IEBus™ Dummy Board For Macro Language

AP-ALDM2 User's Manual

Application Corporation

5-8-18 Haramachida Machida-city, Tokyo 194-0013, Japan

TEL. +81-42-732-1377 FAX. +81-42-732-1378

http://www.apply.co.jp/

Ver.1.0 2003.03.25

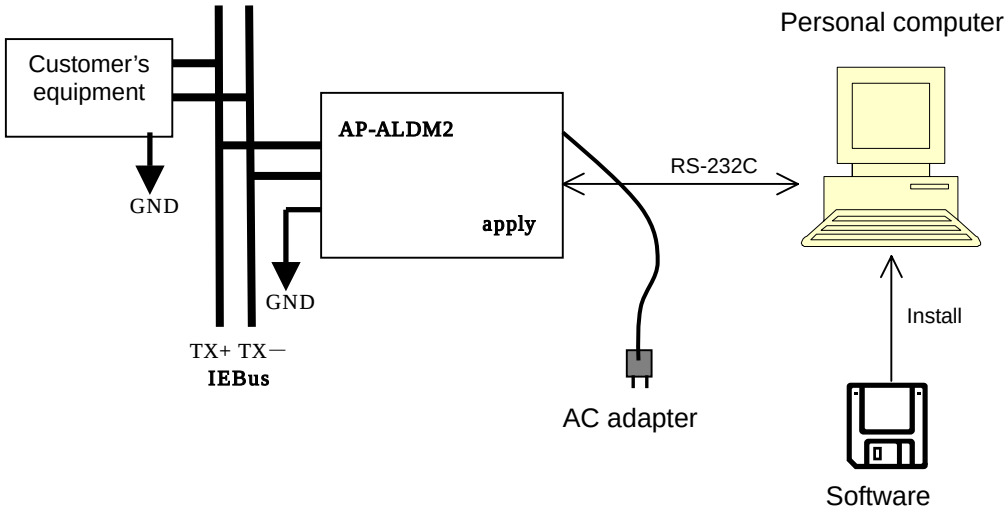
Table of contents

1. Outline

This is the development support equipment for debugging the IEBus communications department of the equipment which the customer developed as dummy equipment of an IEBus unit using a personal computer.
IEBus communication can be transmitted and received with this equipment using the RS-232C part of a personal computer. It is possible to correspond to more complicated protocol by the macro language (AP-Macro) operated on Windows.
Moreover, a communication program can be easily created now by newly developing the macro language which specialized in IEBus. Please use it for the case where the communication place of IEBus is developing, a test of operation etc..
When you carry out the monitor of all the data that flows on IEBus, please use “AP-IEB2” of product of our company.

2. Structure

Figure



Required apparatus

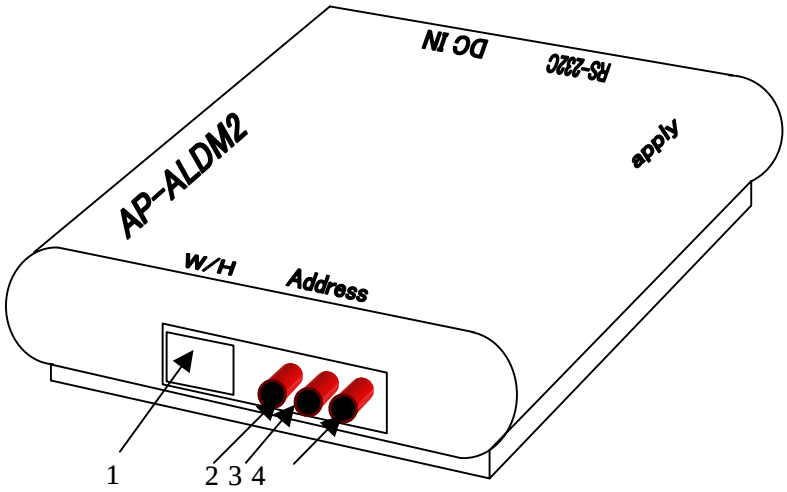
Personal computer	Windows95/98/Me/2000/XP operate and a RS-232C port has 1channel or more.
RS-232C cable	A straight cable with 9pinD-Sub female connector (this equipment side) and a personal computer side connector.
Customer's equipment	Equipment which communicates by making IEBus connection.

Attachments

IEBus dummy board	Equipment which changes communication and RS-232C of IEBus.
Exclusive wire harness	There are 2 kinds of harness. One is with the alligator clip and the other is without the alligator clip.
AC adapter	+12V output
Exclusive soft	Windows95/98/Me/2000/XP application soft which controls this equipment.

Manual

3. Explanation of each part



1. The connector which connects the wire harness
   The harness with the alligator clip or without the alligator clip is connected by it.

