Build 9 by max9701s.

Indice

Media Highway Specifications

Constants

Event Categories

```
EV_CATEGORY_TIMER           = 14
EV_CATEGORY_MODULE_END      = 21
EV_CATEGORY_REMOTE_KEY      = 128
EV_CATEGORY_CONSOLE_KEY     = 132
EV_CATEGORY_MODULE_DOWNLOAD = 135
EV_CATEGORY_VOLUME_DOWNLOAD = 135
```

Volume Identifiers

```
VOLUME_ID_RAM      = 0
VOLUME_ID_FLASH    = 1
VOLUME_ID_FIRMWARE = 2
```

Panel Positions

```
PANEL_POS_PANEL_TOP_LEFT       = 0
PANEL_POS_PANEL_CENTER         = 1
PANEL_POS_DISPLAY_TOP_LEFT     = 2
PANEL_POS_DISPLAY_TOP_RIGHT    = 3
PANEL_POS_DISPLAY_BOTTOM_LEFT  = 4
PANEL_POS_DISPLAY_BOTTOM_RIGHT = 5
PANEL_POS_DISPLAY_CENTER       = 6
PANEL_POS_PANEL_TOP_LEFT       = 7
PANEL_POS_PANEL_CENTER         = 8
PANEL_POS_DISPLAY_UNDER_CENTER = 9
PANEL_POS_DISPLAY_TOP_LEFT     = 10
PANEL_POS_UNKNOWN              = 11
PANEL_POS_UNKNOWN              = 12
PANEL_POS_PANEL_TOP_LEFT       > 12
```

Displaying modalities

```
DISPLAYING_MODE_HIDE = 64
```

Window attributes

```
WINDOW_ATTR_DRAW_COLOR            = 0
WINDOW_ATTR_BACKGROUND_COLOR      = 1
WINDOW_ATTR_DRAW_SIZE             = 6
WINDOW_ATTR_TEXT_VERTICAL_JUSTIFY = 14
WINDOW_ATTR_UNKNOWN               = 27
```

Structures

Opi Table (119 bytes)

```
0    UA                            (6 bytes)
6    NUMBER of supported providers (1 byte)
7    INDEX of provider n. 0        (1 byte)
8    IDENT of provider n. 0        (2 bytes)
10   PPUA of provider n. 0         (4 bytes)
14   INDEX of provider n. 1        (1 byte)
15   IDENT of provider n. 1        (2 bytes)
17   PPUA of provider n. 1         (4 bytes)
21   INDEX of provider n. 2        (1 byte)
22   IDENT of provider n. 2        (2 bytes)
26   PPUA of provider n. 2         (4 bytes)
27   INDEX of provider n. 3        (1 byte)
29   IDENT of provider n. 3        (2 bytes)
31   PPUA of provider n. 3         (4 bytes)
35   INDEX of provider n. 4        (1 byte)
36   IDENT of provider n. 4        (2 bytes)
38   PPUA of provider n. 4         (4 bytes)
42   INDEX of provider n. 5        (1 byte)
43   IDENT of provider n. 5        (2 bytes)
45   PPUA of provider n. 5         (4 bytes)
49   INDEX of provider n. 6        (1 byte)
50   IDENT of provider n. 6        (2 bytes)
52   PPUA of provider n. 6         (4 bytes)
```

56	INDEX of provider n. 7	(1 byte)
57	IDENT of provider n. 7	(2 bytes)
59	PPUA of provider n. 7	(4 bytes)
63	INDEX of provider n. 8	(1 byte)
64	IDENT of provider n. 8	(2 bytes)
66	PPUA of provider n. 8	(4 bytes)
70	INDEX of provider n. 9	(1 byte)
71	IDENT of provider n. 9	(2 bytes)
73	PPUA of provider n. 9	(4 bytes)
77	INDEX of provider n. 10	(1 byte)
78	IDENT of provider n. 10	(2 bytes)
80	PPUA of provider n. 10	(4 bytes)
84	INDEX of provider n. 11	(1 byte)
85	IDENT of provider n. 11	(2 bytes)
87	PPUA of provider n. 11	(4 bytes)
91	INDEX of provider n. 12	(1 byte)
92	IDENT of provider n. 12	(2 bytes)
94	PPUA of provider n. 12	(4 bytes)
98	INDEX of provider n. 13	(1 byte)
99	IDENT of provider n. 13	(2 bytes)
101	PPUA of provider n. 13	(4 bytes)
105	INDEX of provider n. 14	(1 byte)
106	IDENT of provider n. 14	(2 bytes)
108	PPUA of provider n. 14	(4 bytes)
112	INDEX of provider n. 15	(1 byte)
113	IDENT of provider n. 15	(2 bytes)
115	PPUA of provider n. 15	(4 bytes)

