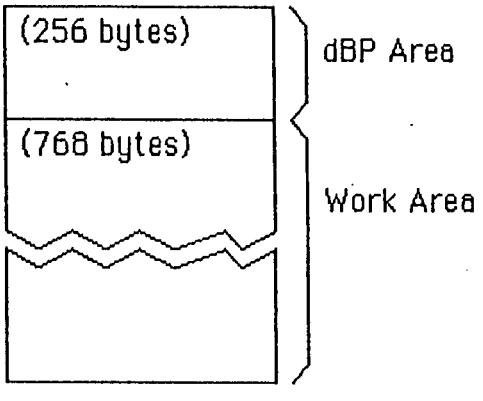
d) sloc: Denotes Start Logical Cluster, a logical sector heading the data which upcoming file will contain.

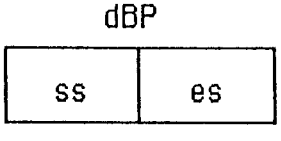
1-4. File Format

A file consists of a file head (1 sector) and file contents (N sectors). Each sector starts from the sector designated by *Dir* and the location of each sector is assigned at random over the entire area of a disk by the *dBP* (data Block Pointer) in the file head.

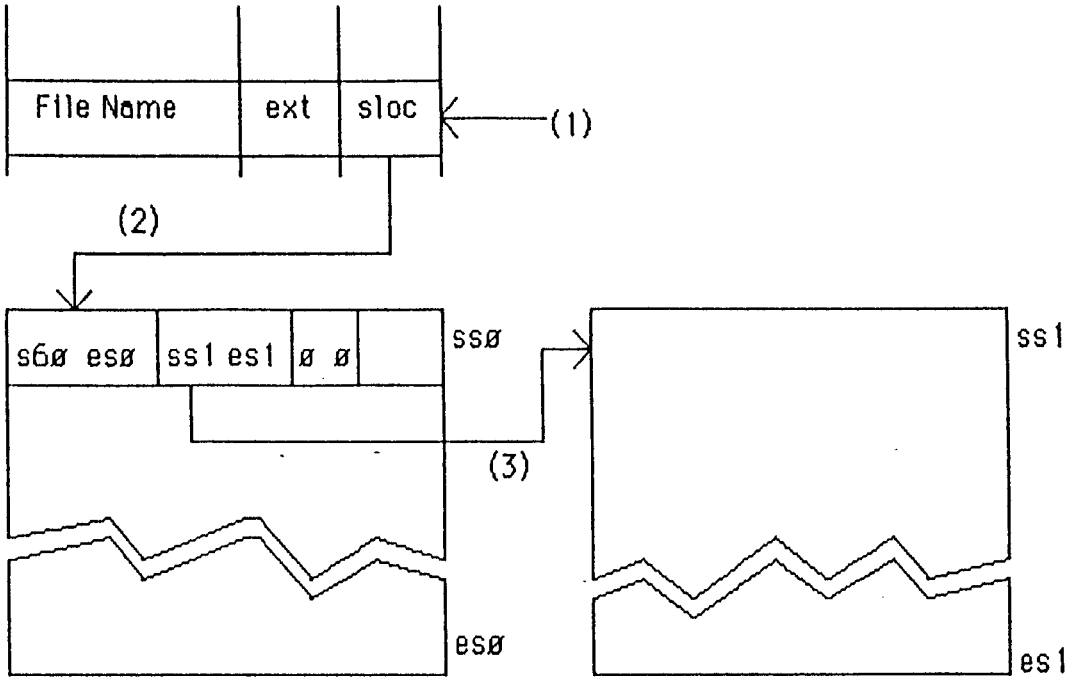


a) File Head: A file head consists of a dBP area (256 bytes) and a work area (768 bytes). The dBP (data Block Pointer) consists of a *ss* (Start Sector) and *es* (End Sector). These are pointers of 2 bytes each existing in the logical sector; *ss* points the head and *es* points the end of each sector even when file contents are randomly located over the entire area of a disk. There exists 64 dBP's in a dBP area, which is a cluster of dBP's. A file will end if a dBP of ss=es=ø exists or when all the 64 dBP's are consumed over.





b) File Search: The FZ-1 searches a file in the following method:



(1) Searches with *Dir* to find a file by the file name and *ext*.
(2) Learns from the file *sloc* which sector contains the file head and reads data from the sector ssø to the sector esø.
(3) Reads data from the sector ss1 to the sector es1 by the designation of next dBP.
(4) The file ends when the next dBP comes to ss2=es2=ø.

1-5. Data Packing

All data are structured by 1,024 bytes into a block. The portions which are divided by the block bytes will be explained with charts according to the file data. Files in general are randomly located in a whole disk area; however, the illustration shows files in consecutive locations.

A parameter classification which is used within a block contains nothing but a file field for the dump over External Port and MIDI. Refer to the Outline of Parameters for the detailed contents of Bank, Voice and Effect parameters.

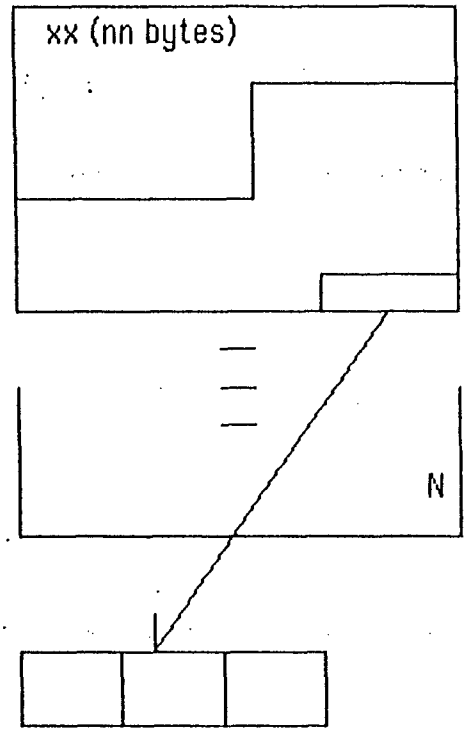


Chart - Legends:

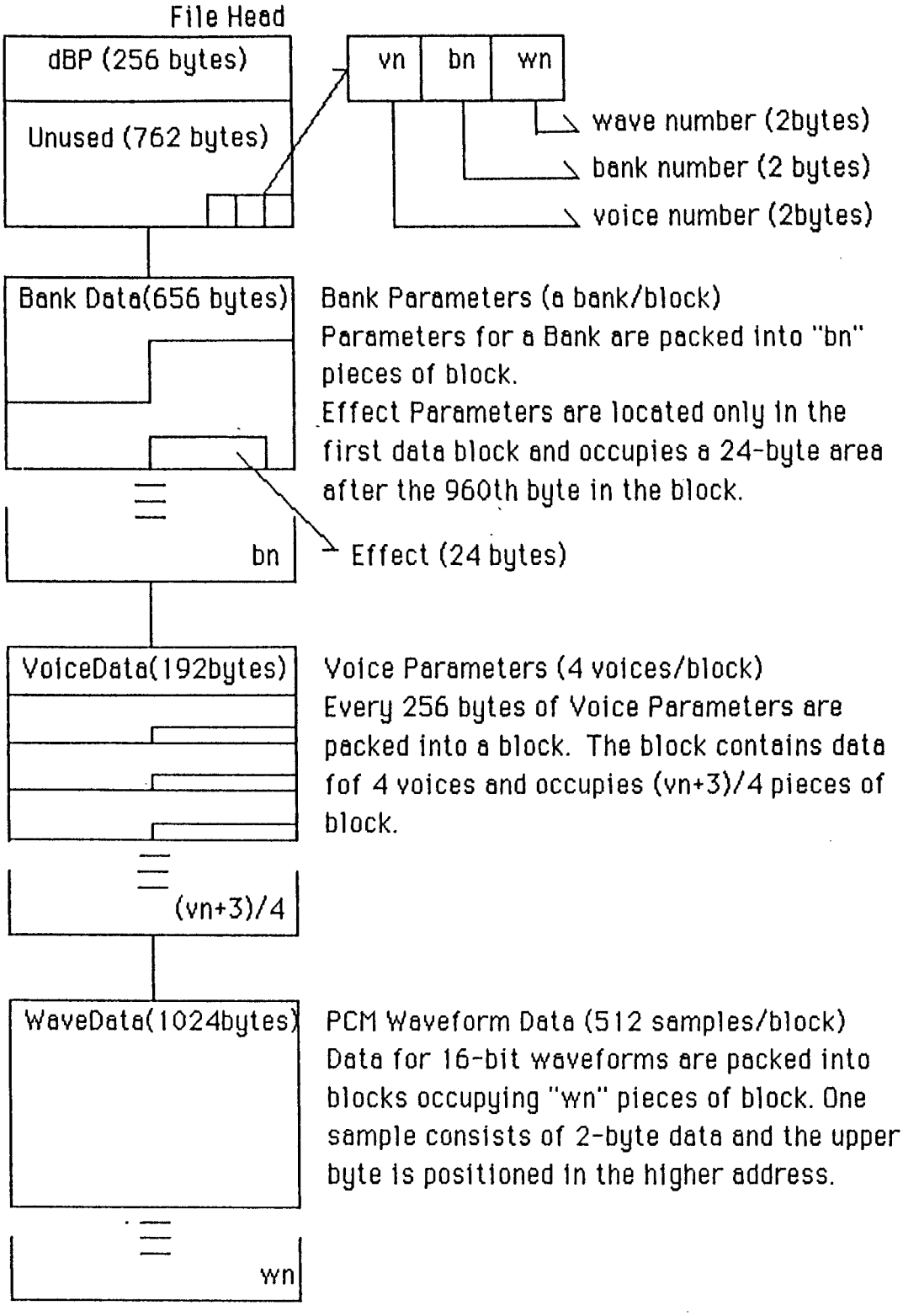
a) The big box denotes a block of 1,024-byte data.

b) The classification inside the box indicates that "nn" bytes are consumed for "xx".

c) The rectangule without upper line shows that "N" pieces of the same block will follow after the block.

d) Itemized explanation will be provided with an arrow out of a box if the classified box is too small.

b) Full Data ext=ø

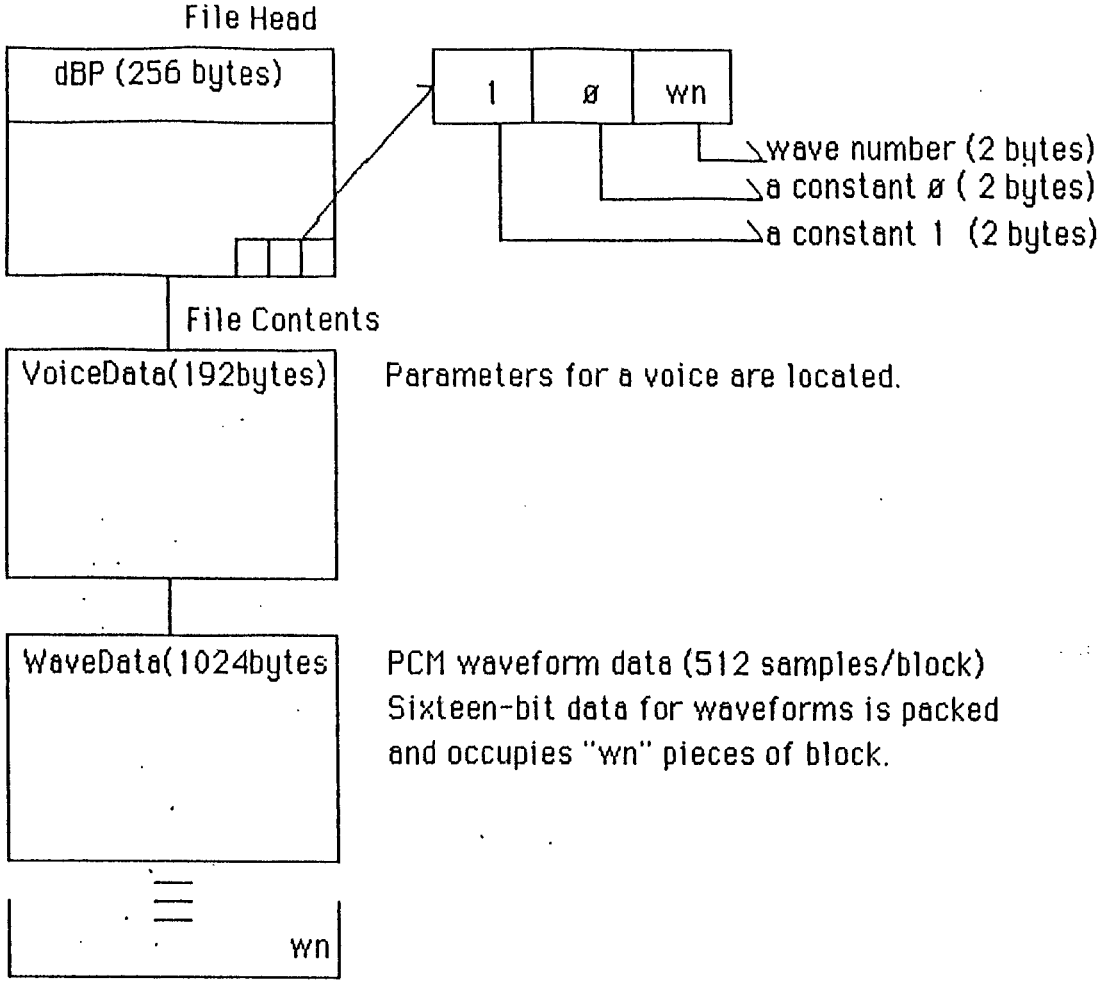


Bank Parameters (a bank/block)
Parameters for a Bank are packed into "bn" pieces of block.
Effect Parameters are located only in the first data block and occupies a 24-byte area after the 960th byte in the block.

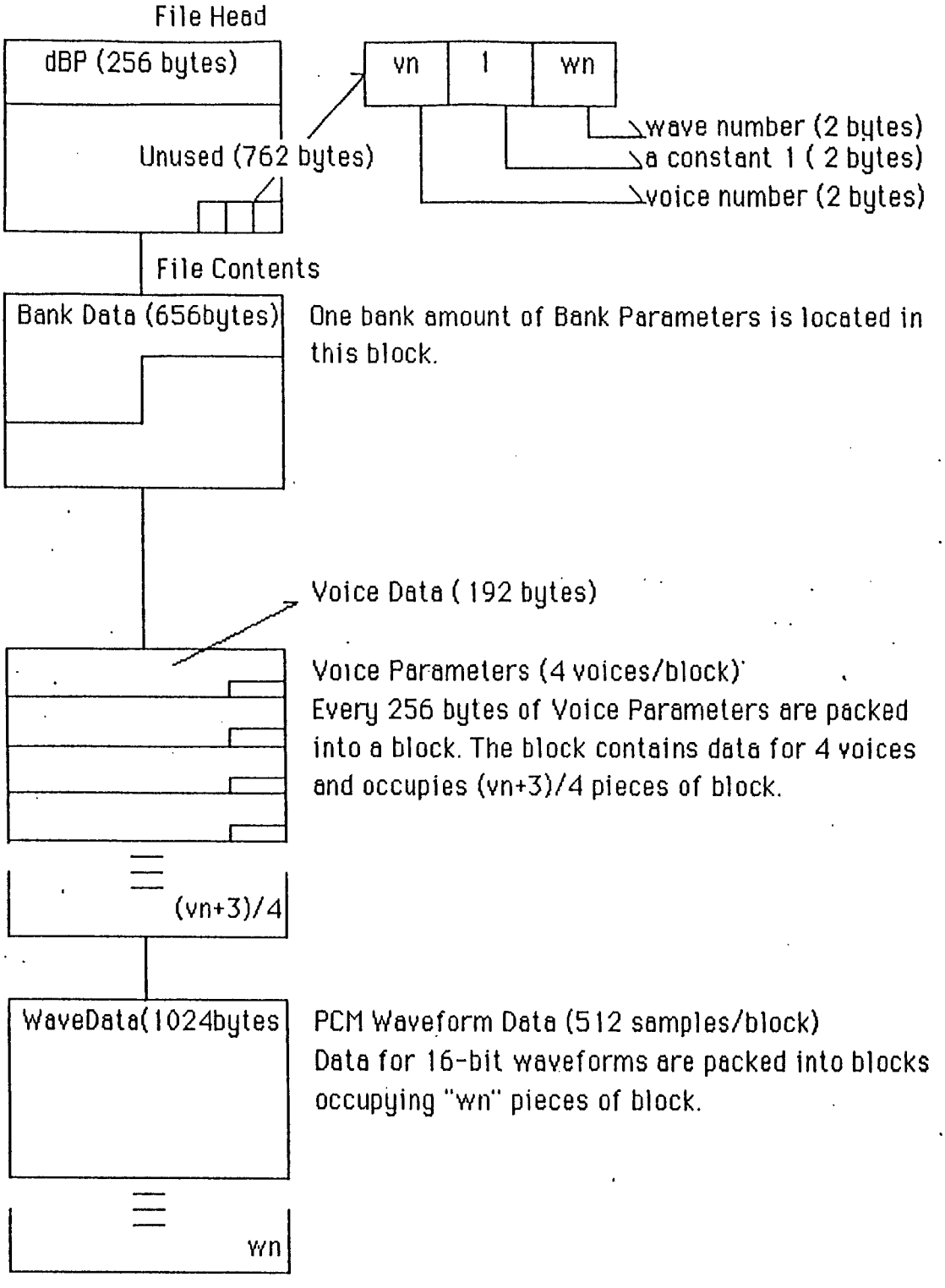
Voice Parameters (4 voices/block)
Every 256 bytes of Voice Parameters are packed into a block. The block contains data fof 4 voices and occupies (vn+3)/4 pieces of block.