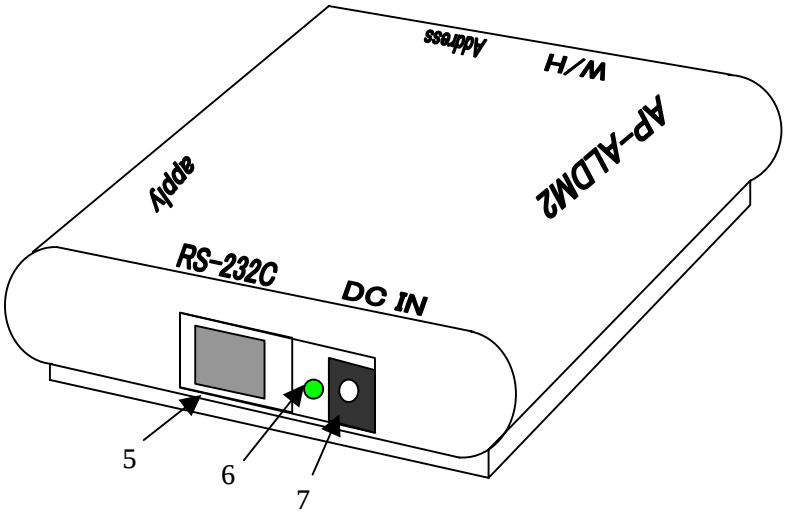
Green alligator clip	TX+ (1pin)
Blue alligator clip	TX- (2pin)
Black alligator clip	GND (3pin)
None	+12V (4pin)

If the attached AC adapter cannot supply [DC IN] with the power, please supply from this +12V.
There are the continuity between +12V and [DC IN].

2. The rotary switch for a setup of the unit address H (bit 11 ~ 8).
3. The rotary switch for a setup of the unit address M (bit 7 ~ 4).
4. The rotary switch for a setup of the unit address L (bit 3 ~ 0).

When a unit address is set as 123h, it sets up below.

Unit address H	1
Unit address M	2
Unit address L	3



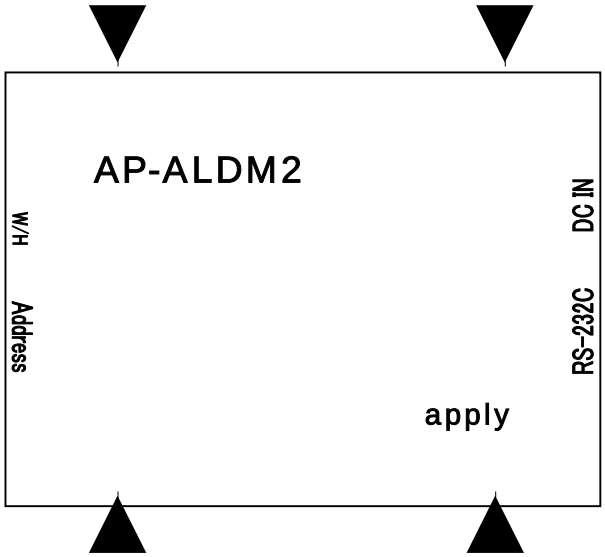
5. RS-232C connector
   Please connect with COM1 or COM2 by the side of personal computer by the straight cable of RS-232C
6. POWER lamp
   The light switches on when a power supply is ON.
7. DC IN connector
   It connects the attached AC adapter.

4. Installation of terminus resistance

In this equipment, 68 ohm and 1/8W resistance is mounted. When you use it as the terminus resistance, you open the body and short the jumper-short-pin JP1.

How to open the body

Please lift the upper part while you hold 4 points of the lower part like the following figure.



5. Software

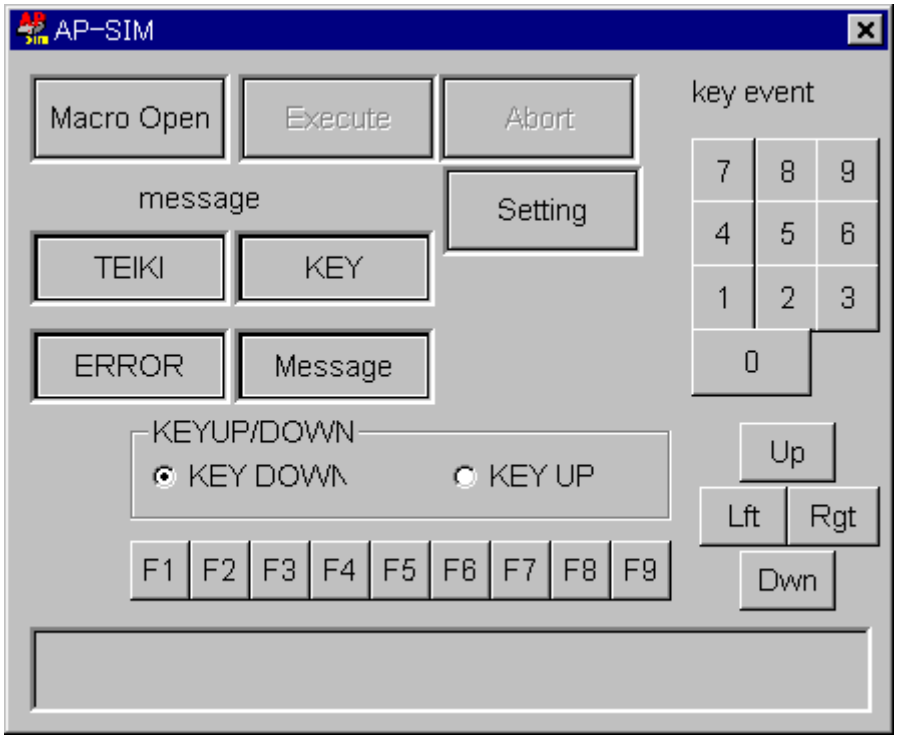
5.1. Install

Please put appending FD into a personal computer and perform setup. exe.
Please specify the holder to install according to the message displayed.