Lut (64 bytes)

0	Red value of color n. 0	(1 byte)
1	Green value of color n. 0	(1 byte)
2	Blue value of color n. 0	(1 byte)
3	Alpha blending value of color n. 0	(1 byte)
4	Red value of color n. 1	(1 byte)
5	Green value of color n. 1	(1 byte)
6	Blue value of color n. 1	(1 byte)
7	Alpha blending value of color n. 1	(1 byte)
8	Red value of color n. 2	(1 byte)
9	Green value of color n. 2	(1 byte)
10	Blue value of color n. 2	(1 byte)
11	Alpha blending value of color n. 2	(1 byte)
12	Red value of color n. 3	(1 byte)
13	Green value of color n. 3	(1 byte)
14	Blue value of color n. 3	(1 byte)
15	Alpha blending value of color n. 3	(1 byte)
16	Red value of color n. 4	(1 byte)
17	Green value of color n. 4	(1 byte)
18	Blue value of color n. 4	(1 byte)
19	Alpha blending value of color n. 4	(1 byte)
20	Red value of color n. 5	(1 byte)
21	Green value of color n. 5	(1 byte)
22	Blue value of color n. 5	(1 byte)
23	Alpha blending value of color n. 5	(1 byte)
24	Red value of color n. 6	(1 byte)
25	Green value of color n. 6	(1 byte)
26	Blue value of color n. 6	(1 byte)
27	Alpha blending value of color n. 6	(1 byte)
28	Red value of color n. 7	(1 byte)
29	Green value of color n. 7	(1 byte)
30	Blue value of color n. 7	(1 byte)

```
31  Alpha blending value of color n. 7  (1 byte)
32  Red value of color n. 8             (1 byte)
33  Green value of color n. 8           (1 byte)
34  Blue value of color n. 8            (1 byte)
35  Alpha blending value of color n. 8  (1 byte)
36  Red value of color n. 9             (1 byte)
37  Green value of color n. 9           (1 byte)
38  Blue value of color n. 9            (1 byte)
39  Alpha blending value of color n. 9  (1 byte)
40  Red value of color n. 10            (1 byte)
41  Green value of color n. 10          (1 byte)
42  Blue value of color n. 10           (1 byte)
43  Alpha blending value of color n. 10 (1 byte)
44  Red value of color n. 11            (1 byte)
45  Green value of color n. 11          (1 byte)
46  Blue value of color n. 11           (1 byte)
47  Alpha blending value of color n. 11 (1 byte)
48  Red value of color n. 12            (1 byte)
49  Green value of color n. 12          (1 byte)
50  Blue value of color n. 12           (1 byte)
51  Alpha blending value of color n. 12 (1 byte)
52  Red value of color n. 13            (1 byte)
53  Green value of color n. 13          (1 byte)
54  Blue value of color n. 13           (1 byte)
55  Alpha blending value of color n. 13 (1 byte)
56  Red value of color n. 14            (1 byte)
57  Green value of color n. 14          (1 byte)
58  Blue value of color n. 14           (1 byte)
59  Alpha blending value of color n. 14 (1 byte)
60  Red value of color n. 15            (1 byte)
61  Green value of color n. 15          (1 byte)
62  Blue value of color n. 15           (1 byte)
63  Alpha blending value of color n. 15 (1 byte)
```

Procedures

proc_006A * Declare_Event

Attach a script to specified events.

```
Declare_Event (  Script_Path,     In
                 Event_Category,  In
                 Event_Code);     In

Event_Code =    -1  All events of specified category
           >=    0  Only specified event
```

See also: Undeclare_Event.