PCM Waveform Data (512 samples/block)
Data for 16-bit waveforms are packed into blocks occupying "wn" pieces of block. One sample consists of 2-byte data and the upper byte is positioned in the higher address.

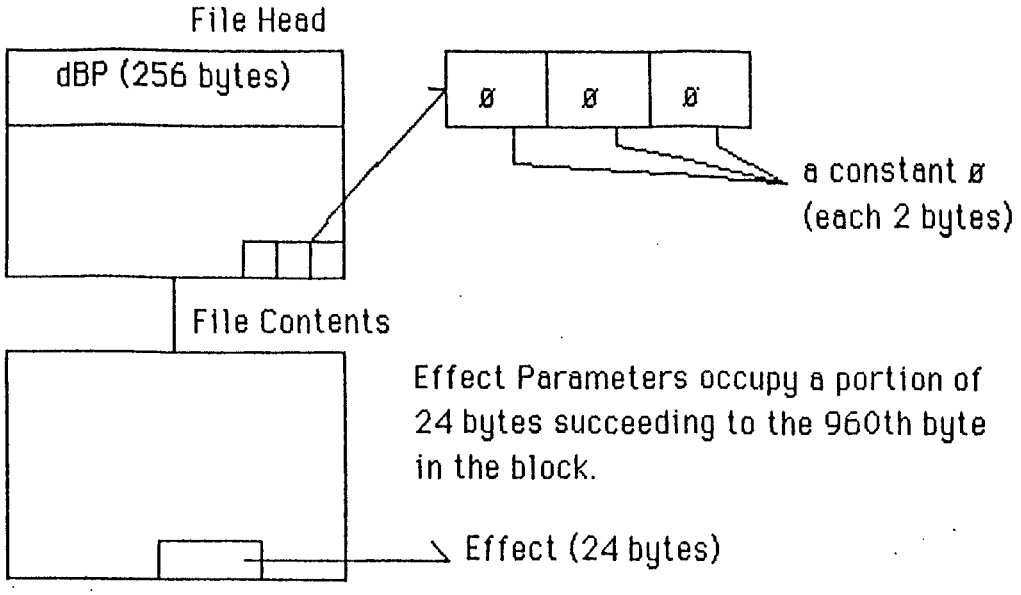
c) Voice Data ext=1



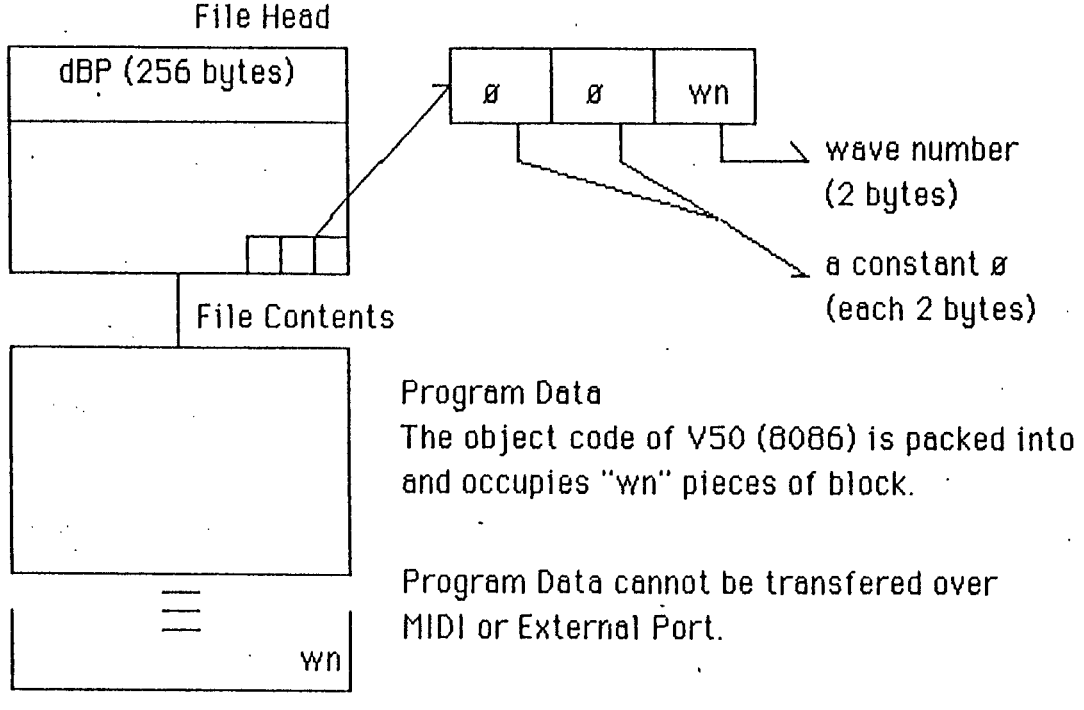
d) Bank Data ext=2



e) Effect Data ext=3

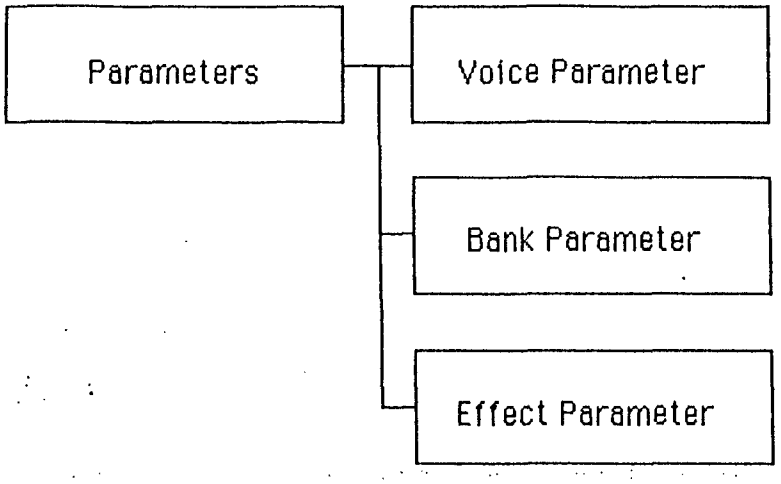


f) Program Data ext=5



2. Outline of "Parameters"

The FZ-1 is provided with 3 different parameters for Voice, Bank and Effect. This chapter will explain the details of each parameter.



2-1. Details of Voice

Voice data is an assemblage of the parameters which point an address on the PCM encoded waveform memory, an address of the loop, or determines the envelope curve corresponding directly with a PCM-sampled waveform or a synthesized waveform.

The size of parameters is 192 bytes and total data of the voice parameters occupies the area of 12,288 bytes maximum since the FZ-1 internal memory can contain up to 64 voices. In the work area voice parameters start with the label named "voic" to be addressed by every 192 bytes in the sequence from Voice 1 to Voice 64 . The content for these 192 bytes is shown in the listing "basic data structure define". It is defined as a structure of "C" of which name is "struct voicedata". The 2-digit hexadecimal numbers existing in the most right end of comment columns are the offset addresses when the header for voicedata is set to ø. The byte sizes for each parameter factor are

long:4 bytes, int: 2 bytes, short: 1 byte.

If a data requires multiple blocks, the byte size will be N times as big as this. (MAXE is a constant 8 in this case.) For each byte, a higher byte is positioned at the high adress.

List A: Basic Data Structure Define

```
/* ---------------- basic data structure define ------------------- */
struct voicedata {
  long               wavst;              /* wave start address                 00*/
  long               waved;              /*      end                           04*/
  long               genst;              /* generator start address            08*/
  long               gened;              /*              end                   0C*/
---------------------------------------------------------------------------------
  int                loop;               /* ga mode status (see gaa)           10*/
  short              loop_sus;           /* loop sustain number (0-8)          12*/
  short              loop_end;           /* loop end     number (0-8)          13*/
  long               loopst[MAXE];       /* loop start address                 14*/
                                         /*  --- b15~b12 for loop fine          */
  long               looped[MAXE];       /* loop end   address                 34*/
                                         /*  --- b15 for jumploop flg           */
  int                loopxf[MAXE];       /* loop x feed time                   54*/
  unsigned int       looptm[MAXE];       /* loop time ( or times)              64*/
---------------------------------------------------------------------------------
  int                dcp;                /* dcp voice pitch with detune74*/
  short              dcf;                /* frequency offset value             76*/
  short              dcq;                /* filter Q  offset value             77*/
---------------------------------------------------------------------------------
  short              dca_sus;            /* dca envelop sustain point  78*/
  short              dca_end;            /* dca envelop end point              79*/
  short              dca_rate[MAXE];     /* dca envelop rate value             7A*/
  unsigned short     dca_stop[MAXE];     /* dca envelop stop value             82*/

  short              dcf_sus;            /* dcf envelop sustain point          8A*/
  short              dcf_end;            /* dcf envelop end point              8B*/
  short              dcf_rate[MAXE];     /* dcf envelop rate value             8C*/
  unsigned short dcf_stop[MAXE];         /* dcf envelop stop value             94*/
---------------------------------------------------------------------------------
  unsigned int       lfo_delay;          /* lfo delay time                     9C*/
  unsigned short     lfo_name;           /* lfo wave form define (b7/8 SYNC ON) 9E*/
  unsigned short     lfo_atck;           /* lfo attack value                   9F*/
  short              lfo_rate;           /* lfo rate (time increment)          A0*/
  short              lfo_dcp;            /* lfo pitch depth                    A1*/
  short              lfo_dca;            /* lfo amp   depth                    A2*/
  short              lfo_dcf;            /* lfo filter depth                   A3*/
  short              lfo_dcq;            /* lfo fileter Q depth                A4*/
  short              vel_dcq_kf;         /* initial touch dcq follow           A5*/
---------------------------------------------------------------------------------
  short              dca_kf;             /* dca keyboard follow depth          A6*/
  short              dca_rs;             /* dca noterate scaling depth         A7*/
  short              dcf_kf;             /* dcf keyboard follow depth          A8*/
  short              dcf_rs;             /* dcf noterate scaling depth         A9*/

  short              vel_dca_kf;         /* initial touchamp key followAA*/
  short              vel_dca_rs;         /* initial touchamp rate scaleAB*/
  short              vel_dcf_kf;         /* initial touchdcf key followAC*/
  short              vel_dcf_rs;         /* initial touchdcf rate scaleAD*/
---------------------------------------------------------------------------------
  unsigned short hwid;                   /* high width MIDI code               AE*/
  unsigned short lwid;                   /* low                                AF*/
  unsigned short cent;                   /* keynote centor                     B0*/
---------------------------------------------------------------------------------
  unsigned short samp;                   /* sampling frequency                 B1*/
---------------------------------------------------------------------------------
  char name[14];                         /* wave name                          B2*/
};                                       /* total byte --- 0C0                   */
```

wavst, waved - Shows the head and end addresses of a PCM encoded waveform data for a designated voice by a Word Address.
wavst≤waved

genst, gened - Shows the head and end addresses of the sounding range for a designated voice by a Word Address.
wavst≤genst≤gened≤waved

loop - Assigns sounding styles for a designated voice.
ox0000: NO SOUND (Waveform not yet defined)
ox01D7: NORMAL (Normal sound)
ox101D: REV (Reversed sound)
ox2014: CUE (Cuing sound)
ox0013: SYN (Synthesized waveform)

loop sus - Assigns the position of Sustain Loop. A number from ø thru 8 can be taken and ø denotes Loop 1 execution. The number ø thru 7 assigns the corresponding loop execution, and 8 denotes no execution of Sustain Loop.

loop end - Assigns the end of multi-loop. A number from ø thru 8 can be taken and the number ø denotes the end of loop 1, and the number ø thru 7 assigns the corresponding loop end. The number 8 assigns execution of all the 8 loops.

loopst, looped - Shows the head and end addresses for a looping range by a Word Address. In correspondence with 8 multi-loops, eight sets of the head and the end addresses are provided. Upper 8 bits for loopst are used for loop fine and take a number among ø - 255.
The MSB for looped is used for loop patterns; 1 for Skip, ø for Trace.
wavst≤genst≤loopst≤looped≤gened≤waved

loopxf - Shows a timing duration for Cross Fade Loop and takes a number among ø - 1023. The number ø designates a minimal distorted sound from artificial data in between samples. Place a figure ø for non-cross fade looping.

looptm - Denotes a timing duration for Multi Loop and takes a number among 1 - 1022. The duration can be set by 16 mili seconds from 16 mili seconds up to 16 seconds.

dcp - Denotes a pitch. The pitch can be corrected by 1/256 semi-tone by a sound range setting.

dcf - Denotes an offset value for Cut Off Frequency on the filter and takes a number among ø - 127. The frequency will never be lowered than the value set in dcf and the value 127 designates the filter should open.

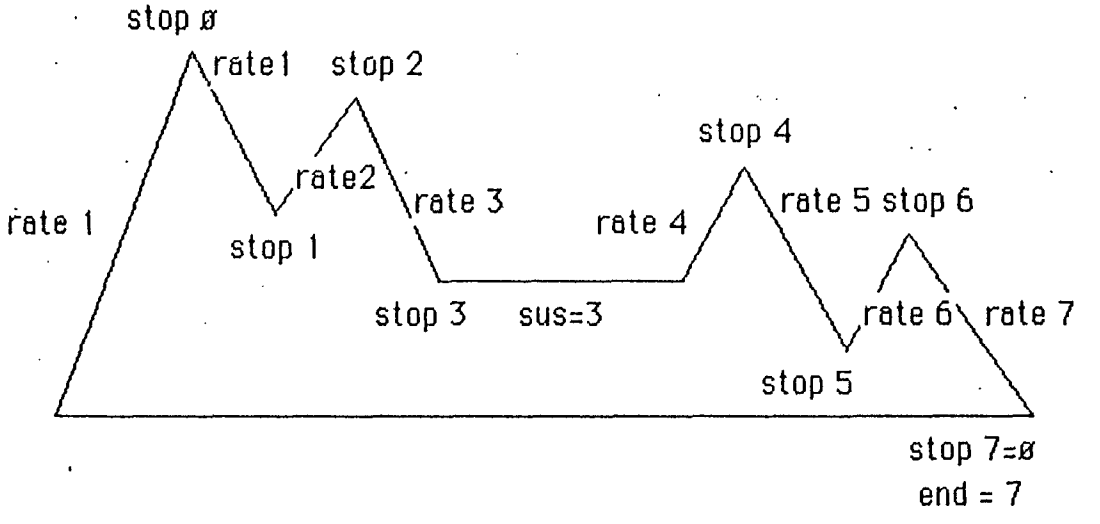
dcq - Denotes an offset value for Resonance on the filter and takes a number among ø - 127; however, notice that the effective bit number is upper 4 bits.

dca sus, dcf sus - Denote Sustain positions on each envelope on Amp and Filter and take numbers among ø - 7.

dca end, dcf end - Denote the end point for an envelope and take a number among ø - 7.

dca rate, dcf rate - Denote slopes for an envelope curve. The lower 7 bits will be a number among ø - 127; an absolute value. The MSB denotes a slope; ø for plus and 1 for minus.

dca stop, dcf stop - Denote an arrival value for each step of an envelope curve and take a number ø - 255.



lfo delay - Denotes a time duration before LFO starts affecting sound and takes a number among ø - 65535. The LFO delay can be set by 2 mili seconds.

lfo name - Denotes the LFO waveform names:
ø: Sine Wave
1: Ascending Saw-Tooth
2: Descending Saw-Tooth
3: Triangle
4: Rectangular
5: Random
The MSB denotes On or Off for phase synchronization; ø for Off and 1 for On.

fo atck - Denotes a rising envelope rate for the LFO effect and takes a number among 1 - 127. A smaller number denotes slower and a bigger number denotes faster.

fo rate - Denotes a frequency for the LFO and takes a number among ø -127.

fo dcp - Denotes a depth of LFO effect on the pitch and takes a number among ø - 127.

fo dca - Denotes a depth of LFO effect on the amplitude and takes a number among ø - 127.

fo dcf - Denotes a depth of LFO effect on the filter and takes a number among ø - 127.