5.2. Execution

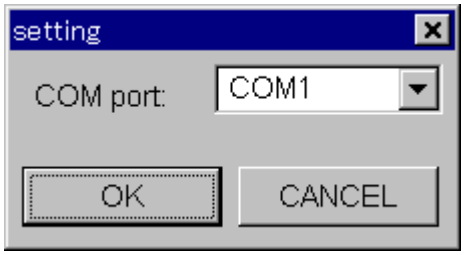
If install normally, since the group of ApSim will be made, apsim. exe is performed from here

The screen at the time of starting



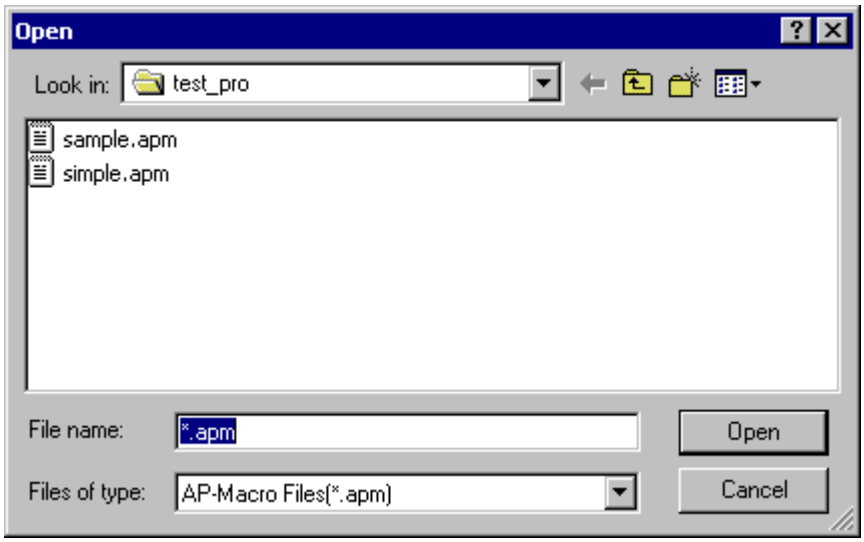
5.2.1. [setting] button

The COM boat which has connected AP-ALDM2 can be set up.

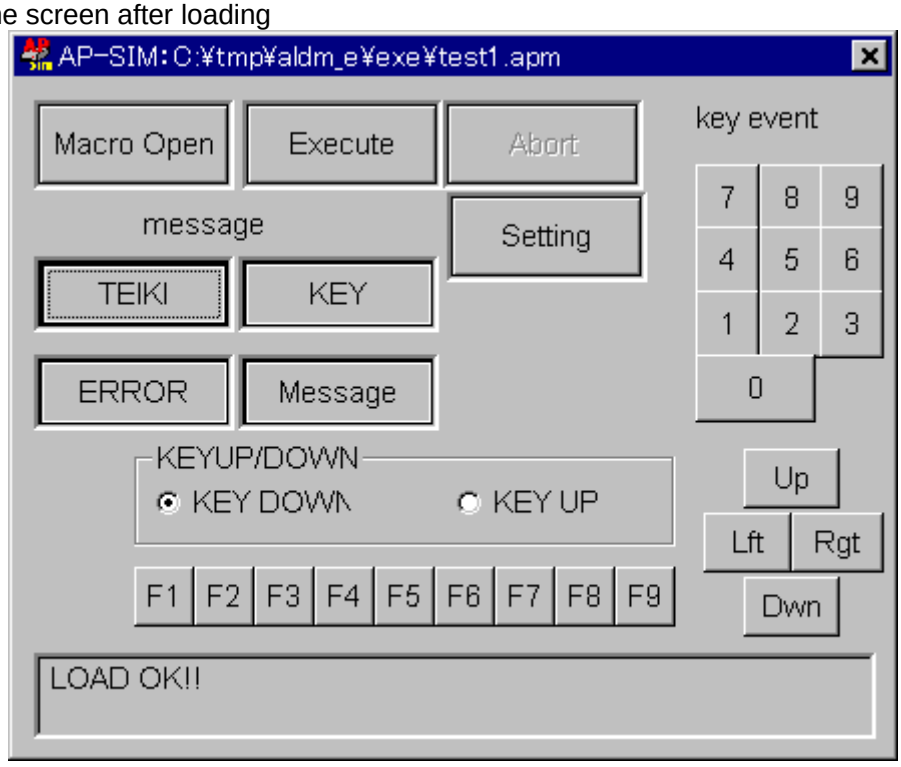


5.2.2. [Macro open] button

The macro file to perform is specified.