proc_006B * Undeclare_Event

Detach a script from specified events.

```
Undeclare_Event (  Event_Category,  In
                   Event_Code);     In

Event_Code =    -1  All events of specified category
           >=    0  Only specified event
```

See also: Declare_Event.

proc_012D * Copy_Adr_To_Int

```
Copy_Adr_To_Int (  Mem_Adr,     In
                   Adr_Value);  Out
```

See also: Copy_Int_To_Adr.

proc_0131 * Flash_Item_Write

```
Flash_Item_Write (  Record_Name,  In
                    Item_Name,    In
                    Buf_Adr,      In
                    Buf_Size,     In
                    Error_Code);  Out
```

See also: Flash_Item_Read, Flash_Item_Info.

proc_0132 * Flash_Item_Info

```
Flash_Item_Info (  Record_Name,  In
                   Item_Name,    In
                   Item_Adr,     Out
                   Item_Size,    Out
                   Error_Code);  Out
```

proc_0133 * Flash_Module_Write

Write a module to flash.

```
Flash_Module_Write (  Module_Name,  In
                      Error_Code);  In
```

proc_0134 * Flash_Text_Write

Read a text item from flash.

```
Flash_Text_Write (  Record_Name,  In
                    Item_Name,    In
                    Text,         In
                    Count,        In
                    Error_Code);  Out
```

See also: Flash_Text_Read.

proc_0135 * Download_Device_Volume

Start downloading of a volume module list.

```
Download_Device_Volume (  Device_Volume_Name,  In
                          Download_id,         In
                          Error_Code);         In
```

Device_Volume_Name = "MLOAD", "LCARD", "RCARD", "MODEM", "SERIAL", "PARALLEL".

Use EV_CATEGORY_VOLUME_DOWNLOAD with Download_id event code for waiting module download termination, Evt_Param1 contains the volume id if Evt_Param2 is zero.

See also: Download_Volume_Module.

proc_0136 * Download_Volume_Module

Start downloading of a volume module.

```
Download_Volume_Module (  Volume_id,    In
                          Module_Name,  In
                          Download_id,  Out
                          Error_Code);  Out
```

Use EV_CATEGORY_VOLUME_DOWNLOAD with Download_id event code for waiting module download termination.

See also: Download_Device_Volume, Abort_Module_Download, Unload_Volume_Module.

proc_013C * Receive_Event

```
Receive_Event (  -1,              In
                 Event_Category,  In
                 Event_Code,      In
                 -1,              In
                 -1,              In
                 0,               In
                 J,               Out
                 Event_Param1,    Out
                 Event_Param2,    Out
                 Error_Code);     Out
```

proc_013D * Flash_Clear

Delete all records in flash.

```
Flash_Clear (  Error_Code);  Out
```

See also: Flash_Record_Delete.

proc_013E * Timer_Stop

```
Timer_Stop (  Timer_id,        In
              Event_Category,  In
              Event_Code);     In
```

See also: Timer_Start.

proc_013F * Close_File

Close a opened file.

```
Close_File (  File_id);  In
```

See also: Open_File.

proc_0142 * Trim_Left

Delete all empty spaces from left of input string.

```
Trim_Left (  Input_String,  In
             Trim_String);  Out
```

See also: Trim_Right.

proc_0146 * Copy_Int_To_Mem

```
Copy_Int_To_Mem (  Buf_Adr,    In
                   Buf_Ofs,    In
                   Int_Value,  In
                   Int_Size);  In
```

proc_0147 * Copy_Mem_To_Mem

```
Copy_Mem_To_Mem (  Dst_Adr,  In
                   Dst_Ofs,  In
                   Src_Adr,  In
                   Src_Ofs,  In
                   Length);  In
```

proc_0148 * Copy_Mem_To_Adr

```
Copy_Mem_To_Adr (  Buf_Adr,   In
                   Buf_Ofs,   In
                   Mem_Adr);  Out
```

proc_0149 * Copy_Mem_To_Int

```
Copy_Mem_To_Int (  Buf_Adr,    In
                   Buf_Ofs,    In
                   Int_Value,  Out
                   Int_Size);  In
```

proc_014A * Copy_Mem_To_Text

```
Copy_Mem_To_Text (  Buf_Adr,  In
                    Buf_Ofs,  In
                    Text,     Out
                    Count);   In
```

proc_014B * Copy_Mem_To_Var

```
Copy_Mem_To_Var (  Class_Name,  In
                   Var_String,  In
                   Buf_Adr,     In
                   Buf_Ofs);    In
```