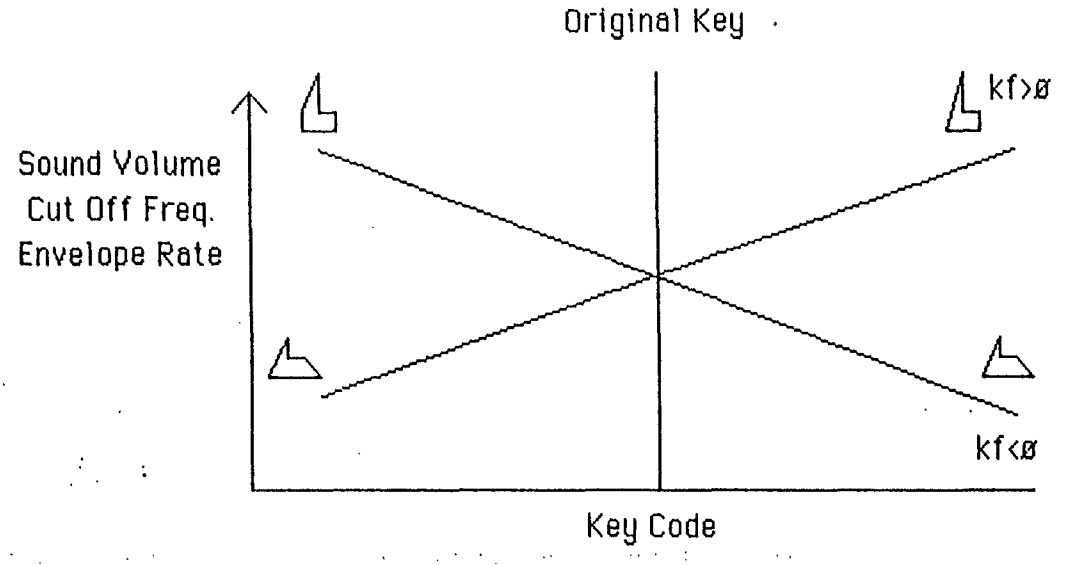
fo deq - Denotes a depth of LFO effect on the resonance and takes a number among ø - 127.

dca kf - Denotes a key follow effect on the amplitude and takes a number among -127 - +127. Centering the original key, "+" assigns upper right and "-" assigns lower right tilt for sound volume.

dca rs - Denotes a rate follow effect for an Amp Envelope and takes a number among -127 - +127. Centering the original key, "+" assigns the sharper rate the higher key and "-" assigns the shallower rate the higher key.

dcf kf - Denotes a key follow parameter on the filter.

dcf ks - Denotes a rate follow effect for a Filter Envelope.



vel dca kf - Denotes a degree against amplitude made by the initial touch response and takes a number among -127 thru +127. A plus (+) number assigns the higher velocity generates the bigger volume; a minus (-) number assigns the lower velocity generates the bigger sound volume.

vel dca rs - Denotes an effect rate against an envelope curve made by the initial touch response and takes a number among -127 thru +127. A plus (+) number assigns the higher velocity generates the sharper curve; a minus (-) number assigns the higher velocity generates the gentler slope.

vel dcf kf - Denotes a filter effect made by the initial touch response.

vel dcf rs

hwid, lwid, cent	- Denote the highest and lowest limitations for a sounding range and the key code for an original sample. Take numbers among ø thru 127 and have the same note code for keyboard positions as the MIDI standard has.
samp	- Denotes a sampling frequency when you used for recording a material and takes a number among ø - 2; ø for 36k Hz, 1 for 18k Hz, and 2 for 9k Hz.
name	- Shows a voice name with 12 ASCII-coded characters. A voice name occupies a 14-byte region and the last 2 bytes should be always ø.

2-2. Details of Bank

Bank data is an assemblage of the parameters which make up a keyboard setting as an actual instrument with key range settings of key split, velocity split, touch response, and/or MIDI basic channel. The size of the parameters is 656 bytes and the total data of the bank parameters occupies 5,248 bytes maximum as the FZ-1 internal memory can store up to 8 banks. The bank parameters start with the label "bank" in the work area to be addressed by every 656 bytes in the sequence from bank 1 to bank 8.

This content of 656-byte data is shown in the list B. Same as the voice data, it is defined as a structure of "C" and the size for every factor is also the same. MAXV is a constant 64 in the format.

List B: Bank Data Structure

```
          HP 64000 - C          70116 compiler
/*-----------------------bank-data-structure----------------------
struct bankdata {
  unsigned int    bstep;          /* bank use voice number        00
  unsigned short  hwid[MAXV];     /* high keynote width           02
  unsigned short  lwid[MAXV];     /* low                          42
  unsigned short  htch[MAXV];     /* high keytouch width          82
  unsigned short  ltch[MAXV];     /* low                          C2
  unsigned short  cent[MAXV];     /* keynote centor               102
  unsigned short  mchn[MAXV];     /* generate midi channel        142
  unsigned short  gchn[MAXV];     /* generate channel select      182
  unsigned short  bvol[MAXV];     /* generate outptu level        1C2
  unsigned int    vp[MAXV];       /* voice data pointer           202
  char name[14];                  /* bank name                    282
  };                              /* --- total byte              290
#define BSIZE   (sizeof (struct bankdata))
```

bstep - Denotes the current number of key splits or the number of voices which the bank uses and takes a number among ø thru 64. The number ø denotes that the current bank is not yet defined.

hwid, lwid - Denotes the highest and lowest limitations for a key split region. The note code corresponds with that of MIDI. These limitations can be set independently from those for voice data.

htch, itch - Denotes the highest and lowest limitations for a velocity split. The code corresponds with the initial values in the MIDI standard and takes a number among 1 - 127.

cent - Denotes an original key position of a key split. The code corresponds with the note code of the MIDI and takes a number among ø - 127 and the center key can be set independently from voice setting.

mchn - Denotes a receiving channel for each area at the Area Mode and takes a number among ø - 15 corresponding with the MIDI basic channels 1 thru 16.

gchn - Denotes a sound generator for a designated area. These 8 bits corresponding with the generator 1 - 8 allow to generate sound when each bit is 1 and prohibit to do so when it is ø. The bit ø stands for the generator 1. For instance, sound from the area will be generated only by the generators 2 and 7 in the case gchn=42h.

bvol - Denotes sound volume for each area and takes a number among ø - 127. This enables to balance sound volumes among the voices which allocated to areas to make up a bank.

vp	- Is placed with the head address for voice parameters which are used for an area. Notice that the voice number among ø - 63 is positioned at these bits when the parameters are dumped from the internal memory to disk, or outside memory over MIDI or External Port.
name	- Shows a bank name with 12-byte ASCII-coded characters. The last 2 bytes should be always ø.

2-3. Details of Effect

Effect data is an assemblage of the effect parameters controlled by the pitch bender, the modulation wheel, the foot volume, the after touch, the face panel controls except keyboard keys. The size of the parameters is 24 bytes headed by *para* in the work area of which content is shown in the list C. The effect data is parameters commonly effective on all banks and all voices. These are defined as a structure of "C" and the size of every factor is all 1 byte.

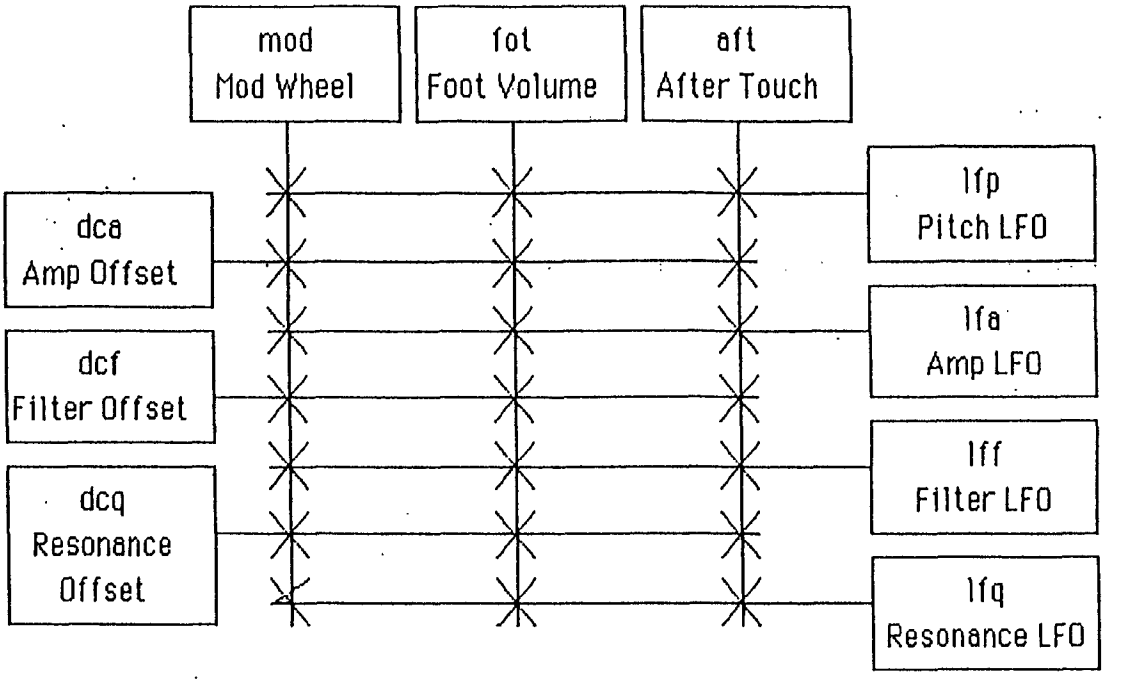
List C: Effect Data Structure

```
struct efectdata {
  short bend;                   /* bender depth                    00
  short mvol;                   /* master volume value             01
  short suss;                   /* sustain switch ON,OFF           02
-------------------------------------------------------------------------
  short mod_lfp;                /* modulation   lfo pitch          03
  short mod_lfa;                /*              lfo amp            04
  short mod_lff;                /*              lfo filter         05
  short mod_lfq;                /*              lfo filter q       06
  short mod_dcf;                /*              filter offset      07
  short mod_dca;                /*              amp    offset      08
  short mod_dcq;                /*              fil q  offset      09
-------------------------------------------------------------------------
  short fot_lfp;                /* foot volume  lfo pitch          0A
  short fot_lfa;                /*              lfo amp            0B
  short fot_lff;                /*              lfo filter         0C
  short fot_lfq;                /*              lfo filter q       0D
  short fot_dca;                /*              amp    offset      0E
  short fot_dcf;                /*              filter offset      0F
  short fot_dcq;                /*              fil q  offset      10
-------------------------------------------------------------------------
  short aft_lfp;                /* after touch  lfo pitch          11
  short aft_lfa;                /*              lfo amp            12
  short aft_lff;                /*              lfo filter         13
  short aft_lfq;                /*              lfo filter q       14
  short aft_dca;                /*              amp    offset      15
  short aft_dcf;                /*              filter offset      16
  short aft_dcq;                /*              fil q  offset      17
};                        /* total byte = 18                      */
#define ESIZE   (sizeof (struct efectdata))
```

bend - Denotes an effect degree of the Pitch Bender by a 1/8 semi-tone step and takes a number among ø - 127.

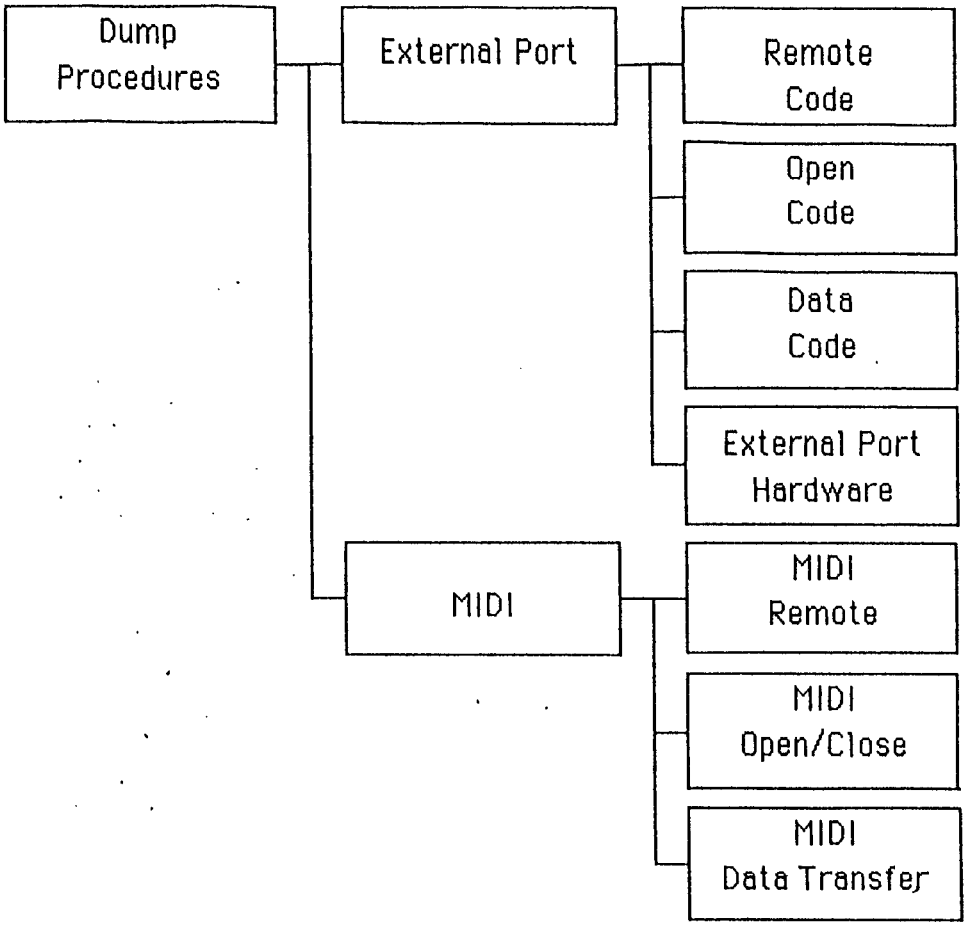
mvol, suss - Unused. Normally "ø" is placed.

mod, fot, aft / lfp, lfa dca, lff dcf, lfq dcq - Denotes an influential degree made by each controller and takes a number among ø - 127. These effects form a matrix as shown below:



3. Outline of "Dump"

The FZ-1 outputs or inputs data over the External Port or the MIDI ports. The data dump features will be explained in detail.