The screen after loading



5.2.3. [Execution] button

Loaded macro is executed.

5.2.4. [Abort] button

Macro under execution is forced to terminate.

5.2.5. [TEIKI] button

A TEIKI message window is displayed.

Only a first one command displays the contents of execution of fixed transmitting processing.

5.2.6. [KEY] button

A KEY message window is displayed.

When the key input defined in macro occurs, only one command displays the contents of execution first.

5.2.7. [ERROR] button

An ERROR message window is displayed.

Generating of an error displays an error code and an error line on a window during macro execution.

5.2.8. [Massage] button

A MAIN message window is displayed.

The message which used the ECHO sentence (refer to macro language specification) is displayed.

5.2.9. Key up / Down

The event when clicking the button for key events (0 ~ 9, an arrow, F1 ~ F9) currently displayed on the window is setup.

If it is set as [KEY DOWN], the click of each button will be processed as “having detached the key”.

If it is set as [KEY UP], the click of each button will be processed as “having detached the key”.

It is not influenced by the setup here when actually operation a keyboard. It is setup when clicking a button with a mouse.

5.3. Uninstall

Apsim can be chosen and deleted from “an addition and deletion” of a control panel of application.

6. Macro language specification

The language specification of AP-Macro is explained. AP-Macro describes one command to one line. In Apsim, this is interpreted in interpreter and it performs it one line at a time.

6.1. Comment

In AP-Macro, it becomes a comment after the "#" (character) when these is a "#" (character) at the beginning of a line, the whole line is treated as a comment, this is treated as a comment from the portion, when there is a "#" (character) from the middle of a line, and it is disregarded at the time of execution.

Example:

```
# Transmission of a frame
SEND frame0
RECV frame1     # Receive the reply to the transmitted frame
```

6.2. The model of a variable and declaration

A variable can be used in AP-Macro. There are an IDATA type, an INT type, an IFRAME type, and an IFMASK type as model.
The small letter of the alphabet must describe the 1st character of a variable. The character that can be used for a variable are only the alphabet, an underline, and a number, and the number of the maximum characters is 255 characters.

<**IDATA** type>

An **IDATA** type is used for the arrangement of the byte value of the fixed size of 255 bytes defining the arrangement of IEBus frame data. Each entry is 8 bit value without a mark.

Example:

```
IDATA a={0x00,0x12,0x13}
a[10]=0x55
```

<**INT** type>

An **INT** type is used for defining an integral value. An integral value is 32 bits (with a mark 4 bytes) in size. As a value, it is a value to –2G (GIGA) ~ +2G (GIGA). The value of an **INT** type variable can also be substituted for each members (adr, etc.) of each **IDATA** type., entry, **IFRAME**, and **IFMASK**. At this time, only the low rank bit of 32 bits of **INT** models is substituted as mark-less number.

Example:

```
IDATA a
INT y=1
a[2]=y
```

If it restricts to an INT type, it can treat also as arrangement to dimensional 1 and 2 dimensional.

Example:

```
INT y[2]
y[0]=50
y[1]=30

INT x[2][3]
x[0][2]=50
x[1][2]=30
```

<**IFRAME** type>

An **IFRAME** type defines the IEBus frames It is a structure object internally and each member is as follows. The same system as the C language performs access to each member. The value currently initialized about no member us 0.

```
IFRAME {
  INT bit          Broadcast bit
  INT adr          Slave address
  INT size         Message length
  IDATA data       Frame data arrangement
}
```

Example:

```
IFRAME frame0
IDATA a
frame0.data[3]=5
frame0.data=a
IFRAME frame1={1,0x123,5,0x00,0x01,0x02}
```

Supplement)

Above, 1 is the broadcast bit. 0x123 is the slave address. 5 is the message length. Subsequent 0x00, 0x01…. is the data. And the area of 255-byte fixation is secured inside like a DATA type.

The portion that is not initialized is 0x00.

Broadcast bit is broadcast with 0. Not broadcast bit is broadcast with 1.

<**IFMASK** type>

An **IFMASK** type defines an IEBus frame mask. This is used when carrying out the comparison check of the value and regulation value which used the below-mentioned **CHECK** command and masked at a part of received from. Since it is the same as an **IFRAME** type, completely similarly [the access method etc.] internal comparison can be performed.

Example:

```
IFRAME checkframe={1,0x123,5,0x00,0x01,0x02}
IFRAME recvframe
IFMASK fmask={0,0xffff,0x0,0xff,0xff,0xff}
RECV recvframe
CHECK checkframe recvframe fmask
```

Supplement)

In the above-mentioned **IFMASK** declaration this broadcast bit is a non-mask (besides for a check). A slave address is a mask (candidate for a check). Message length is a non-mask (besides for a check). The amount of first 3 bytes of data is mask (candidate for a check). The command which takes the frame received by the **RECV** command and the frame declared as **checkframe** by **fmask**, and compares “AND” is the **CHECK** command.