See also: Copy_Var_To_Mem.

proc_014C * Copy_Text_To_Mem

```
Copy_Text_To_Mem (  Buf_Adr,  In
                    Buf_Ofs,  In
                    Text,     In
                    Count);   In
```

proc_014D * Copy_Var_To_Mem

```
Copy_Var_To_Mem (  Class_Name,  In
                   Var_String,  In
                   Buf_Adr,     In
                   Buf_Ofs);    In
```

See also: Copy_Mem_To_Var.

proc_0151 * Get_Cpu_Speed

```
Get_Cpu_Speed (  Cpu_Speed);  Out
```

Cpu_Speed is specified in MHz.

proc_0152 * Create_Window

```
Create_Window (  Left,            In
                 Top,             In
                 Width,           In
                 Height,          In
                 Display_Number,  In
                 Window_id);      Out
```

Window_id = -1 **Error creating window**

proc_0153 * Create_Display

```
Create_Display (  Display_Number,  In
                  Left,            In
                  Top,             In
                  Width,           In
                  Height,          In
                  Lut_Flag,        In
                  Lut_Adr,         In
                  ?,               In
                  Error_Code);     Out
```

Lut_Flag = 0 **Set default application Lut**
            1 **Set specified Lut**

See also: Free_Display.

proc_0155 * Get_Date

```
Get_Date (  Date_Day,      Out
            Date_Month,    Out
```

```
            Date_Year,     Out
            Date_String);  Out
```

Example:

Date_Day = 20, Date_Month = 12, Date_Year = 2002, Date_String = "20/12/2002".

See also: Get_Time.

proc_0157 * Delete_Text_Line

```
Delete_Text_Line (  Text,          In/Out
                    Line_Number);  In
```

proc_0158 * Unload_Ico_File

```
Unload_Ico_File (  Ico_File_id);  In
```

proc_0159 * Free_Window

```
Free_Window (  Window_id);  In
```

See also: Create_Window.

proc_015A * Device_Alloc_Buffer

```
Device_Alloc_Buffer (  Client_id,    In
                       Buf_Size,     In
                       Lock_Flag,    In
                       Buf_Adr,      Out
                       Error_Code);  Out
```

```
Lock_Flag =   0  Unlock buffer
              1  Lock buffer
```

See also: Device_Free_Buffer.

proc_015B * Device_Call

```
Device_Call (  Client_id,    In
               Device_id,    In
               Command_id,   In
               Em_Adr,       In
               Rec_Adr,      In
               Call_Report,  Out
               Error_Code);  Out
```

proc_015C * Device_Close_Channel

```
Device_Close_Channel (  Client_id,    In
                        Error_Code);  Out
```

proc_015D * Device_Close_Device

```
Device_Close_Device (  Client_id,    In
                       Device_id,    In
                       Error_Code);  Out
```

proc_015E * Device_Event

```
Device_Event (  Client_id,       In
                Device_id,       In
                Event_Cmde,      In
                Event_id,        In
                Event_Priority,  In
                Error_Code);     Out

Event_Cmde =   GET_EVENT = 0  Not send device events
               RET_EVENT = 1  Send device events
```

Generated events have category value equal to Event_id and code value equal to Client_id.

proc_015F * Device_Free_Buffer

```
Device_Free_Buffer (  Client_id,    In
                      Buf_Adr,      In
                      Error_Code);  Out
```

See also: Device_Alloc_Buffer.

proc_0160 * Device_Info_List

```
Device_Info_List (  Client_id,    In
                    Device_id,    In
                    Rec_Adr,      In
                    Error_Code);  Out
```

Channel_Buf_Adr contains the address in the memory zone where the following information is found:

```
0   0x106F?          (2 byte)
2   Device_Version   (2 byte)
4   Max_Clients      (4 byte)
8   Current_Clients  (4 byte)
```

proc_0163 * Device_Io

```
Device_Io (  Client_id,       In
             Device_id,       In
             Command_id,      In
             Event_Category,  In
             Event_Priority,  In
             Em_Adr,          In
             Rec_Adr,         In
             Io_Report,       Out
             Error_Code);     Out
```