3-1. Procedures for Data Dump

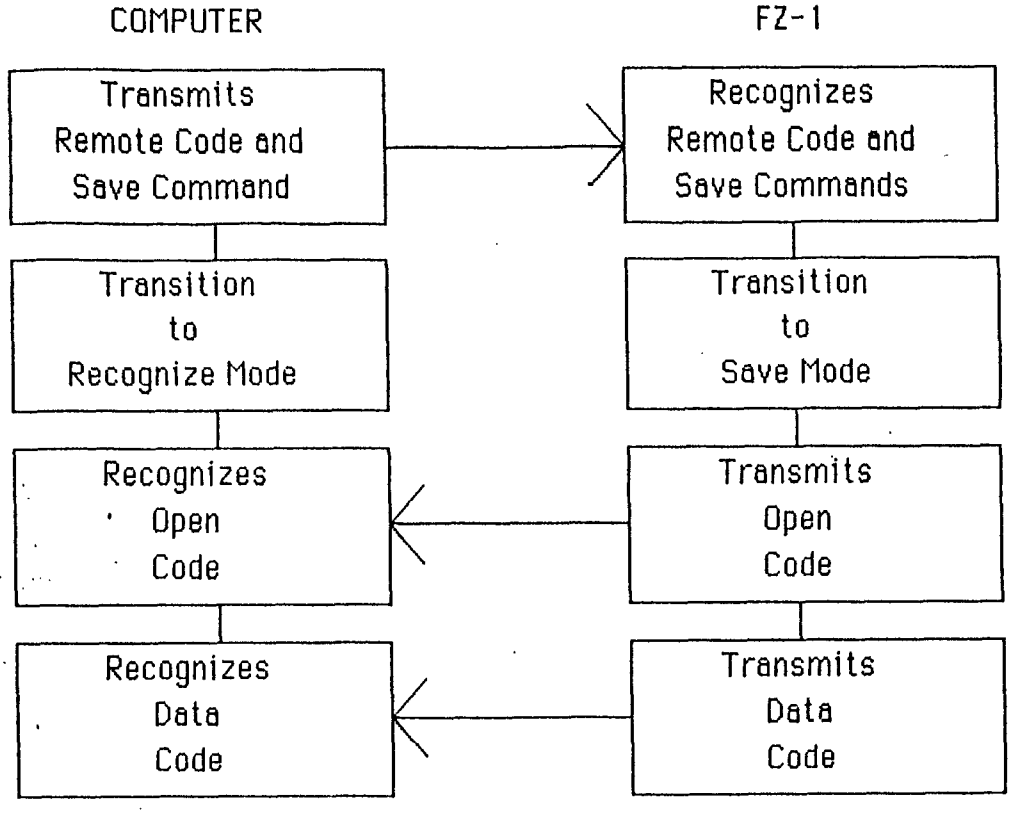
The FZ-1 unit has a capability of inputting and outputting data for waveforms, control parameters, etc. over an external port and MIDI ports as well as a capability of dumping data on micro floppy disks. The data transfer will be done by the following two methods:

a) From an FZ-1 unit to another FZ-1 unit:
1. Set the Master unit to the Save Mode
2. Set the Slave unit to the Load, Merge or Verify Mode
3. Set the both units to MIDI or to External Port and push the buttons "Enter" and "Yes" on the both units to save/load data onto the disk in the Slave unit. To transfer 1 Megabyte of data amount to another unit or computer, it will take 20 or 30 minutes over MIDI and 40 seconds over the External Port.

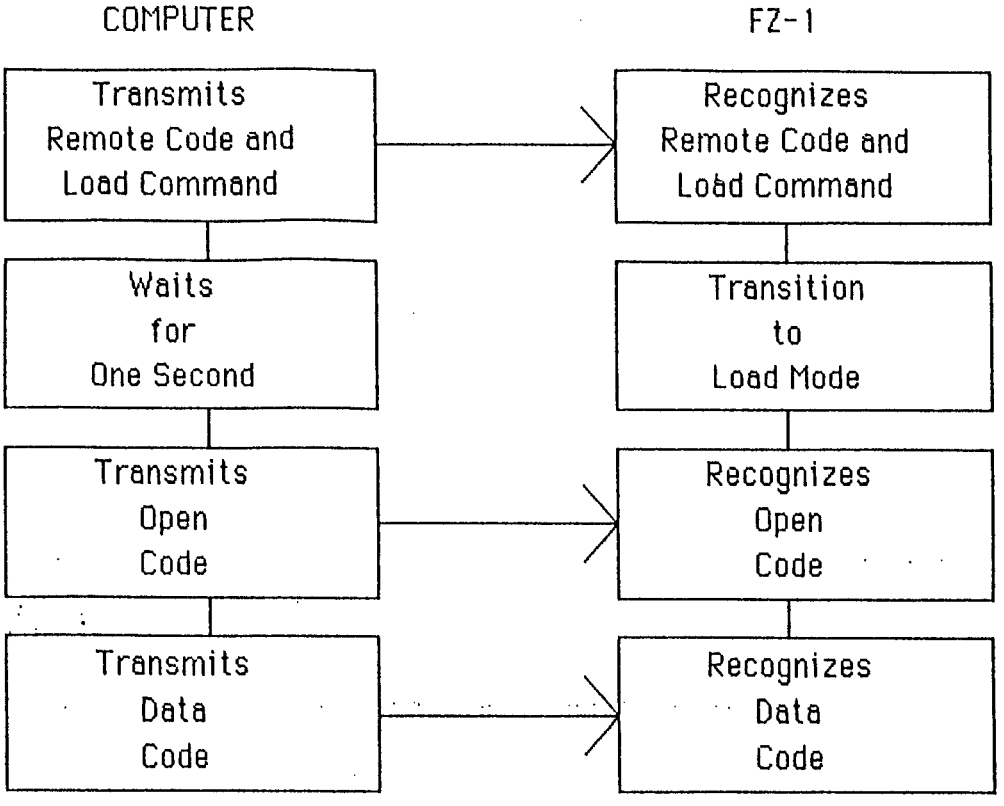
b) From an FZ-1 unit to a computer or vice versa:
An FZ-1 unit can be hooked up to a computer which is equipped with MIDI compatible ports or with the equivalent External Port to transfer each other waveform data and parameters. The device in the computer should be set to MIDI or the External Port and start up with a command from the computer the connected FZ-1 unit which is set to the Remote Mode. The Remote Mode implies a control of the FZ-1 unit, which is in the Save or Load Mode, from outside.

3-1-1. Outline of External Port

a) Data transfer from FZ-1 to a computer



b) Data Transfer from a Computer to FZ-1



Note: Merge and Verify are done in the same way as Load.

3-1-2. Details of Remote Code

The Remote Code, a 17-byte data block as illustrated below, is effective only for data transfer from a computer to an FZ-1 unit.

```
 ø                                                          16
[7F][  ][  ][  ][  ][  ][eb][ev][sta][mod][  ][  ][  ][  ][  ][  ][sum]
```

The mark [ ] denotes data of a byte and transfer will be done from the left to right. Blank bytes have no meaning and are normally filled with ø.

[7F]: Is a constant of hexadecimal 7F and denotes the header for the remote code. When an FZ-1 receives 7F in its waiting status, the unit recognizes the following as remote codes.

[eb]: Denotes Edit Bank, which will change the bank number while being edited inside an FZ-1 unit. The values from ø thru 7 stand for the banks 1 thru 8. The value 7F means no bank number change to stay the number unchanged as it is.

[ev]: Denotes Edit Voice, which will change the voice number while being edited inside an FZ-1 unit. The values ø thru 63 stand for the voices 1 thru 64. The value 7F means no voice number change to stay the number unchanged as it is.

[sta]: Designates the data which will be transmitted and the value will be among ø thru 3.

sta	Name	Data To Be Designated
ø	FULL	Entire data saved in the FZ-1 internal memory
1	VOICE	Data for voices, waveforms designated by "ev"
2	BANK	Data for banks, voices, waveforms designated by "eb"
3	EFFECT	Data for effects saved in the FZ-1 internal memory

[mod]: Designates a destination and a processing for the data which will be transmitted and the value will be among ø thru 3.

mod	Name	Destination	Processing
ø	SAVE	from FZ-1 to Computer	Transmit internal data
1	LOAD	from Computer to FZ-1	Load received data to internal memory
2	MERGE	from Computer to FZ-1	Merge received data with data in internal memory
3	VERIFY	from Computer to FZ-1	Compare and check received data with data existing in internal memory

[sum]: Denotes check sum. A value is placed to be complement for 2 after adding all data figures of 16 bytes (ø thru 15).

The FZ-1 in the Remote Mode changes its mode according to the designated codes, when the unit receives the afore-mentioned codes. The same mode changes occur for a linkage between 2 units of FZ-1 after the buttons "Enter" and "Yes" are depressed on the slave unit end.

3-1-3. Details of Open Code

The Open Code is a 17-byte data block determining a size of the following data code before execution of the data transfer. The 17-byte data block is illustrated as follows:

```
 ø                                                          16
[sta][1L][1H][bn][Vn][WL][WH][eb][ev][  ][  ][  ][  ][  ][  ][sum]
```

[sta]: Is a header designating the data which will follow to be transmitted like the [sta] byte for the Remote Code.

[1L][1H]: Is a 2-byte value for determining a block size of the data code.

[bn]: Determines the number of banks of which data will be included in the followingly transmitted data block. The value will be among ø thru 8 and "ø" denotes no bank data in the coming data block.

[Vn]: Determines the number of voices of which data will be included in the followingly transmitted data block. The value will be among ø thru 64 and "ø" denotes no voice data in the coming data block.

[WL][WH]: Determines the number of PCM-sampled waveforms of which data will be included in the succeedingly transmiteed data block. A data unit consists of 1024 bytes and 512 samples and fragments will be rounded up. "ø" denotes no PCM-sampled waveform data in the coming block.

[eb][ev][sum]: Denote the same as those of the Remote Code do.

The relations among data/parameters will be as follows:

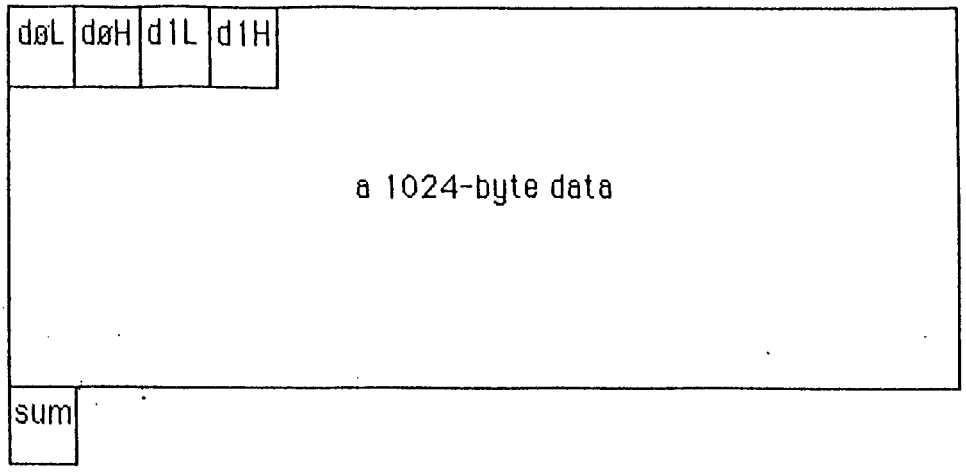
Name	Data Contents Indluded	sta	bn	vn	WHL
FULL	Entire data	ø	N	N	N
BANK	Bank parameters and voice parameters/PCM waveform data within the bank	2	1	N	N
VOICE	Voice parametes and PCM	1	ø	1	N
EFFECT	Effect parameters	3	ø	ø	ø
PARA	Entire parameters	ø	N	N	ø

"N" in the above table denotes a natural figure more than ø.

The transfer of only "PARA" is rarely executed and there is no possibility of the execution in the save command for data communication from an FZ-1 to a computer or to another FZ-1 unit. The "PARA" revises only exsisting parameters and transmits no waveform data. The "PARA" works effectively for the waveform data saved last in the internal memory. Address management should be done at the transmitting side, i.e., the management should be done at the connected computer which transmits the "PARA" data. The other Name-Data-Value combinations than the above table are prohibited to use; therefore, it cannot be done to revise nothing but the parameters for a particular bank or particular voice.

3-1-4. Details of Data Code

Data Code is a data entity being transmitted by every 1025 bytes (consisiting of a 1024-byte data and a 1-byte check sum) as a data unit.



In regard to the details of data, refer to the Outline of Parameters.
In regard to the way of packing into a 1024-byte data, refer to the Outline of Disk Format.

3-1-5. Details of the External Port - the hardware:

An FZ-1 machine is equipped with an External (Input/Output) Port and the machine sends and receives over the External Port data each for Full, Bank, Voice, Effect, Optional Application Program, Sequences, etc., utilizing its 8-bit bidirectional data port in the mode 2 and its another 8-bit data input port in the mode ø of a Programmable Parallel Interface Unit (PIU) consisting of a uPD71055 chip. Refer to the schematic diagram for the FZ-1 External I/O and the documentation of the NEC's MOS IC uPD71055.

a) Device: The uPD71055, a cMOS chip of NEC make, is completely compatible with an Intel's 8255 nMOS-type parallel interface chip. The uPD71055 has a set of 3 programmable 8-bit Input/Output ports (Port ø, Port 1 and Port 2) of which functions are described below:

Port ø: Is equipped with an independent 8-bit register for input and operates not only as an 8-bit unidirectional I/O port but also as an 8-bit bidirectional data port in its mode 2.

Port 1: Is equipped with an 8-bit register which operates either for input or output and works as an 8-bit I/O port.

Port 2: Is equipped with two registers which works as independent I/O ports against the divided upper 4-bit and lower 4-bit data. In the modes 1 and 2, this port functions for interrupts as well as peripheral control signals. For the purpose the Port 2 allows an operation for Set or Reset by a bit.

b) Operation: For transmitting data, the FZ-1 first confirms a sending-standby status ^O^B^F=High on its end and also a receiving-standby status ^B^U^S^Y=High on the end of a connected machine. Following the confirmation, the output data is sent to the uPD71055 to output only the signal ^S^T^B^O and does not transfer the very data to a bus.

The uPD71055 latches data into its buffer at the rising edge of a signal ^W^R and simultaneously lowers the signal ^O^B^F level. The signal ^O^B^F raises its level at a timing of the signal ^A^C^K drop. The FZ-1 machine which is ready to be input detects poling the INT signal fed from the uPD71055 and lowers the level of signals ^B^U^S^Y^O and ^A^C^K^O when confirming the INT signal leveled at High. By lowering the signal ^A^C^K^O

level to Low, the machine raises the signal "O"B"F on the output side at High in order to send out data to a bus.
After the data which outcoming to the bus is latched and held in, the machine finishes the transfer of one-byte data returning the signals "B"U"S"Y"O and "A"C"K"O to High level. See the Charts 1 and 2 for the examples of programming.