6.3. Constant

A constant can be used in **AP-Macro**. A constant value is the same specification as the C language. The numerical value to which x are attached to the head is hexadecimal number. The numerical value to which 0 is attached to the head is octal number. It is decimal number if nothing is attached.
A contact is used for substitution of the initialization value of a variable, and a value, a shift value etc.

6.4. Label

A label can be used in **AP-Macro**. A label is used when the position of a macro file is shown as destinations, such as the below-mentioned **GOTO** command and **IF ~ THEN** command etc. The effective range of a label (scope) is only the inside of procedure (oral statement). Conversely, the label inside procedure cannot be referred to from a main processing part.
The small letter of the alphabet must describe the 1st character of a label. Moreover, the characters that can be used for a label are only the alphabet, an underline, and a number of the maximum character is 255 characters.

Example:
label0:

It is the line which finished with the ":" character as mentioned above and starts by the label name of a small letter. Other command etc. cannot be described in the line which described the label.

6.5. Procedure

AP-Macro can be described a mass of processing of procedure. Procedure is registered as procedure for the below-mentioned fixed processing. Procedure is registered as procedure for key procedure. It is for calling as a subroutine by the GOSUB command.

Example:
PROC proc0
…….Description of processing
ENDPROC

It surely starts in **PROC** as mentioned above, and must be finished as **ENDPROC**. Execution of **GOTO** to the label out of **PROC** is restricted in the same **PROC** (the same is said of the label of **IF**). Moreover, don't perform setup of fixed transmission or key input, and release within **PROC**.

6.6. Operation

AP-Macro can describe the substitution and operation to a variable. The kind of operation which can be described to one line as a formula is shown below.

a[3] += 5	Addition of a constant value
a[3] -= 5	Subtraction of a constant value
a[3] |= 5	Logical sum of a constant value

a[3] &= 5	The logical product of a constant value
a[3] /= 5	Division of a constant value
a[3] *= 5	Multiplication of a constant value
a[3] <<= 5	Left bit shift
a[3] >>= 5	Right bit shift
a[3] = 5	Substitution of a constant value
a[3] += b	Addition to a variable
a[3] -= b	Subtraction of a variable
a[3] |= b	Logical sum of a variable
a[3] &= b	The logical product of a variable
a[3] /= b	Division of a variable
a[3] *= b	Multiplication of a variable
a[3] <<= b	Left bit shift
a[3] >>= b	Right bit shift
a[3] = b	Substitution of a variable

However, the variable type of left position must be the same.
Moreover, between left position, a operator, and right position, the blank (a blank character or TAB) must be contained. A blank must not be between + of a operator, etc. and =.
About logical sum, a logical product, and bit shift operation, the **INT** type which it is with a mark as a variable is also treated as a value without a mark.

6.7. Flow control command

There is the following as a command of the flow control in **AP-Macro**.

<**SWITCH ~ CASE** command>

As a parameter of the **SWITCH** command, you have to specify a variable name. The model of a variable can specify each **INT** type or **IDATA** type entry, **IFRAME** type, and **IFMASK** type member.
The nesting structure of including a **SWITCH** ~ **ENDSWITCH** command into a **SWITCH** ~ **ENDSWITCH** command can also be described.

Example:

```
IDATA      abc
SWITCH abc[3]
 CASE 0
   ……. Description of processing
 ENDCASE
 CASE 0x11
   ……. Description of processing
 ENDCASE
 CASE 0x22
   ……. Description of processing
 ENDCASE
 CASE DEFAULT
   ……. Description of processing
 ENDCASE
ENDSWITCH
```

<**IF ~ THEN** command>

It files to the label described after **THEN** according to the evaluation result of a formula described after **IF**. The variable with a mark is treated as a variable with a mark. The variable without a mark is treated as having no mark.

Example:
IF a != 0 THEN label0
IF a == 0 THEN label0
IF a & 1 THEN label0
IF a < 1 THEN label0
IF a > 1 THEN label0
IF a <= 1 THEN label0
IF a >= 1 THEN label0

<**WHILE** command>

While the evaluation result of a formula described after **WHILE** is true, processing to **ENDWHILE** is performed. The nesting structure if including a **WHILE** ~ **ENDWHILE** command into a **WHILE** ~ **ENDWHILE** command can be also be described.

Example:
WHILE a! =0
……. Description of processing
ENDWHILE

<**GOTO** command>

It files to the label described after **GOTO**.

Example:
GOTO label0

<**EXIT** command>

Macro processing is made to finish compulsory. It specifies without a parameter, and a broad view will be ended if **EXIT** is performed. However, this command cannot be used within procedure.

Example:
EXIT

6.8. Other command

In addition to this in **AP-Macro**, there are the following commands in addition to the command for the below-mentioned CSV file type.

<**WAIT** command>

The part sleep is specified and carries out. (unit: mS)

Example:
WAIT 100

<**SEND** command>

The specified frame is transmitted.