Generated events have code value equal to Client_id.

proc_0164 * Device_Lock_Buffer

```
Device_Lock_Buffer (  Client_id,    In
                      Lock_Flag,    In
                      Buf_Adr,      In
                      Error_Code);  Out

Lock_Flag =   0  Unlock buffer
              2  Lock buffer
```

proc_0165 * Device_Open_Channel

```
Device_Create_Client (  Client_id,    Out
                        Error_Code);  Out
```

proc_0166 * Device_Open_Device

```
Device_Open_Device (  Client_id,    In
                      Device_id,    In
                      Error_Code);  Out
```

proc_016D * Window_Draw_Bitmap

```
Window_Draw_Bitmap (  Window_id,    In
                      Ico_File_id,  In
                      Bitmap_Name,  In
                      Left,         In
                      Top,          In
                      Width,        In
                      Height,       In
                      ?);           In

Width  =   -1  Use bitmap width
Height =   -1  Use bitmap height
```

proc_0172 * ?

```
? (  Text,   In
     Int,    In
     Text,   In
     Int,    In
     Pos,    Out
     Char);  In

Example:

proc_0172(embook.T,1,Tt,61,I,"\n");
proc_0172(Tt,1,embook.T,70,I,"<");
proc_0172(Tt,1,embook.T,70,I,">");
proc_0172(embook.T,1,Tt,61,I,Chr(64));
```

proc_0173 * Extract_Text_Line

```
Extract_Text_Line (  Text,          In
                     Line_String,   Out
```

```
                    Line_Number);  In
```

proc_0174 * Copy_Text_Line

```
Copy_Text_Line (  Text,          In
                  Line_String,   Out
                  Line_Length);  In
```

A text line is delimited by "\n" (0x0A) character.

proc_017A * Free_Display

```
Free_Display (  Display_Num,  In
                Error_Code);  Out
```

See also: Create_Display.

proc_017C * Semaphore_Release

```
Semaphore_Release (  Semaphore_id);  In
```

See also: Semaphore_Acquire.

proc_017D * Get_Ticks

```
Get_Ticks (  Ticks_Count);  Out
```

Ticks_Count is specified in 1/100 of second and start from 0 at launching.

proc_0184 * Get_Display_Info

```
Get_Display_Info (  Display_Num,   In
                    Left,          Out
                    Top,           Out
                    Width,         Out
                    Height,        Out
                    Display_Num);  Out
```

Error if Display_Num on exit is equal to –1.

proc_0185 * Eeprom_Read

```
Eeprom_Read (  Buf_Adr,      In
               Buf_Ofs,      In
               Count,        In
               Eeprom_Ofs,   In
               Error_Code);  Out
```

Buf_Ofs, Eeprom_Ofs and Count are specified in bits.

See also: Eeprom_Write.

proc 018E * Get_Module_List

```
Get_Module_List (  Volume_id,    In
                   Text_List,    Out
                   Text_Size,    In
                   Error_Code);  Out
```

proc_0192 * Get_On_Event_Info

Return info about last event occurred.

```
Get_On_Event_Info (  Event_Category,  Out
                     Event_Code,      Out
                     Event_Param1,    Out
                     Event_Param2);   Out
```

See also: Wait_Event, Wait_Multiple_Event.

proc_0194 * Get_Panel_Id

```
Get_Panel_Id (  Panel_Name,  In
                Panel_Id);   Out
```

proc_0195 * Get_Key_Pressed

```
Get_Key_Pressed (  Key_Code);  Out
```

proc_0196 * Get_Panel_Page

```
Get_Panel_Page (  Panel_id,     In
                  Panel_Page);  Out
```

proc_019D * Widget_Get_Enum

```
Widget_Get_Enum (  Panel_id,     In
                   Widget_Name,  In
                   Enum_Value);  Out
```

See also: Widget_Set_Enum.

proc_019E * Widget_Get_Focused

```
Widget_Get_Focus (  Panel_id,      In
                    Widget_Name);  Out
```