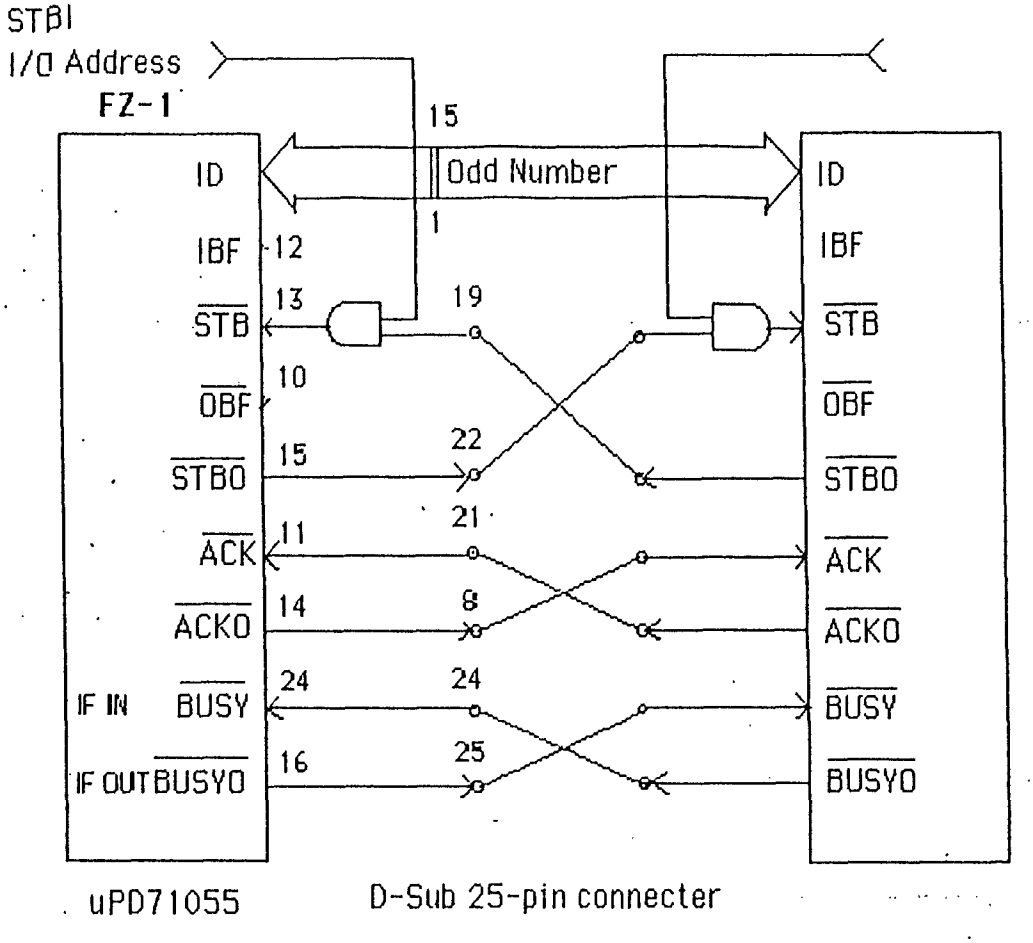
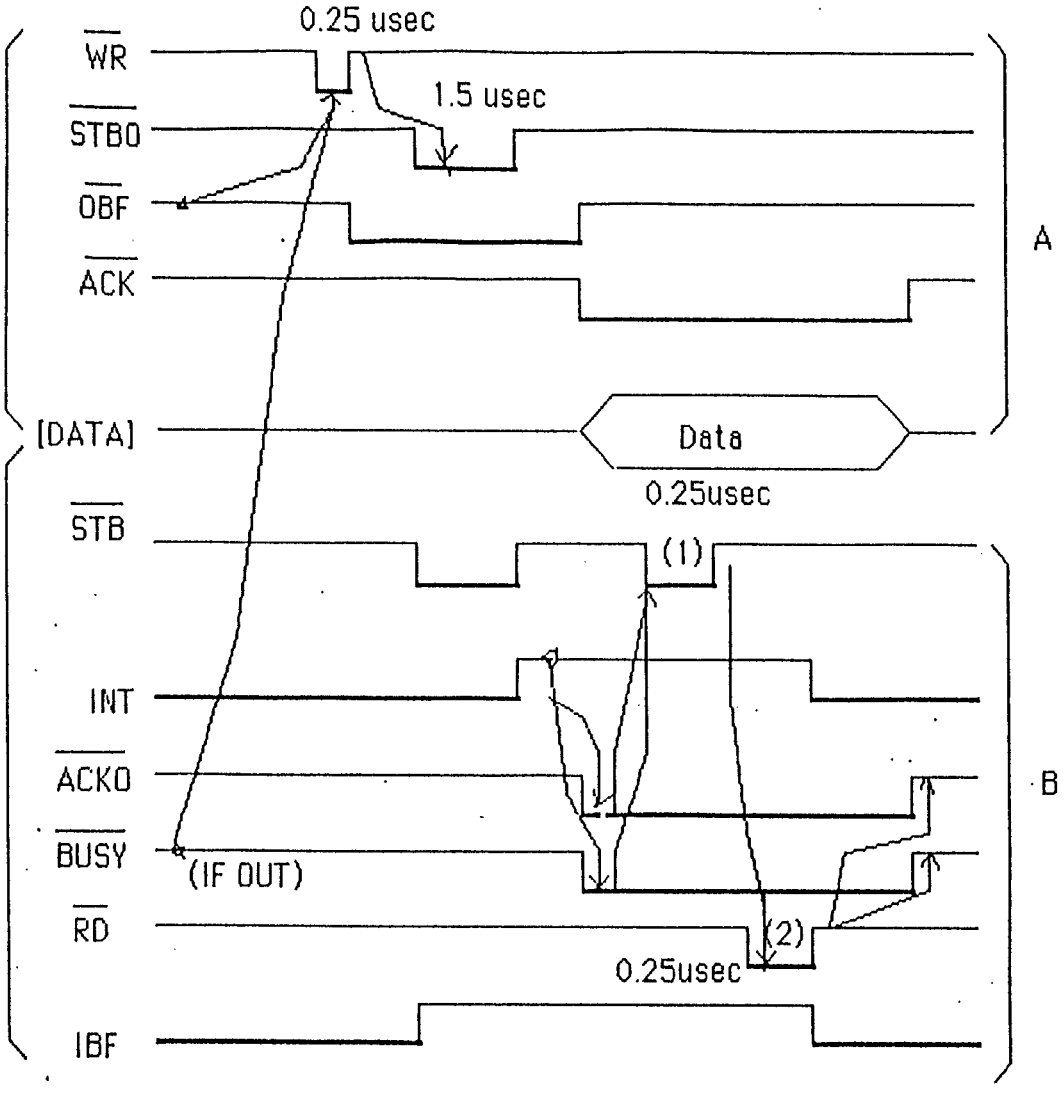


Chart 1



A: Master (Save) 1 Byte Transport
B: Slave (Load) 1 Byte Receive

Chart 2: External Port Read/Write Timing

c) Examples of Input and Output Programs:

c-1) Output Program:

```
portout   IN     AL, PIA12          ^O^B^F==H ?      Confirms a stand
          AND    AL, #+00080H       "   "            -by status for out
          BZ     SHORT Rportout                      put
          IN     AL, PIA11          ^B^U^S^Y==H ?    "     "
          AND    AL, #+00040H       "   "            "     "
          BZ     SHORT Rportout

          MOV    AL, CL             ;CL: Output Data
          OUT    101S, AL
          MOV    AL, #+0005H        ^S^T^B^O=L
          OUT    PIA12, AL          "     "
          MOV    AL, #+00007        ^S^T^B^O=H
          OUT    PIA12, AL          "     "
Rportout  RET
```

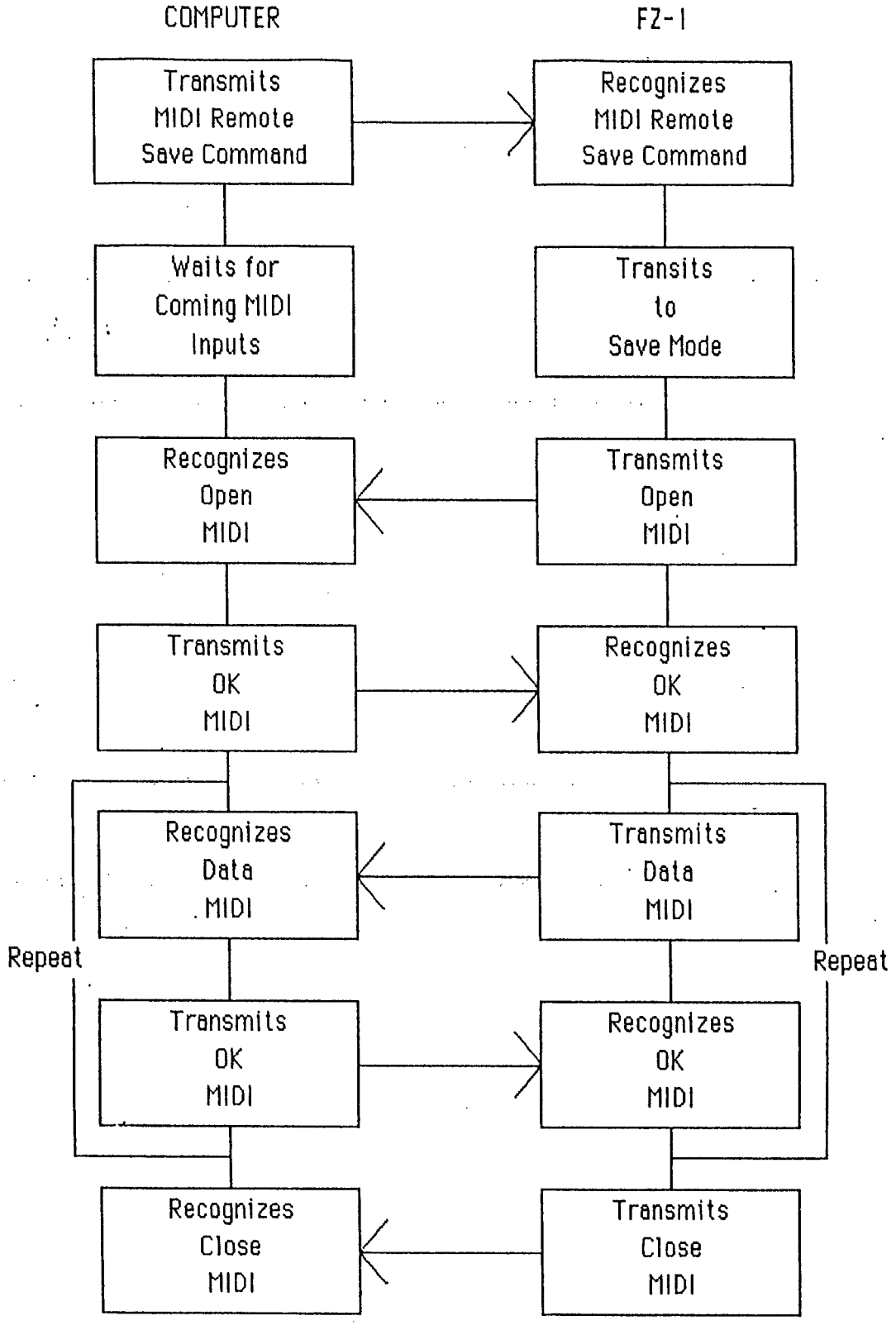
c-2) Input Program:

```
portin    IN     AL, PIA12          INT==H ?
          AND    AL, #+00008H       "   "
          BZ     SHORT Rportin

          MOV    AL, #+00002H       ^S^T^B^O&^A^C^K^O=L
          OUT    PIA12, AL          "     "     "

          OUT    STBI, AL           ;Latches data (AL:dummy)

          IN     AL, 101S           ;Inputs data to AL

          MOV    BL, AL             ;Stores data in BL

          MOV    AL, #+00007        ^S^T^B^O&^A^C^K^O=H
          OUT    PIA12, AL          "   "     "

Rportin   RET
```

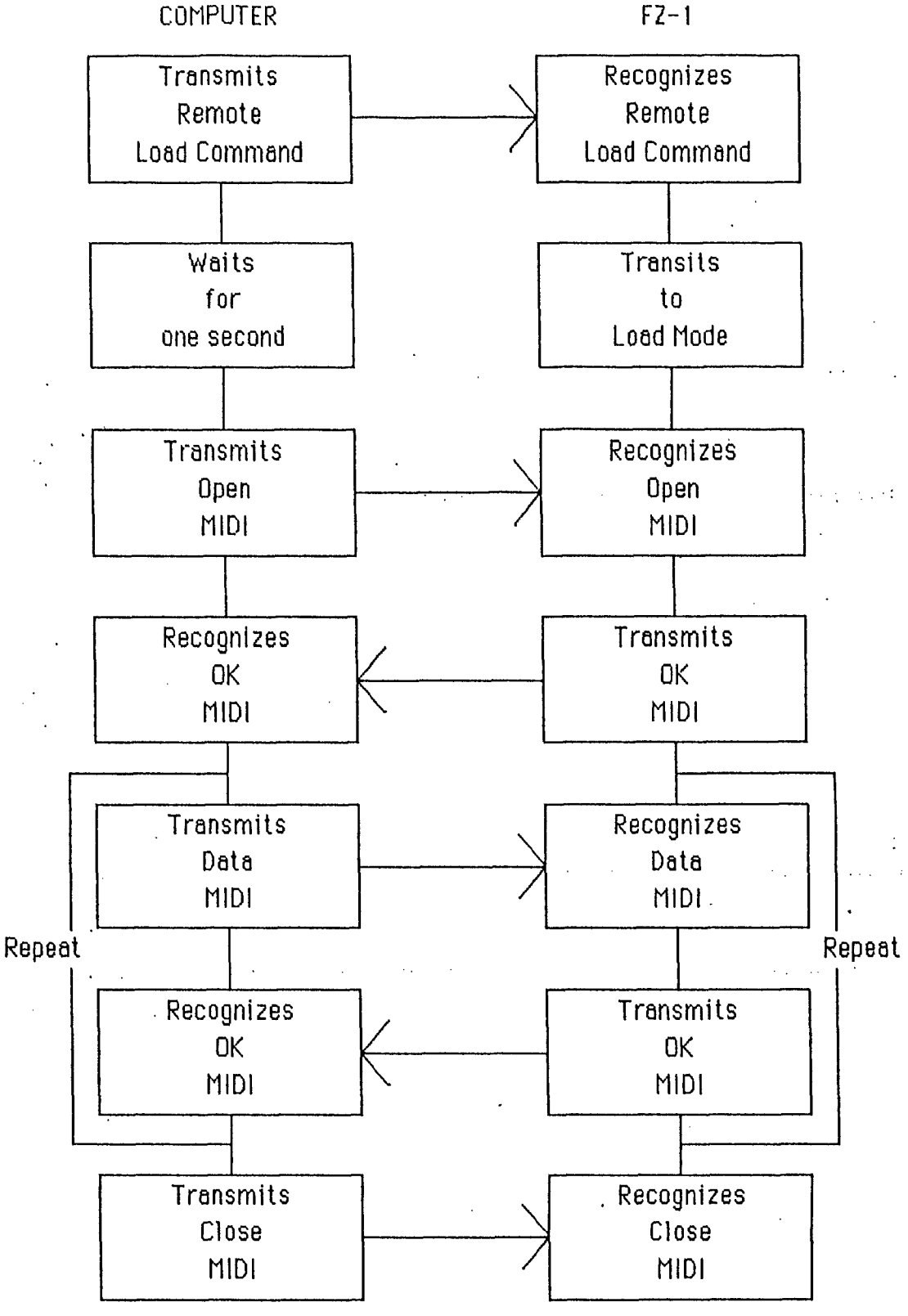
3-2. Outline of MIDI

The data transfer over MIDI is executed as follows.

a) Data transfer from an FZ-1 to a computer



b) Data transfer from a computer to an FZ-1



3-2-1. Details of Remote MIDI

The Remote MIDI is a MIDI System Exclusive Code which is exclusively provided for data transfer from a computer to an FZ-1 unit.

[Fø][44][ø2][øø][7n].....[7F].....[eb][ev]sta][mod].....[F7]

In this appendix the mark [ ] denotes a byte data and the transfer will be executed from the left to the right.

[7n]: For this byte, "n" for a Basic Channel number is to be placed in the lower 4-bit portion, and 7 is to be placed in the upper 4-bit portion. This is used for selective remote control in the case of plurial data connections.

[eb]: Same as in the details of Remote Code
[ev]: " " "
[sta]: " " "
[mod]: " " "

3-2-2. Details of Open/Close MIDI

A) Details of Open MIDI

The Open MIDI is a MIDI Exclusive Code determining a size of the data code which will be transfered preceding MIDI data.

[Fø][44][ø2][øø][7n][7ø].....[sta].....[bn][Vn][øWø][øW1][øW2][øW3][eb][ev].....[F7]

[sta]: Status - Same as the details of Open Code
[bn]: Bank Number - " "
[vn]: Voice Number - " "
[øWø]: Wave Number - Determines the number of PCM-sampled waveforms within the transmitted data. A data unit consists of 1024 bytes (512 samples). The original value is of 2 bytes and developed by 4 bits into 4 bytes to output as a MIDI code. "W3" should be a higher bit.
[eb]: Edit Bank - Same as the details of Remote Code
[ev]: Edit Voice - " "

B) Details of OK MIDI

The OK MIDI is a MIDI Exclusive Data to be sent to the data transmitting end as an answer message to the code Open MIDI or Data MIDI.

[Fø][44][ø2][øø][7n].....[72].....[F7]

If a format or a check sum is wrong for the latest data which have been received, the message ERR MIDI is transmitted instead of OK MIDI. The code is as follows:

[Fø][44][ø2][øø][7n].....[73].....[F7]

Receiving the message ERR MIDI the data transmitting end will respond the following:

a) Against the Open MIDI code, the failure in Open will show the Data Error on its screen.

b) Against the Data MIDI, the machine will transmit again last data.

C) Details of Data MIDI

The data code is an entity of the data to be transmitted. The data will be developed into 2 bytes from a byte for transmit. The 64 bytes of data will be developed into 128 bytes for one-time transmit.

[Fø][44][ø2][øø][7n].....[74].....[ødL][ødH][ødL][.....].....[Msum].....[F7]

<—— 128 bytes ——>

[ødL][ødH]: "ø" is input in their upper 4-bit portions after one byte of data is developed into lower 4 bits and into upper 4 bits.