Example:
SEND frame0

<**RECV** command>

A frame is received to the specified variable. A timeout value (unit: mS) can be specified. A timeout can be also omitted. (At the time of an abbreviation, it does not carry out a timeout but receives.)
If the timeout occurs, -1 is set to the system definition variable **errno**.
If it is received normally, 0 is set to it.

Example:
RECV frame0 100

<**CHECK** command>

Only the portion of 1 check the specified frame by the pattern specified by the **IFMASK** type variable. 0 is set to the system definition variable **errno** which will be later mentioned if the compared result is equal, and –1 will be set if not equal.

Example:
IFRAME checkframe={1,0x123,5,0x00,0x01,0x02}
IFRAME recvframe
IFMASK fmask={0x0,0xffff,0x0,0xff,0xff,0xff}
RECV recvframe
CHECK checkframe recvframe fmask
IF errno != 0 THEN label_error
ECHO check OK
……. Description of processing
label_error:
ECHO check error

Supplement)

In the above-mentioned IFMASK declaration, this broadcast bit is a non-mask (besides for a check). An address is a mask (candidate for a check). Massage length is a non-mask (besides for a check). The amount of first 3 bytes of data is a mask (candidate for a check). The frame received by the **RECV** command and the frame declared as **checkframe** are taken by **fmask** and "&" is compared.

<**KEYGOTO** command>

The jump place label jumped when the specified key is pushed, or when it is detached is defined. Keys are the number of 0 to 9, the function key of f1 ~ f9, the arrow key of up/ down/ left/ right, and the alphabetic character (there is no distinction of a capital letter and a small letter) of a ~ z.

Example:
KEYGOTO 0 label0 (u)

"u" behind a label is omissible as an option. If it specifies, it will become the definition at the time of detaching a key.

<**KEYGOSUB** command>

Key processing procedure performed when the specified key is pushed, or when it is detached is defined. Keys are the number of 0 to 9, the function key of f1 ~ f9, the arrow key of up/ down/ left/ right, and the alphabetic character (there is no distinction of a capital letter and a small letter) of a ~ z.

Example:
KEYGOSUB 0 proc0 (u)

"u" after a procedure name is omissible as an option. If it specifies, it will become the definition at the time of detaching a key.

<**BEEP** command>

The beep sound defined by the system is sounded. minfo (message information) of a parameter, mwarn (message warning), syserr (system error), mques

(inquiry), and OK (general beep sound) are each sound assigned by [control panel] – [sound]. It does not sound, when there is no sound card in a personal computer and volume is extracted. pc of parameter is beep sound of built-in speaker. These either can be specified.

Example:
BEEP pc/minfo/mwarn/syserr/mques/ok

<**TEIKI** command>

Fixed processing procedure us registered. The number which can be registered is from 0 to 9. The last parameter is the time interval of a milli second unit.

Example:
TEIKI 0 proc0 100

<**TSTOP** command>

Registration cancellation of fixed processing procedure is carried out.

Example:
TSTOP 0

<**ECHO** command>

A specification character sequence is displayed on a debugging character sequence display window.

Example:
ECHO This is a debugging program.

<**GOSUB** command>

Procedure is called. Since a calling agency is saved at an internal stack, it is also possible to call procedure further from called procedure.

Example:
GOSUB proc0

<**XOR** command>

Exclusive logical sum operation of the variable specified to be the left side, and the variable/ constant specified to be the right side is carried out, and a result is substituted for the left side. The specified variable/ constant are treated as a value without a mark. Only one element can be specified variable is an arrangement variable.

Example:
INT a=0xffffffff
INT b=0xCCCCCCCC
INT c[2][3]
XOR a 0x30303030
XOR a b
c[1][0]=0xddddddddd
XOR c[1][0] b

<**INV** command>

Bit reversal of the specified variable is carried out, and a result is substituted. The specified variable/ treated as a value without a mark. Only one element can be specified when the specified variable is an arrangement variable.

Example:
INT a=0xffffffff
INT c[2][3]
INV a
c[1][0]=0xddddddddd
INV c[1][0]

<**INCLUDE** command>

A file included and it develops in the place. The file name or extensions of an included file are not cared about anything. It surrounds by “” and the full pass of a file is specified. It becomes a current directory when a path is omitted.

Example:
INCLUDE “file.inc”
INCLUDE “c:¥user¥file.inc”

<**DEFINE** command>

The replacement character sequence of a constant or a variable is defined. A replacement character sequence must be a character sequence which starts in an English small letter like a variable. Moreover, don’t overlap a variable name, a label name, a PROC name, and a command name. A definition is possible to 256 pieces. It can’t define only the member if a variable. Moreover, the inside of the character sequence surrounded by “” can’t be defined.

Example:
DEFINE abcd 1
DEFINE aa55 frame.data[1]

<**TIMEGOSUB** command>

Procedure performed after the appointed time (milli second) is defined. The number which can be registered is from 0 to 9. Timer execution is a one time and is applicable to timeout processing etc. Periodic timer processing should use the **TEIKI** command.