See also: Widget_Set_Focus.

proc_01A2 * Widget_Get_Value

```
Widget_Get_Value (  Panel_id,     In
                    Widget_Name,  In
                    Value);       Out
```

See also: Widget_Set_Value.

proc_01A7 * Widget_Get_Text

```
Widget_Get_Text (  Panel_id,     In
                   Widget_Name,  In
                   Text);        Out
```

See also: Widget_Set_Text.

proc_01A9 * Widget_Get_Window_Id

```
Widget_Get_Window_Id (  Panel_id,     In
                        Widget_Name,  In
                        Window_id);   Out
```

proc_01AB * Tcs_Get_Channel_Struct_Size

```
Tcs_Get_Channel_Struct_Size (  Struct_Size);  Out
```

proc_01AC * Tpt_Get_Transponder_Struct_Size

```
Tpt_Get_Transponder_Struct_Size (  Struct_Size);  Out
```

proc_01B3 * Hide_Panel

```
Hide_Panel (  Panel_id);  In
```

See also: Show_Panel.

proc_01B6 * Insert_Text_Line

```
Insert_Text_Line (  Text,          In/Out
                    Line_String,   In
                    Line_Number);  In
```

proc_01B7 * Copy_Int_To_Adr

```
Copy_Int_To_Adr (  Mem_Adr,     Out
                   Adr_Value);  In
```

See also: Copy_Adr_To_Int.

proc_01B9 * Int_To_Str

```
Int_To_Str (  Value,   In
              String,  Out
              Width,   In
              Mode);   In
```

```
Mode =   0       Fill string at left with "0" character.
         1       Fill string at left with " " character.
         2       Fill string at right with " " character.
```

Others **Normal integer to string conversion.**

See also: Str_To_Int.

proc_01BA * Unload_Volume_Module

```
Clear_Module (  Module_Name,  In
                Error_Code);  Out
```

See also: Download_Volume_Module.

proc_01BB * Abort_Module_Download

Abort download of a volume module.

```
Abort_Module_Download (  Download_id,  In
                         Error_Code);  Out
```

See also: Download_Volume_Module.

proc_01BD * Get_Device_Volume_Id

```
Get_Device_Volume_Id (  Device_Volume_Name,  In
                        Volume_Id,           Out
                        Error_Code);         Out
```

```
Device_Volume_Name =   "MLOAD", "LCARD", "RCARD", "MODEM", "SERIAL", "PARALLEL".
```

See also: Open_Device_Volume.

proc_01BE * Load_Volume_Module

```
Load_Volume_Module (  Volume_Id,    In
                      Module_Name,  In
                      Error_Code);  Out
```

proc_01BF * Close_Volume

```
Clear_Volume (  Volume_id,    In
                Error_Code);  Out
```

proc_01C0 * Get_Text_Lines

```
Get_Text_Lines (  Text,          In
                  Lines_Count);  Out
```

proc_01C1 * Search_Text_Line

```
Search_Text_Line (  Text,           In
                    Search_String,  In
                    ?,              In
                    Search_Mode,    In
                    Line_Number);   Out
```

```
Search_Mode =   0  Partial line search
                1  Full line search

Line_Number =   -1  Line not found
            >   0   Line found
```

proc_01C3 * Load_Module

Load and execute a volume application module.

```
Load_Module (  Module_Name,  In
               Module_id,    Out
               Error_Code);  Out
```

Use EV_CATEGORY_MODULE_END for waiting a module termination.

proc_01C8 * Module_End

```
Module_End (  );
```

proc_01CB * New_Panel

```
New_Panel (  Panel_Name,       In
             Display_Number,   In
             Panel_Position,   In
             Position_X,       In
             Position_Y,       In
             Displaying_Mode,  In
             Panel_id);        Out
```

See also: Quit_Panel.

proc_01CD * Open_File

```
Open_File (  0?,         In
             File_Path,  In
             ??,         In
             File_id);   Out
```

?? = 0, 4

```
File_Id =   -1  Open file error
        <>  -1  Open file ok
```

See also: Close_File.

proc_01D2 * Send_Event

```
Send_Event (  Event_Category,  In
              Event_Code,      In
              Event_Param1,    In
              Event_Param2);   In
```

proc_01D3 * Quit_Panel

```
Quit_Panel (  Panel_id);  In
```

See also: New_Panel.