[Msum]: Denotes a check sum. The value comes from the logical AND function on "7F" and "a compliment for 2" of the total addition number for developed 128 bytes.

Same as Disk or Port, 1024 bytes of data are regarded as a data unit. For the transfer of this, a transaction between the Data MIDI and the OK MIDI repeats itself 16 times. Refer to the outline of Parameters for the details of Data and also refer to the outline of Disk for the way of packing data into 1024 bytes.

D) Details of Effect MIDI

[Fø][44][ø2][øø][7n].....[78].....[en][vv].....[F7]

"en" denotes an effect number:
The number øø for the bender depth, and the numbers ø1 and ø2 are unused.
"vv" denotes a value among the figures øø thru 7F.

en	lfo pitch	lfo amp	lfo filter	dca	dcf
mod w	ø3	ø4	ø5	ø7	øB
foot v	øA	øB	øC	øE	øF
after t	11	12	13	15	16

Note: Exclusive info for Effect is transmitted the same way as the Control Commands.

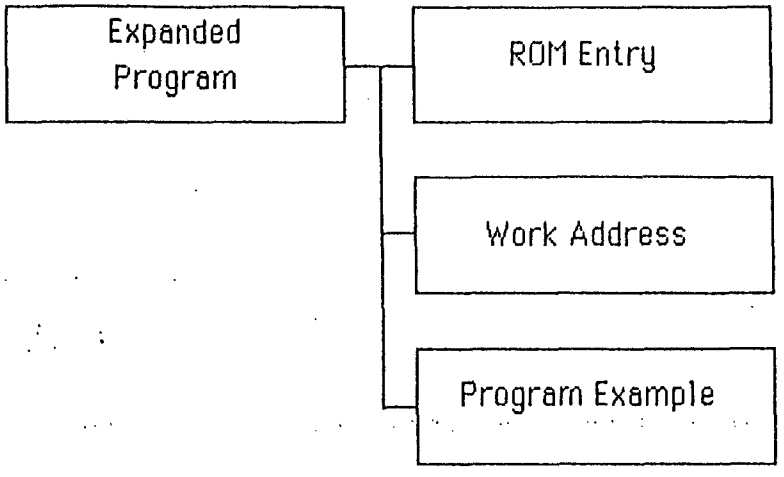
E) Details of Close MIDI

The Close MIDI is a MIDI System Exclusive Data which will be transmitted to the data receiver succeeding to the end of Data MIDI transfer.

[Fø][44][ø2][øø][7n].....[71].....[F7]

4. Optional Software

The FZ-1 is capable of loading expanded software and executing the program. This chapter offers specific knowledge on the FZ-1 to developers of optional software.



4-1. Expanded Program

The FZ-1 features that the data in a program file (ext=5) on its disk can be loaded to an address 6000h and after in the memory of the CPU work area and the FAR CALL to the address 6000h can be executed.

The FZ-1 installs a V50 chip (of NEC make) for the CPU. Since the V50 is upper compatible at the code level with Intel's 8086, you can develop programs on the chip 8086, a much more popular micro processor.

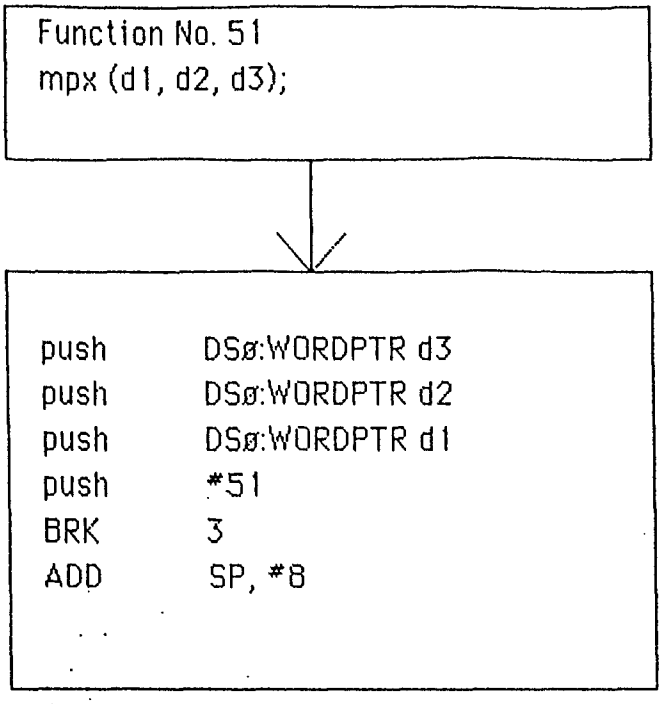
The tools necessary for development of programs will be:
a) Assembler and compiler for V50 (or 8086)
b) Conversion tool from Object File to FZ-1 Program File

For the details of FZ-1 Program File, refer to the Outline of "Disk". Expanded programs should be created at 6000h for its execution address and the 36k byte area (6000h - EFFFh) can be used.

4-2. ROM Entry

The FZ-1 is well designed so that every sub-routine existing in the firmware can be utilized with Break. The sub-routines are named by function numbers. There exists two types of parameters for each sub-routine; some are given in stack and the others are given directly to the WORK AREA. A sub-routine which returns a value has always a value in BW (BX for the 8086). Registers which are retained at all the sub-routines are nothing but SP and BP. The segments except DS1 are retained.

An example of a sub-routine call is shown below:



Succeeding to the push to stack behind the parameters and the final push of the function number, the command BRK 3 execution makes it possible to call "mpx" the sub-routine within the ROM.

4-3. Work Address

The work addresses to be used for the FZ-1 will be shown in the list E. For details of each work, refer to the details of FZ-1 Work.

List E:

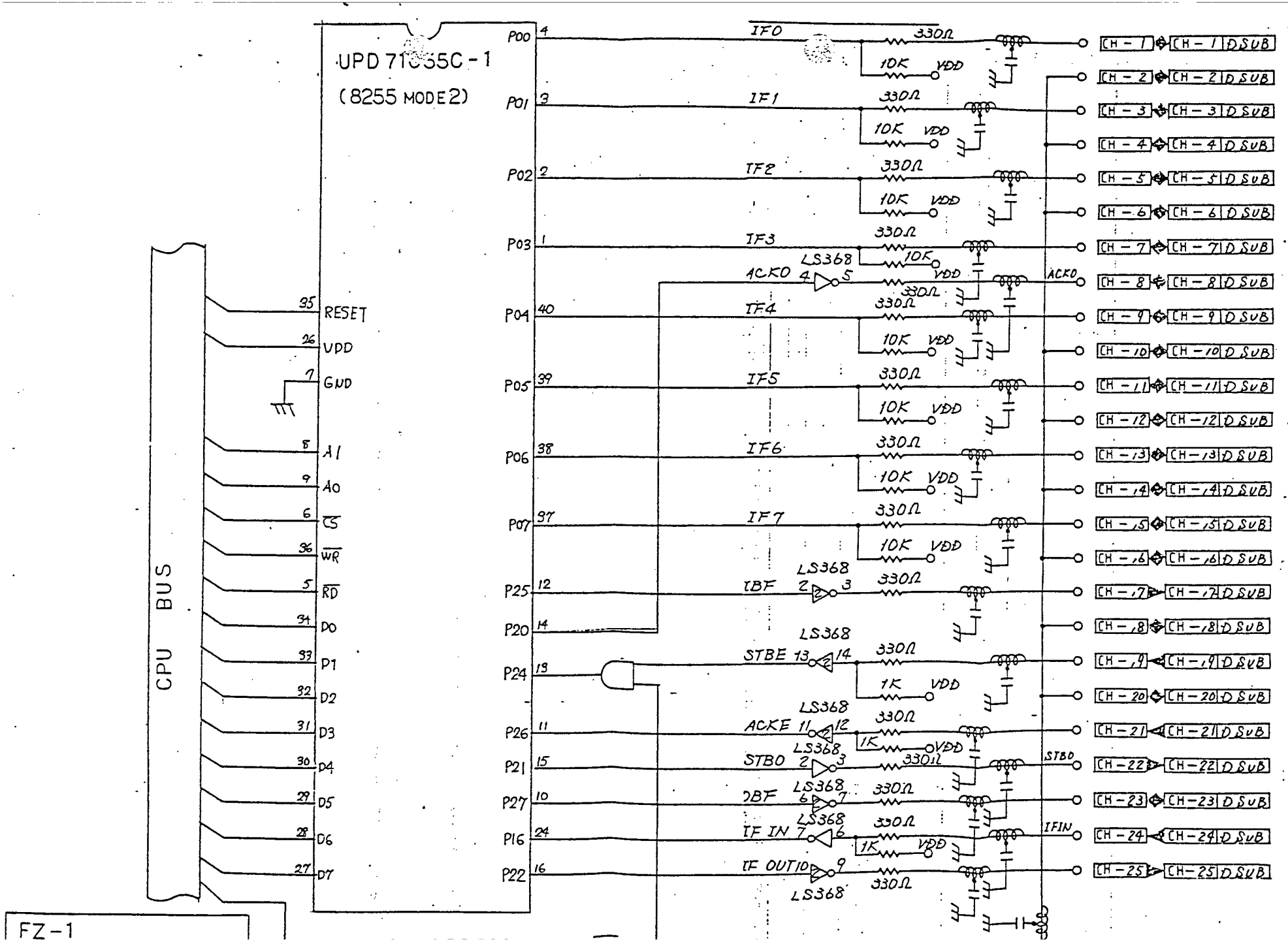
"70116"

DATA ORG 0400H

0400	keycount	DBS	1
0401	lastresp	DBS	1
0402	keymap	DBS	8
040A	sch	DBS	2
040C	olddca	DBS	16
041C	newdca	DBS	16
042C	key	DBS	1
042D	kkk	DBS	1
042E	kls	DBS	1
042F	sls	DBS	1
0430	ki0	DBS	4
0434	ki1	DBS	4
0438	rpc	DBS	2
043A	adc1	DBS	8
0442	adcb1	DBS	2
0444	env	DBS	1
0445	vol	DBS	1
0446	old	DBS	8
044E	max	DBS	8
0456	min	DBS	8
045E	cenh	DBS	1
045F	cen1	DBS	1
0460	stat	DBS	3
0463	par1	DBS	3
0466	nownote	DBS	2
0468	genbit	DBS	2
046A	lastiy	DBS	2
046C	excn	DBS	1
046D	nowled	DBS	1
046E	rand	DBS	2
0470	jump0	DBS	2
0472	lev	DBS	1
0473	lv0	DBS	3
0476	dm	DBS	1
0477	dm0	DBS	3
047A	sm	DBS	1
047B	sm0	DBS	3
047E	mm	DBS	1
047F	mm0	DBS	3
0482	lpos	DBS	2
0484	cpos	DBS	2
0486	lmax	DBS	2
0488	cmax	DBS	2
048A	loff	DBS	2
048C	ltop	DBS	2
048E	vpos	DBS	2
0490	posv	DBS	2
0492	posp	DBS	2
0494	vhi	DBS	2
0496	wid	DBS	4
049A	pos	DBS	16
04AA	grast	DBS	4
04AE	graed	DBS	4

04B2	pp1st	DBS	4
04B6	pp1ed	DBS	4
04BA	pp2st	DBS	4
04BE	pp2ed	DBS	4
04C2	mcount	DBS	2
04C4	mlevel	DBS	2
04C6	mpeek	DBS	2
04C8	mtrig	DBS	2
04CA	bb0	DBS	1
04CB	bb1	DBS	1
04CC	l_pos	DBS	4
04D0	l_cur	DBS	2
04D2	l_vhi	DBS	2
04D4	l_brk	DBS	2
04D6	trig	DBS	1
04D7	rmod	DBS	1
04D8	gain	DBS	1
04D9	rate	DBS	1
04DA	length	DBS	2
04DC	sintable	DBS	48
050C	add_v1	DBS	1
050D	add_v2	DBS	1
050E	add_l1	DBS	1
050F	add_l2	DBS	1
0510	add_t1	DBS	1
0511	add_t2	DBS	1
0512	add_dly	DBS	4
0516	add_xs1	DBS	4
051A	add_xs2	DBS	4
051E	devnum	DBS	2
0520	restat	DBS	2
0522	remode	DBS	2
0524	cat	DBS	160
05C4	dloc	DBS	2
05C6	xysheet	DBS	768
08C6	voice_num	DBS	2
08C8	bank_num	DBS	2
08CA	wave_num	DBS	2
08CC	cnv_sta	DBS	2
08CE	cnv_pos	DBS	2
08D0	cnv_rp	DBS	4
08D4	memsize	DBS	2
08D6	mi	DBS	260
09DA	mo	DBS	260
0ADE	kb	DBS	260
0BE2	si	DBS	260
0CE6	so	DBS	260
0DEA	midirev	DBS	1
0DEB	midisnd	DBS	1
0DEC	midimsk	DBS	1
0DED	midiprg	DBS	1
0DEE	seq	DBS	2
0DF0	godtime	DBS	4
0DF4	oldtime	DBS	4
0DF8	tempo	DBS	2
0DFA	mastertune	DBS	2

LOCATION	OBJECT CODE	LINE	SOURCE LINE
0DFC	pbn	DBS	2
0DFE	pb	DBS	4
0E02	evn	DBS	2
0E04	ev	DBS	4
0E08	bank	DBS	5248
2288	voic	DBS	12288
5288	pare	DBS	24
52A0	nowe	DBS	384
5420	gene	DBS	464
55F0	psa	DBS	2
55F2	pca	DBS	2
55F4	pwa	DBS	2


UPD 71055C-1
(8255 MODE2)
CPU BUS
35 RESET
26 VDD
7 GND
8 A1
9 A0
6 CS
36 WR
5 RD
34 D0
33 D1
32 D2
31 D3
30 D4
29 D5
28 D6
27 D7
P00 4
P01 3
P02 2
P03 1
P04 40
P05 39
P06 38
P07 37
P25 12
P20 14
P24 19
P26 11
P21 15
P27 10
P16 24
P22 16
IF0
IF1
IF2
IF3
IF4
IF5
IF6
IF7
ACKO
IBF
STBE
ACKE
STBO
OBF
IF IN
IF OUT
LS368
330Ω
10K
1K
VDD
CH-1 D SUB
CH-2 D SUB
CH-3 D SUB
CH-4 D SUB
CH-5 D SUB
CH-6 D SUB
CH-7 D SUB
CH-8 D SUB
CH-9 D SUB
CH-10 D SUB
CH-11 D SUB
CH-12 D SUB
CH-13 D SUB
CH-14 D SUB
CH-15 D SUB
CH-16 D SUB
CH-17 D SUB
CH-18 D SUB
CH-19 D SUB
CH-20 D SUB
CH-21 D SUB
CH-22 D SUB
CH-23 D SUB
CH-24 D SUB
CH-25 D SUB
ACKO
STBO
IFIN
FZ-1