Example:
TIMEGOSUB 0 proc0 800

<**TIMEGOTO** command>

This label into which an execution position is changed is defined after the appointed time (milli second). The number which can be registered from 0 to 9. Timer execution is a one time and is applicable to timeout processing etc. Periodic timer processing should use the **TEIKI** command. The restriction matter (scope) of a label is the same other cases, and is restricted in the same PROC or main processing.

Example:
TIMEGOTO 0 label0 800

<**TIMESTOP** command>

Registration cancellation of procedure and label branch which the **TIMEGOSUB** command and the **TIMEGOTO** command defined is carried out. A number is from 0 to 9.

Example:
TIMESTOP 0

6.9. The command for CSV type command

In **AP-Macro**, the file (text file of the one-line one record which divided each field with the comma) of CEV type can be used. The command for using the file of CSV type is as follows.

<**COPENR** command>

CSV type file is read and it opens in the mode. The file name or extension of a file is not cared about anything. It surrounds by “” and the full pass of a file is specified. It becomes a current directory when pass is omitted. Reading of the file after this is altogether performed to this file.

Example:
COPENR “c:¥user¥frame.csv”

<**COPENW** command>

CSV type file is read and it opens in the mode. The file name or extension of a file is not cared about anything. It surrounds by “” and the full pass of a file is specified. It becomes a current directory when pass is omitted. Reading of the file after this is altogether performed to this file.

Example:
COPENW “c:¥user¥frame.csv”

<**CCLOSE** command>

The file opened by **COPENR** or **COPENW** is closed. It distinguishes in the ‘**R**’ character or the ‘**W**’ character.

Example:
CCLOSE R
CCLOSE W

<**CLOAD** command>

It reads into the variable specified from the file opened in reading mode by one record (one line). Specification of arrangement element and one member can also be performed like “**data0[3]**” and “**frame.bit**”. The data (data which had not gone into a variable) of the field in which it remained in a part for one record is thrown away. The variable of the 1-dimensional arrangement and the variable of 2-dimensional arrangement of an **IDATA** type or an **INT** type, not only specification of one element but specification of the whole arrangement and specification called the whole sequence of 2-dimensional arrangement can be performed.

Example:
INT int0
INT int1[2]
INT int2[2][3]
CLOAD frame0
CLOAD frame0.bit
CLOAD data0
CLOAD data0[2]
CLOAD int0
CLOAD int1
CLOAD int1[1]
CLOAD int2
CLOAD int2[1]
CLOAD int2[1][1]

<**CSAVE** command>

The constant of the variable specified to be the file opened in beginning mode are written out by one record (one line). Specification of 1 arrangement element and one number can also be performed like "**data0[3]**" and "**frame.bit**".

The variable of the 1-dimensional arrangement and the variable of 2-dimensional arrangement of an **IDATA** type or an **INT** type, not only specification of one element but specification of the whole arrangement and specification called the whole sequence of 2-dimensional arrangement can be performed.

Example:

```
INT int0
INT int1[2]
INT int2[2][3]
CSAVE frame0
CSAVE frame0.bit
CSAVE data0
CSAVE data0[2]
CSAVE int0
CSAVE int1
CSAVE int1[1]
CSAVE int2
CSAVE int2[1]
CSAVE int2[1][1]
```

6.10 The variable of a system definition

In AP-Macro, there is a variable of the integer value of a system definition called **errno**. this saves the error value generated when command lines, such as the **CHECK** command, were performed. It is used in case it is used combining the **CHECK** command and an **IF ~ THEN** command.

6.11. Sample macro

The sample of the macro file described in the **AP-Macro** language is shown below. Left end "line number:" is the thing of facility for explanation among the macro file text, and it must not describe for an actual macro file.

```
 1: #
 2: # Sample macro
 3: #
 4:
 5: # Variable declaration
 6: IFRAME      fSend0 = {1, 0x123, 5, 0x00, 0x11, 0x22, 0x33, 0x44}
 7: IFRAME      fSend1 = {1, 0x123, 3, 0x56, 0x78, 0x9a}
 8: IFRAME      fSend2 = {1, 0x123, 1, 0xff}
 9: IFRAME      fSend3 = {1, 0x123, 1, 0x55}
10: IFRAME      fSend4 = {1, 0x123, 1, 0xee}
11: IFRAME      fCmp =   {0,     0x123,  0,     0x00, 0x42}
12: IFMASK      fMask = {0x00, 0xFFFF, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00}
13: IFRAME      fRecv
14: INT         cnt
15:
16: # Procedure of fixed transmission
17: PROC        teikiProc0
18:     cnt = 3
19:     WHILE   cnt!=0              # 3 times repetition
20:     SEND    fSend1
```

```
21:     cnt -= 1
22:     ENDWHILE
23:ENDPROC
24:
25: # 5 The procedure when you push down the key
26:  PROC       key5Proc
27:     SEND    fSend2
28: ENDPROC
29:
30: # disposition
31:
32: KEYGOSUB 5 key5Proc
33:
34: label0:
35: RECV        fRecv               # It waits for reception of the frame suitable for comditions
36: IF errno!=0 THEN error
37: CHECK       fRecv fCmp fMask
38: IF errno==0 THEN label1
39: GOTO label0
40:
41: label1:
42: WAIT 500
43: SEND fSend2
44: IF errno!=0 THEN error
45: label2:
46: RECV fRecv                      # Processing is changed by the 1st byte of receiving data
47: IF errno!=0 THEN error
48: SWITCH fRecv.data[0]
49:      CASE 0x11
50:               TEIKI 0 teikiProc0 5000
51:              GOTO label3
52:     ENDCASE
53:     CASE 0x22
54:     SEND fSend3
55:               IF errno!=0 THEN error
56:     ENDCASE
57:     CASE DEFAULT
58:               SEND  fSend3
59:               IF errno!=0 THEN error
60:     ENDCASE
61: ENDSWITCH
62: GOTO label2
63:
64: label3:
65: RECV        fRecv               # The 1st byte receiving data waits to 0x88
66: IF errno!=0 THEN error
67: IF fRecv.data[0]==0x88 success
68: SEND fSend3
69: IF errno!=0 THEN error
70: goto label3
71:
72: error:
73:      EXIT
74: success:
75:      EXIT
```