proc_01D4 * Random

Generate a random value.

```
Random (  Random_Value);  Out
```

proc_01D6 * File_Read

```
File_Read (  File_id,      In
             Buf_Adr,      In
             Count,        In
             Bytes_Read);  Out
```

See also: File_Open, File_Seek.

proc_01D7 * Load_Ico_File

```
Load_Ico_File (  0?,             In
                 Ico_File_Path,  In
                 Ico_File_id);   Out
```

proc_01DA * File_Read_Text

```
File_Read_Text (  File_id,      In
                  Text,         Out
                  0?,           In
                  Count,        In
                  Bytes_Read);  Out
```

See also: File_Open, File_Seek.

proc_01E3 * Flash_Item_Read

```
Flash_Item_Read (  Record_Name,  In
                   Item_Name,    In
                   Buf_Adr,      In
                   Buf_Size,     In
                   Error_Code);  Out
```

proc_01E4 * Flash_Text_Read

```
Flash_Text_Read (  Record_Name,  In
                   Item_Name,    In
                   Text,         In
                   Text_Size,    In
                   Error_Code);  Out
```

proc_01E9 * File_Seek

```
File_Seek (  File_id,      In
             Seek_Value,   In
             Seek_Mode,    In
             Error_Code);  Out

Seek_Mode =   0  Absolute seeking
              1  Relative seeking
```

See also: Open_File.

proc_01EB * Fill_Mem

```
Fill_Mem (  Buf_Adr,  In
            Buf_Ofs,  In
            Value,    In
            Length);  In
```

proc_01ED * Open_Device_Volume

```
Open_Device_Volume (  Device_Volume_Name,
                      Client_Id,
                      Error_Code);

Device_Volume_Name =   "MLOAD", "LCARD", "RCARD", "MODEM", "SERIAL", "PARALLEL".
```

See also: Get_Device_Volume_Id.

proc_01EF * Eeprom_Write

```
Eeprom_Write (  Buf_Adr,      In
                Buf_Ofs,      In
                Count,        In
                Eeprom_Ofs,   In
                Error_Code);  Out

Buf_Ofs, Eeprom_Ofs and Count are specified in bits.
```

See also: Eeprom_Read.

proc_01F6 * Set_Panel_Page

```
Set_Panel_Page (  Panel_id,     In
                  Panel_Page);  In
```

proc_01FB * Widget_Set_Enum

```
Widget_Set_Enum (  Panel_id,     In
                   Widget_Name,  In
                   Enum_Value);  In
```

See also: Widget_Get_Enum.

proc_01FC * Widget_Set_Focus

```
Widget_Set_Focus (  Panel_id,      In
                    Widget_Name);  In
```

See also: Widget_Get_Focused.

proc_01FD * Widget_Set_Bitmap

```
Widget_Set_Bitmap (  Panel_id,      In
                     Widget_Name,   In
                     Bitmap_Name);  In
```

proc_0200 * Widget_Set_Value

```
Widget_Set_Value (  Panel_id,     In
                    Widget_Name,  In
                    Value);       In
```

See also: Widget_Get_Value.

proc_0204 * Widget_Set_Text

```
Widget_Set_Text (  Panel_id,     In
                   Widget_Name,  In
                   Text);        In
```

See also: Widget_Get_Text.

proc_0208 * Timer_Start

```
Timer_Start (  Timer_Mode?,    In
               Timeout,        In
               Event_Category, In
               Event_Code,     In
               Timer_id);      Out
```

See also: Timer_Stop.

proc_0209 * Enable_Trace

```
Enable_Trace (  Trace_Number,    In
                Trace_Enabled);  In

Trace_Enabled =   FALSE = 0  Disable
                  TRUE  = 1  Enable
```

proc_020B * Show_Panel

```
Show_Panel (  Panel_id);  In
```

See also: Hide_Panel.

proc_020C * String_Of_Space

Generate a string contains specified number of spaces.

```
String_Of_Space (  Space_String,  Out
                   Space_Count);  In
```

proc_020F * Trim_Right

Delete all empty spaces from right of input string.

```
Trim_Right (  Input_String,  In
              Trim_String);  Out
```