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

     ##     ####     ##     #     #     #
    #  #    #       #  #    #    #      #
   #    #   ####   #    #   ####        #
#  ######       #  ######   #  #        #
#  #    #  #    #  #    #   #   #       #
   #    #   ####   #    #   #    #      #

t id: lp-4106    Printer: lp

r 17 18:34:21 JST 1987
```

FZ-1 WORK


```
N OBJECT CODE LINE     SOURCE LINE

                         1 "70116"              --- means "as above"
                         2          DATA
                         3       ORG    0400H
0                        4 keycount      DBS          1       ; numbers of key_on
1                        5 lastresp      DBS          1       ; the last touch response vaule 1~127
2                        6 keymap        DBS          8       ; key on/off table
A                        7 sch           DBS          2       ; big timer counter
C                        8 olddca        DBS         16       ; generater data
C                        9 newdca        DBS         16       ; ---
C                       10 key           DBS          1       ; console key code
D                       11 kkk           DBS          1       ; ---
E                       12 kls           DBS          1       ; ---
F                       13 sls           DBS          1       ; ---
0                       14 ki0           DBS          4       ; ---
4                       15 ki1           DBS          4       ; ---
8                       16 rpc           DBS          2       ; repeat counter for console key
A                       17 adc1          DBS          8       ; adc() static
2                       18 adcb1         DBS          2       ; ---
4                       19 env           DBS          1       ; ad convert vaule of line or mic in
5                       20 vol           DBS          1       ; ad convert value of entry volume
6                       21 old           DBS          8       ; last ad convert value
E                       22 max           DBS          8       ; max ad value
6                       23 min           DBS          8       ; min ad value
E                       24 cenh          DBS          1       ; center high limit for bender
F                       25 cenl          DBS          1       ; center low limit for bender
0                       26 stat          DBS          3       ; midi status byte
3                       27 par1          DBS          3       ; midi first data byte
6                       28 nownote       DBS          2       ; last MIDI key code and response
8                       29 genbit        DBS          2       ; generater bit num
A                       30 lastiy        DBS          2       ; last generater pointer
C                       31 excn          DBS          1       ; exclusive midi data counter
D                       32 nowled        DBS          1       ; now led
E                       33 rand          DBS          2       ; random number for lfo generate
0                       34 jump0         DBS          2       ; con()'s static
2                       35 lev           DBS          1       ; ---
3                       36 lv0           DBS          3       ; ---
6                       37 dm            DBS          1       ; ---
7                       38 dm0           DBS          3       ; ---
A                       39 sm            DBS          1       ; ---
B                       40 sm0           DBS          3       ; ---
E                       41 mm            DBS          1       ; ---
F                       42 mm0           DBS          3       ; ---
2                       43 lpos          DBS          2       ; para_change() parameter
4                       44 cpos          DBS          2       ; ---
6                       45 lmax          DBS          2       ; ---
8                       46 cmax          DBS          2       ; ---
A                       47 loff          DBS          2       ; ---
C                       48 ltop          DBS          2       ; ---
E                       49 vpos          DBS          2       ; ---
0                       50 posv          DBS          2       ; graph() parameter
2                       51 posp          DBS          2       ; ---
4                       52 vhi           DBS          2       ; ---
6                       53 wid           DBS          4       ; ---
```

```
ON OBJECT CODE LINE     SOURCE LINE

32                  57 pp1st       DBS        4       ; ---
36                  58 pp1ed       DBS        4       ; ---
3A                  59 pp2st       DBS        4       ; ---
3E                  60 pp2ed       DBS        4       ; ---
42                  61 mcount      DBS        2       ; meter() or jobbing() static
44                  62 mlevel      DBS        2       ; ---
46                  63 mpeek       DBS        2       ; ---
48                  64 mtrig       DBS        2       ; ---
4A                  65 bb0         DBS        1       ; brink() static
4B                  66 bb1         DBS        1       ; ---
4C                  67 l_pos       DBS        4       ; d_graph() static
50                  68 l_cur       DBS        2       ; ---
52                  69 l_vhi       DBS        2       ; ---
54                  70 l_brk       DBS        2       ; ---
56                  71 trig        DBS        1       ; recording trig level (0~255)
57                  72 rmod        DBS        1       ; recording mode
58                  73 gain        DBS        1       ; recording gain 0=L 1=H
59                  74 rate        DBS        1       ; sampling_rate_(0:36kHz,1:18kHz,2:9kHz)
5A                  75 length      DBS        2       ; recording length ( 10msec )
5C                  76 sintable    DBS       48       ; sin table for sin_synthesis
5C                  77 add_v1      DBS        1       ; source 1 voice # (0~63)
5D                  78 add_v2      DBS        1       ; source 2 voice # (0~63)
5E                  79 add_l1      DBS        1       ; source 1 mix level (0~255)
5F                  80 add_l2      DBS        1       ; source_2_mix_level_(0~255)
60                  81 add_t1      DBS        1       ; source 1 detune (-127~127)
61                  82 add_t2      DBS        1       ; source 2 detune (-127~127)
62                  83 add_dly     DBS        4       ; source 2 delay WORD address
66                  84 add_xs1     DBS        4       ; xmix start WORD address
6A                  85 add_xs2     DBS        4       ; xmix end   WORD address
6E                  86 devnum      DBS        2       ; device_number_(0:FDD,1:MIDI,2:PORT)
70                  87 restat      DBS        2       ; remote() static
72                  88 remode      DBS        2       ; ---
74                  89 cat         DBS      160       ; claster alocation table
74                  90 dloc        DBS        2       ; disk location counter
76                  91 xysheet     DBS      768       ; lcd graphic dot image
76                  92 voice_num   DBS        2       ; disk_subroutine's_static
78                  93 bank_num    DBS        2       ; ---
7A                  94 wave_num    DBS        2       ; ---
7C                  95 cnv_sta     DBS        2       ; ---
7E                  96 cnv_pos     DBS        2       ; ---
80                  97 cnv_rp      DBS        4       ; ---
84                  98 memsize     DBS        2       ; wave_memory_size_(*64Kbyte)
86                  99 mi          DBS      260       ; midi input ring buffer
8A                 100 mo          DBS      260       ; midi output ring buufer
8E                 101 kb          DBS      260       ; keyboard ring buffer
92                 102 si          DBS      260       ; sequencer input ring buffer
96                 103 so          DBS      260       ; sequencer output ring buffer
9A                 104 midirev     DBS        1       ; midi recieve_channel
9B                 105 midisnd     DBS        1       ; midi send channel
9C                 106 midimsk     DBS        1       ; midi_mask_status
9D                 107 midiprg     DBS        1       ; midi program change resister (-1:MASK)
9E                 108 seq         DBS        2       ; sequencer status
A0                 109 godtime     DBS        4       ; god time for sequencer
A4                 110 oldtime     DBS        4       ; old_time_for_sequencer
```

```
ON OBJECT CODE LINE     SOURCE LINE

FC              113 pbn         DBS             2       ; play bank number (0-7)
FE              114 pb          DBS             4       ; &bank[pbn]
02              115 evn         DBS             2       ; edit voice number (0-63)
04              116 ev          DBS             4       ; &voic[evn]
08              117 bank        DBS          5248       ; struct bankdata bank[8]
88              118 voic        DBS         12288       ; struct voicdata voic[64]
B8              119 pare        DBS            24       ; struct paradata para
A0              120 nowe        DBS           384       ; run time paradata nowe[16]
20              121 gene        DBS           464       ; run time generater data
F0              122 psa         DBS             2       ; rom entry static
F2              123 pca         DBS             2       ; ---
F4              124 pwa         DBS             2       ; ---

=     0
```

FZ-1 ROM ENTRY

```
/*************** FUNCTION No.  0  ********************/
  name            : entry
  function        : all system initilize
  usage           : entry();

/*************** FUNCTION No.  6  ********************/
  name    : mgetc
  func    : get char now
  usage   : c=mgetc();
                int c;  c=ERROR no key    ERROR = -1
/*************** FUNCTION No.  7  ********************/
  name    : unmgetc
  func    : back get char now
  usage   : unmgetc( c );
                int c; return key code

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

   16                   /* move up cursor       */
   17                   /*      down            */
   18                   /*      right           */
   19                   /*      left            */
   20                   /* entry menu           */
   21                   /* escape               */
   22                   /* display mode change  */
   24                   /* play mode            */
   25                   /* parameter modify     */
NU 26                   /* set/menu switch      */
   27                   /* transpose            */
   28                   /* tunning              */

/*************** FUNCTION No.  8  ********************/
ame     : contsw
unc     : check continue push switch
sage    : push = contsw( c );
                int push; (OK:continus push)  OK = 0
                int c;    check character
/*************** FUNCTION No.  10 ********************/
  name    : main volume
  func    : read main entrya volume
  usage   : v = mvol();
                int v; if v = ERROR, not change
                       else v = 0~127 value.
/*************** FUNCTION No.  11 ********************/
  name    : envelop value
  func    : read envelop value
  usage   : v = evol();
                int v;
/*************** FUNCTION No.  20 ********************/
  name    : all note off
  func    : note off by select dev
  usage   : all noteoff(i);