Explanation of each line

1st line ~ 5th line:

They are a comment line and a blank line. It is ignored at the time of execution.

6th line ~ 14th line:

The variable used within this board view is declared. By **AP-Macro**, variable declaration must be summarized and must be described before processing description.

The part to which initialization data was abbreviated is made into 0, and is initialized. The initialization data of an **IFRAME** type variable is packed sequentially from the left to this broadcast bit, an address, message length, and data portion, and is set. The abridged data portion is 0 too.
Moreover, in AP-Macro, all variables turn into a global variable. There is no concept of a scope about a variable. As for this, the same is said of the inside of procedure. All variables are referred to in common fair.

17th line ~ 23rd line:
It is procedure for using it as fixed processing later. Here, **fSend1 IFRAME** type variable is transmitted.

25th line ~ 28th line:
It is procedure for using it as fixed processing later. Here, **fSend2 IFRAME** type variable is transmitted.

30th line ~:
it performs as main processing from here.

32nd line:
Key processing is defined. **Key5Proc** will come to be started if the key of '5' of a number is pushed after this.

33rd line:
The label is defined.

35th, 36th line:
It receives to a **fRecv** variable. In the following line, supposing it confirmed whether the error took place by the **RECV** command of the last line and the error has taken place, it will fly to the label error of the 72nd line.

37th ~ 39th line:
A **fMask** variable and "&" are taken and the comparison check of the frame received to the **fRecv** variable is carried out with the contents of a **fCmp** variable. In the following line, it confirms whether the result of the **CHECK** command of the last line is equal, it will fly to label 1 of the 41st line. Otherwise, it returns to label 0 of the 34th line by the **GOTO** command of the following line, and reception ~ check is repeated again.

42nd ~ 44th line:
The **WAIT** command a 500 milli second it waits and the contents of **fSend2** variable are transmitted. If a transmitting error takes place, it will fly to the label error of the 72nf line.

45th ~62nd line:
Here, processing is divided by the 1st byte of the data part of the received frame using the **SWITCH-CASE** command. When the 1st byte of a data part is 0x11, it sets so that **teikiProc** procedure may be started every 5 seconds by the **TEIKI** command, and progresses to **label 3** of the 64th line. When the 1st byte of a data part is 0x22, after transmitting the contents of **fSend3** variable, it returns to **label 2** by **GOTO**, and waits for reception again. **CASE DEFAULT** is the portion performed when the 1st byte of a data part is not 0x11 or 0x22, either and is doing the thing same as contents as the case of 0x22.

64th ~ 70th line:
Here, processing is divided by the 1st byte of the data part of the received frame using the **IF ~ THEN** command. When the 1st byte of a data part is 0x88, it files to success of the 74th line and macroscopic processing is ended. When that is not right, after transmitting the

contents of **fSend3** variable, it returns to label3 of the 64th line again.

72nd line:
It is the label **error** which files when a reception error and a transmitting error take place in macroscopic.

74th line:
It is the label **success** which files when all processing finally progress normally in microscope.

7. Communication specification

The following is shown the communication specification of RS-232C between AP-ALDM2 and a personal computer. All data is a binary form.

AP-ALDM2 -> personal computer

Broadcast bit	1 byte
Master address	2 byte
Message length	1 byte
Data	1 ~ 32 byte (owing to the message length)
Status	1 byte (00h is normal and the communication error occurs by 01h.)

personal computer -> AP-ALDM2

Broadcast bit	1 byte
Slave address	2 byte
Message length	1 byte
Data	1 ~ 32 byte (owing to the message length)

Communication parameter

Baud rate	115200bps
Parity	None
Data bit	8 bit
Stop bit	1 bit
Flow control	CTS/RTS flow control