/*************** 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
/*************** FUNCTION No.  22 ********************/
  name    : control_on
  func    : send now control value to MIDI out
  usage   : control_on();
/*************** FUNCTION No.  23 ********************/
  name    : control_off
  func    : control initalize to MIDI out
  usage   : control_off();
/*************** 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    hbyte : lbyte
                             KEY_CODE is MIDI note
                             TOUCH    is MIDI touch
/*************** FUNCTION No.  42 ********************/
  name    : gene_off
  func    : generater sound off
  usage   : gene_off( g )
                int g; generator number (0~7)
/*************** FUNCTION No.  43 ********************/
  name    : gabinit
  func    : gate array initialize do
  usage   : gabint();
/*************** 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
/*************** FUNCTION No.  46 ********************/
  name    : rec_trig
  func    : start_post_recording_by_line_1
  usage   : rec_trig();
/*************** 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
/*************** FUNCTION No.  48 ********************/
  name    : set_gain
  func    : set recording gain
  usage   : set_gain(_g_);
```

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

/****************** 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

/****************** 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
/****************** FUNCTION No.  62  ********************/
   name     : set func
   func     : set function # from lv0[0],lv0[1],lv0[2]
   usage    : fn = set_func();

/****************** FUNCTION No.  63  ********************/
   name     : check func
   func     : check function # from lv0[0],lv0[1],lv0[2]
   usage    : fn = chk_func();
/****************** FUNCTION No.  65  ********************/
   name    para_change
   func    all parameter change operation
   usage    : vl = para_change( ppp );        ERROR   no key code
                int vl; last character, ERROR-1 for return
                struct pppdata ppp[];   para data

-------- para_change()'s static parameter ---------------- */
pos;                            /* line pos              */
pos;                            /* column pos            */
max;                            /* line max limit        */
max;                            /* coumn max limit       */
off;                            /* line offset ((0 first*/
top;                            /* line top limit        */
pos;                            /* coumn parameter pos   */

t_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    */
};
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)
*/
/****************** FUNCTION No.  66  ********************/
   name    ; meter
   func    ; sumlate audio level meter
   usage   ; meter( v );
                int v; (ERROR=open  meter, else value )
/****************** 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
/****************** 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.
/****************** 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
/****************** FUNCTION No.  73  ********************/
   name     : d_change
   func     : display change parameter with value,key or swit
   usage    : d_change( mode,v,para );
                int     v;
```

```
/*************** FUNCTION No.  74  ********************/
  name     : d_change_all
  func     : display change with offset dl
  usage    : d_change_all( mode,ppp );
                     int mode; OK=new, ERROR=only valu
                     struct pppdata ppp[];
/*************** FUNCTION No.  75  ********************/
  name : brink
  func : brink timer counter
  usage: b = brink();
         int b;          ERROR -- no change mode
                         0     -- normal write
                         1     -- reverse write;
/*************** FUNCTION No.  76  ********************/
  name     : graph
  func     : graphic edit wave point
  usage    :  c = graph( mode,ppp1,ppp2 );
              int c;          mgetc character
                                  ERROR-1 for end
              int mode;       0 = only display, nocursol
                              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 ---------------------*
osv;                          /* graphic items number  * /
                                 b15  : 0=new wave     *
                                 b14  : 0=new disp     *
                                 b0   : 0=1st,1=2nd    *
osp;                          /* graphic level         *
ni;                           /* verchcal shift        *
id;                           /* graphic width         *
os[4];                        /* graphic postions      *
-ast,graed;                   /* graphic window limit  *
olst,pp1ed;                   /* ppp1 value     limit  *
o2st,pp2ed;                   /* ppp2 value     limit  *
/*************** FUNCTION No.  77  ********************/
  name     : draw for graph subroutine
  usage    : c = d_graph( mode,pos,wid,cur,vhi );
              int c;          cursor changed=ERROR, els
              int mode;       OK=full, ERROR=if changed
              long pos;       wave start address (WORD.
              long wid;       display width      (WORD.
              int cur;        cursor pos         (0~95)
              int vhi;        vertical closeup   ( 0~7)
/*************** FUNCTION No.  79  ********************/
  name     : egraph
  func     : envelop data graphic subroutine
  usage    : c = rgraph( p,e );
              struct paradata p[];
              struct envelop *e;
              int c;                  return char
                                                  func     ;
  usage    : printvn();
/******************** FUNCTION No.  81  ********************/
  name     : print voice number with status
  func     ;
  usage    : printst();
/******************** FUNCTION No.  82  ********************/
  name     : print bank number
  func     ;
  usage    : printbn();
/******************** FUNCTION No.  83  ********************/
  name     : print_name
  func     : print name char (************)
  usage    : print_name( l,m,c,name );
                 int l,m;          line, mode
                 int c;            left brace ( , { or [
                 char n[];         source name string
/******************** 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
/******************** FUNCTION No.  85  ********************/
  name : yes_no
  func : operation yes no question
  usage: yes_no( l,s );
              int l;
              char *s;          some mesage ( 15;
/******************** FUNCTION No.  86  ********************/
  name     : jobbing
  func     : animetion display in jobbing.
  usage    : jobbing();
  remark   : use static
                                  mlevel .. 0 position.
                                  mpeek  .. line position
                                  define in yes_no()
/******************** 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
/******************** FUNCTION No.  88  ********************/
  name     : generator
  func     : generator operation
  usage    : generator();
```

```
name    : envelop
func    : envelop operation
usage   : envelop( vv );
              int vv;          (0:DCA 1:DCF)

/************** FUNCTION No.  96  ********************/
name    : loop_set
func    : loop_set operation
usage   : loop_set();
/************** FUNCTION No.  97  ********************/
name    : lfo_set
func    : lfo_set operation
usage   : lfo_set();

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

/************** FUNCTION No.  99  ********************/
name    : tuning
func    : key set opearation
usage   : tunning();
/************** FUNCTION No. 100  ********************/
name    : delete_voice
func    : delete_voice operation
usage   : delete_voice();

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

/************** FUNCTION No. 102  ********************/
name    : create_bank
func    : create_bank operation
usage   : create_bank();
/************** FUNCTION No. 103  ********************/
name    : delete_bank
func    : delete_bank operation
usage   : delete_bank();

/************** FUNCTION No. 104  ********************/
name    : delete area
func    : delete bank area
usage   : dearea_bank();

/************** FUNCTION No. 105  ********************/
name    : bender_range
func    : bender_range operation
usage   : bender_range();
/************** FUNCTION No. 106  ********************/
name    : vol_dev
func    : volume device editor (moduW,after,foot)

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

/********************* FUNCTION No. 108  *********************/
name    : set_copy
func    : set copy voice or bank number
usage   : set_copy( mode );
              int mode;          mode = VOICE or BANKE
                                 VOICECPY       0
                                 VOICEREP       1
                                 BANKCPY        2
                                 BANKREP        3

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

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

/********************* FUNCTION No. 111  *********************/
name    : keyboard_set
func    :
usage   : keyboard_set();
/********************* FUNCTION No. 112 *********************/
name    : level_fix
func    :
usage   : level_fix();

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

/********************* FUNCTION No. 115  *********************/
name    : rec_do
func    : recording operation
usage   : rec_do( mode );
              int mode; (OK for auto, ERROR for manual)
remark  : trig's msb use for fast rectrig. 1 mean manual
          trig. see in ADCA.S fast envelop mode.
```

```
 func    : initialize voice data after recording
 usage   : init_voice( vp,ll );
           struct voicedata *vp;  voicedata  pointe
           long ll;               voice length;
                                  if ll is zero, set zero

 remark  : vp->loop is null, then all parameter initialize
           isn't null, address 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
/*************** FUNCTION No. 119  ********************
  name    : rec delete
  func    : delete wave without key pos,name
  usage   : rec_wdelete( ev );
*************** FUNCTION No. 120  ********************
  name    :  preset_wave()
  func    :  set preset wave data
  usage   :  preset_wave()

*************** FUNCTION No. 121  ********************
  name    :  sin_add()
  func    :  set sin add synthesizer parameters.
  usage   :  sin_add() ;
  use static
        unsigned short sintable[ MAXSIN ]; sin add table(
                                   =96
*************** FUNCTION No. 123  ********************
  name    :  cut_sample()
  func    :  cut from pcm data
  usage   :  cut_sample()
*************** FUNCTION No. 124  ********************
  name    :  hand_drawing()
  func    :  draw wave data by hand (or cursor)
  usage   :  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 :
        short add_v1,add_v2;   add voice number1,2 (0~63)

*************** FUNCTION No. 127  ********************/
  name    :  add_level()
  func    :
  usage   :  add_level()
  static :
```

```
 name    :  add_delay()
 func    :
 usage   :  add_delay()
 static :
          long add_dly;  add delay address offset      */

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

/******************** FUNCTION No. 130  ********************/
 name    :  add_cross()
 func    :
 usage   :  add_cross()
 use static :
          long add_xs1;  start cross address (WORD address)
          long add_xs2;  end   cross address (WORD address)
/******************** 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.
/******************** FUNCTION No. 132  ********************/
 name    :  check mix excute
 func    :  check_add_v1_&_add_v2_before_mixexecute.
 usage   :  chk_mix( mode );
          int mode;
                    (0=MIX, 1=CROSS, 2=REVERSE)
/******************** FUNCTION No. 133  ********************/
 name    :  init synthesizer
 func    :  initalize_for_one_wave_synthesizer
 usage   :  init_synthe( st );
          int st; OK    : first set
                  ERROR : second set
/******************** FUNCTION No. 134  ********************/
 name    :  preset_write
 func    :  preset_write_caluculaterfor_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
          else nop
/******************** FUNCTION No. 135  ********************/
```

```
usage    : sin_write();

************** FUNCTION No. 136 ********************/
 name     : cut_write
 func     : cut_write caluculaterfor PCM data
 usage    : cut_write( st,ed );
            long st,ed; cut data address (WORD )

************** FUNCTION No. 137 ********************/
 name     : mix
 func     : mix caluculaterfor PCM data
 usage    : mix( mode );
            int mode; OK for mix, ERROR for xmix

************** FUNCTION No. 138 ********************/
 name     : rev
 func     : rev caluculaterfor PCM data
 usage    : rev();
************** FUNCTION No. 139 ********************/
 name     : load_all
 func     : data transfer waveram form DISK,PORT,MIDI
 usage    : load_all( mode,sta );
              mode : LOAD ,MERGE, VERIFY
              sta  : DATA_STAT,BANK_,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,MERGE,VERIFY
              int dev;      device number (DISK,PORT,MID
              int sta;      files status(FULL,BANK,VOICE
              char *name;   file name (in.use dev==DISK

************** FUNCTION No. 141 ********************/
 name     : save_all
 func     : data transmit waveram to DISK,PORT,MIDI
 usage    : save_all( mode,sta );
************** FUNCTION No. 142 ********************/
 name     : erase_all
 func     : erase disk file
 usage    : erase_all( i );
              int i;  file status (DATA,BANK,VOICE,EFEC
************** FUNCTION No. 143 ********************/
 name     : format_all
 func     : listing voice bank files in DISK
 usage    : format_all();
************** FUNCTION No. 144 ********************/
 name     : print_device name
 func     : print device name
 usage    : print_dev( sta );
              int sta;  data status
************** FUNCTION No. 145 ********************/
 usage    : select_dev();
/********************* FUNCTION No. 147 ********************/
 name     : mopen
 function : mopen file name
 usage    : err = mopen(name,ext,fbuf,dbuf);
              int err;        ERROR=open error
              char *name;     file name
              int  ext;       file status
              struct fcb *fbuf;  --> 260 byte work area.
              char dbuf[SYSSIZ]; --> 1024 byte buffer.
/********************* FUNCTION No. 148 ********************/
 name     : mcreat
 function : mcreate file name / open file
 usage    : err = mcreat(name,ext,fbuf,dbuf);
              int err;        ERROR=mcreat miss
              char *name;     mcreate name
              int  ext;       file status
              struct fcb *fbuf;
              char dbuf[SYSSIZ];
/********************* FUNCTION No. 149 ********************/
 name     : mclose
 function : mclose open file
 usage    : err = mclose( fbuf,dbuf );
              int err;        mclose error
              struct fcb *fbuf;
              char dbuf[SYSSIZ];
/********************* FUNCTION No. 150 ********************/
 name     : mread
 function : mread ls claster ( ls * 1024 byte)
 usage    : err = mread(f,ls,data);
              int err;        mread error
              struct fcb *f;
              int ls;         mread claster #
              char data[];    data buffer
 remark   : if datasize less than ls*1024, system d
/********************* FUNCTION No. 151 ********************/
 name     : mwrite
 function : mwrite ls claster ( ls * 256 byte)
 usage    : err = mwrite(f,ls,data);
              int err;        mwrite error
              struct fcb *f;
              int ls;         mwrite claster #
              char data[];    data buffer
 remark   : if endof dbp, ls!=1 fatal error.
/********************* FUNCTION No. 152 ********************/
 name     : delete
 function : delete file
 usage    : err = delete( name,ext );
              int err;        delete error
              char *name;     delete file
              int  ext;       file status
/********************* FUNCTION No. 153 ********************/
 name     : serch
 function: serch file name in dir
```

```
                                (0    fdc err occar
                char *name;     filename;
                int ext;  file extension status
                struct syspar *d;

  remark : this function all initalize fdc
          call befor disk operaion
          name==0 : serch null dir for write
          name==1 : only read dir area with disk_name

          if ext's msb is 1 ,High byte of ext 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
*************** FUNCTION No. 155  ********************/
  name     : catset
  func     : set cat bit
  usage    : catset( i );
                int i;  claster position
*************** FUNCTION No. 156  ********************/
  name     : catreset
  func     : reset cat bit
  usage    : catreset( i );
                int i;  claster position
*************** FUNCTION No. 157  ********************/
  name     : save
  func     : data transfer from wavemem to device
  usage    : save( dev,sta,name );
                int mode;       save mode (SAVE only)
                int dev;        device number
                int sta;        files status
                char *name;     file name
  remark : use dido & eido
*************** FUNCTION No. 158  ********************/
  name     : load
  func     : data transfer from device to wavemem
  usage    : load( mode,dev,sta,name );
                int mode;       load mode (LOAD,MERGE,VER
                int dev;        device number
                int sta;        files status
                char *name;     file name
  remark : use dido & eido
*************** FUNCTION No. 159  ********************/
  name     : format_do
  func     : format disk & initalize cat,disk top,dir
  usage    : format_do(disk,pass );
                char *disk;     disk name
                char *pass;     pass word
  remark : use dido & eido
*************** FUNCTION No. 160  ********************/
  usage    : err=set_files(stat,name);
                int err;        OK      : set name
                                ERROR-1 : ESC or PLAY
                int  stat;      file status(DATA,PROG.etc
                char name[12];  file name area MUST be 12
  remark   : if stat(0 ,High byte of ext is egnored.
/*********************FUNCTION No 161 *********************/
  name     : lcd initialize
  func     : send dispoff/mac/time/dirpon/"ZZ-5000 HEAR"
  usage    : lcdinit();
/********************* 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  print
                                        1 = reverse & pri
                                        2 = reverse,only
                        char *s; null terminated string
/********************* FUNCTION No. 163  *********************/
  name             : cls
  function         : clear_screen
  usage            : cls(xs,ys,xe,ye,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 lc

                                MSB = 1, no send lcd();
  remark           : xs(=xe && ys(=ye must be !!
                     if !(0(=x(=95 && 0(=y(=63) broke memory
/********************* 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
/********************* 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
```

```
  function        : draw_boxline
  usage           : boxline(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
*****************  FUNCTION No. 167  ********************/
    name    : lcd_vol
    func    : lcd volume set, move constrast
    usage   : lcd_vol();
*****************  FUNCTION No. 168  ********************/
  name      : bdelete
  func      : delete bankdata & voicedata using in bank
  usage     : bdelete( bstep,bn );
                        int bstep;      delete from bstep;
                        struct bankdata *bn;

  remark    : Delete bank & voice in bank. If a voice in bn
              bank is used by other bank, then not delete this
              voice.
                The bstep is a offset. Should be delete voice
              between bstep and bn->bstep.
*****************  FUNCTION No. 169  ********************/
  name      : wdelete
  func      : wave data delete
  usage     : wdelete( wn );
                        struct voicedata *vn; voice data number

  remark    : Delete voice and wave data. If wn voice is used
              by other voice, then not delete wave data
*****************  FUNCTION No. 170  ********************/
  name      : wunuse
  func      : delete wave unused part
  usage     : wunuse( vn );
                        struct voicedata *vn;       voice numbe
*****************  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 memsi

*****************  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 twise.
*****************  FUNCTION No. 173  ********************/
  name      : ad_adjust
                        long l;         adjust offset(WORD)
                        struct voicedata *v;
/*********************  FUNCTION No. 174  ********************/
  name      : ad_chk
  func      : address check and repair by memsize
  usage     : err = ad_chk( v );
                        struct voicedata *v;
/*********************  FUNCTION No. 178  ********************/
  name      : merge_bank
  func      : merge source bankdata into dest bankdata
  usage     : merge_bank( d,s );
                        struct bankdata *d;
                        struct bankdata *s;     merge source
/*********************  FUNCTION No. 181  ********************/
  name      : use_wave
  func      : check voice no.vp use number.
  usage     : err = use_wave( vp );
                        int vp;         serch voice #
                        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 voic
/*********************  FUNCTION No. 182  ********************/
  name      : use_voice
  func      : check voice vp's use number.
  usage     : err = use_voice( vp,bb );
                        int vp;         serch voice #
                        int bb;         serch bank  #
                        int err;
                                0       : no use voice in bank bb.
                                1       : first  use in voice
                                2~64    : second use
/*********************  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 #
                        int bb;         serch bank  #
                        int err;
                                ERROR   : no use voice in bank b
                                OK      ; use in bank
/*********************  FUNCTION No. 184  ********************/
  name      : set_wbvnum
  func      : preset saving wave,bank,voice number
  usage     : length = set_wbvnum( sta );
                        int length;     data blocknumber (1024byt
                        int sta;        data status(DATA,BANK,VOI
/*********************  FUNCTION No. 188  ********************/
```

```
usage   : err = lcd(s);
          int err;          err=OK, err=ERROR timeout
          char *s;          0xFF terminated string
remark  : Use outp,inp
************** FUNCTION No. 189 ********************/
name    : dido
func    : disenable all interapt, close time,midi,key
usage   : dido();
************** FUNCTION No. 190 ********************/
name    : eido
func    : disenable all interapt, open time,midi,key
usage   : eido();
************** FUNCTION No. 191 ********************/
name            : cread
function        : physical claster read
usage           : err = cread(pcl,data);
                  int err;          read err
                  int pcl;          claster#
                  char *data;       data buf
remark          : datasize must be greater than FDDSIZbyt
                                    1024 byte
************** FUNCTION No. 192 ********************/
name            : cwrite
function        : physical claster write/
usage           : err = cwrite(pcl,data);
                  int err;          write err
                  int pcl;          claster#
                  char *data;       data buf
remark          : datasize must be greater than FDDSIZbyt
************** FUNCTION No. 194 ********************/
name    : fdc_format()
func    : fdd first fdc_format
usge    : err = fdc_format();
          int err;          error flag

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

************** FUNCTION No. 196 ********************/
name    : seek
func    : seek hed at c cylinder
usage   : seek( c );
          int c;  cylinder number
************** FUNCTION No. 199 ********************/
name    : fdc_check
func    : check fdc status
usage   : st3 = fdc_check();
          int st3 ; 0 = OK no error for read & writ
                    -5 = write protected
                    -6 = not double side disk
                    -7 = not ready, or motor broken
                    -1 = device falut, FDD   broken
```

```
func    : movmemory by BKM
usage   : movmem(s,d,n)
          char *s;          source pointer
          char *d;          destnation pointer
          unsigned n;       byte number (even)
/********************* FUNCTION No. 201 ********************/
name    : cmpmem
func    : cmp memory by BKM
usage   : cmpmem(s,d,n)
          char *s;          source pointer
          char *d;          destnation pointer
          unsigned n;       byte number (even)
/********************* FUNCTION No. 202 ********************/
name    : setmem
func    : set memory by data
usage   : setmem(s,n,d);
          char *s;          set destnation data poine
          int  d;           set data value
          unsigned n;       set number( even )
/********************* 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
          unisgned int 1; data length (BYTE even)
/********************* 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
          unisgned int 1; data length (BYTE even )
/********************* FUNCTION No. 205 ********************/
name    : wput
func    : wave data phisical read
usage   : wput( ad,data )
          long ad;          wave ram addreaa (WORD.)
          int data;         write data
/********************* FUNCTION No. 206 ********************/
name    : wget
func    : wave data phisical read
usage   : data = wget( ad )
          long ad;          wave ram addreaa (WORD.)
          int  data;        write data
/********************* FUNCTION No. 207 ********************/
name    : wcomp
func    : wave data phisical verify comparate
```

```
                char *data;     read buffer point
                unisgned int l; data length (BYTE even)
*************** FUNCTION No. 208 ********************/
  name    : led
  func    : set led light
  usage   : led(playmode);
*************** FUNCTION No. 209  ********************/
  name    : cnvtdec
  func    : convert unsigned binaly into decimal character
  usage   : cnvtbd( d,c,s );
                unsigned int d;  convert data
                int c;           character count (2-6)
                char *s;         char .
*************** FUNCTION No. 210  ********************/
  name    : set_wid
  func    : serch wave data max & min between pos0 & pos1
  usage   : set_wid( gmax,gmin,pos0,pos1 );
            int *gmax,*gmin;
            long pos0,pos1; (WORD address)
*************** FUNCTION No. 211  ********************/
  name    : iocreat
  func    : io port open for write
  usage   : iocreat(mode,sta,&fbuf,&work );
*************** FUNCTION No. 212  ********************/
  name    : ioopen
  func    : io port open for write
  usage   : ioopen(mode,sta,&fbuf,&work );
*************** FUNCTION No. 214  ********************/
  name    : ininit
  func    : load mode initialize
  usage   : ininit();
*************** FUNCTION No. 215  ********************/
  name    : outinit
  func    : save mode initialize
  usage   : outinit();
*************** FUNCTION No. 216  ********************/
  name    : iowrite
  func    : io output 1024byte
  usage   : iowrite( work,byte );   00004,00008
                char *work;     1024byte buffer
*************** FUNCTION No. 217  ********************/
  name    : ioread
  func    : io input 1024byte
  usage   : ioread( work,byte );  00004 00008
                char *work;     1024byte buffer
*************** FUNCTION No. 218  ********************/
```

```
  usage   : midiopen(mode,stat,buf,work );
                int mode;       LOAD,MERGE,SAVE,VERIFY
                int stat;       DATA,BANK,VOICE,EFECT
                char *buf;      256 byte buffer
                char *work;     1024byte buffer
/*********************** FUNCTION No. 219 ********************/
  name    : midicreat
  func    : midi io output initalize
  usage   : midicreat( mode,stat,buf,work );
                int mode;       LOAD,MERGE,SAVE,VERIFY
                int stat;       DATA,BANK,VOICE,EFECT
                char *buf;      256 byte buffer
                char *work;     1024byte buffer
/*********************** FUNCTION No. 220  ********************/
  name    : midiread
  func    : master io input 1024byte
  usage   : midiread(buf,work)
                char *buf;      256 byte buffer
                char *work;     1024byte buffer
/*********************** FUNCTION No. 221  ********************/
  name    : midiwrite
  func    : master io output 1024byte
  usage   : midiwrite( buf,work );
                char *buf;      256 byte buffer
                char *work;     1024byte buffer
/*********************** FUNCTION No. 222  ********************/
  name    : midipeek
  funk    : 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;  send data count (byte)
/*********************** FUNCTION No. 224  ********************/
  name    : midiclose
  func    : master io output 1024byte
  usage   : midiclose( buf );
                char *buf;      256 byte buffer
/*********************** FUNCTION No. 225  ********************/
  name    : mportout
  func    : output port data with check summ
  usage   : mportout( buf,byte );
                char *buf; send data pointer
                int byte;  send data count (byte)
/*********************** FUNCTION No. 226  ********************/
```

```
usage   : sportin( buf,byte );
                char *buf; send data pointer
                int byte; send data count (byte)
/****************** FUNCTION No. 228  ********************/
  name    : play mode
  func    : play mode define
  usage   : play_mode();

/****************** FUNCTION No. 230  ********************/
  name    : tune mode
  func    : tune mode define
  usage   : tune_mode( mode );
                int mode; TMOD : tunning mode
                          RMOD : transpose mode

/****************** FUNCTION No. 232  ********************/
  name    : lcut
  func    : limitter v between max and min
usage   : lcut( v,min,max );
                long *v;
                long min,max;

/****************** FUNCTION No. 233  ********************/
  name    : lswap
  func    : ~~limitter v between max and min~~ swap long data
  usage   : lswap( a,b );
                long *a,*b;
/****************** FUNCTION No. 234  ********************/
  name    : ck_lcd
  func    : full set dot
  usage   : ck_lcd();

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