See also: Trim_Left.

proc_0210 * Lower_Case

```
Lower_Case (  Input_String,   In
              Lower_String);  Out
```

See also: Upper_Case.

proc_0212 * Str_To_Int

```
Str_To_Int (  String,  In
              Value);  Out
```

See also: Int_To_Str.

proc_0214 * Upper_Case

```
Upper_Case (  Input_String,    In
              Upper_String);   Out
```

See also: Lower_Case.

proc_0217 * Semaphore_Acquire

```
Semaphore_Acquire (  Semaphore_Var,   In
                     Semaphore_id);   Out
```

See also: Semaphore_Release.

proc_0218 * Tcs_Channel_Flags

```
Tcs_Channel_Flags (  Tcs_Adr,         In
                     Channel_Number,  In
                     Flags_Cmd,       In
                     Channel_Flags,   In/Out
                     Error_Code);     Out

Flags_Cmd =   0  Get channel flags
              1  Set channel flags
```

proc_0219 * Tcs_Swap_Channels

```
Tcs_Swap_Channels (  Tcs_Adr,          In
                     Channel_Number1,  In
                     Channel_Number2,  In
                     Swap_Mode);       In

Swap_Mode =   0  Channel 1 before channel 2
              1  Channel 1 after channel 2
              2  Swap channel 1 with channel 2
```

proc_021A * Tcs_Clear

```
Tcs_Clear (  Tcs_Adr);  In
```

proc_021B * Tcs_Read_Channel

```
Tcs_Read_Channel (  Tcs_Adr,              In
                    Channel_Number,       In
                    Channel_Flags,        Out
                    Channel_Transponder,  Out
                    Channel_Sid,          Out
                    Channel_Buf_Adr,      In
                    Channel_Video_Ecm,    Out
                    Channel_Audio_Ecm,    Out
                    Error_Code);          Out
```

Channel_Buf_Adr contains the address in the memory zone where the following information is found:

```
0   VIDEO_PID      (2 byte)
2   AUDIO_PID      (2 byte)
4   TELETEXT_PID   (2 byte)
6   SUBTITLES_PID  (2 byte)
8   PCR_PID        (2 byte)
```

proc_021C * Tcs_Get_Channel_Name

```
Tcs_Get_Channel_Name (  Tcs_Adr,         In
                        Channel_Number,  In
                        Channel_Name,    Out
                        Error_Code);     Out
```

proc_021D * Tcs_Next_Channel

```
Tcs_Next_Channel (  Tcs_Adr,         In
                    Channel_Number,  In/Out
                    Direction,       In
                    Exists,          In
                    Favorite,        In
                    Error_Code);     Out

Direction =   0  Previous channel
              1  Successive channel
```

```
Exists =   0  Not existing channel
           1  Existing channel

Favorite =   0  Not favorite channel
             1  Favorite channel
```

proc_021E * Tcs_Search_Channel

```
Tcs_Search_Channel (  Tcs_Adr,      In
                      Int?,         Out
                      Int?,         In
                      Int?,         In
                      Int?,         In
                      Tpt_Adr,      In
                      Error_Code);  Out
```

proc_021F * Tcs_Write_Channel

```
Tcs_Write_Channel (  Tcs_Adr,              In
                     Channel_Number,       In
                     Channel_Flags,        In
                     Channel_Transponder,  In
                     Channel_Sid,          In
                     Channel_Buf_Adr,      In
                     Error_Code);          Out
```

Channel_Buf_Adr contains the address in the memory zone where the following information is found:

```
0   VIDEO_PID      (2 byte)
2   AUDIO_PID      (2 byte)
4   VIDEO_ECM      (2 byte)
6   AUDIO_ECM      (2 byte)
8   PCR_PID        (2 byte)
```

proc_0220 * Tcs_Set_Channel_Name

```
Tcs_Set_Channel_Name (  Tcs_Adr,         In
                        Channel_Number,  In
                        Channel_Name,    In
                        Error_Code);     Out
```

proc_0223 * Get_Time

```
Get_Time (  Time_Hours,                 Out
            Time_Minutes,               Out
            Time_Seconds,               Out
            Time_Hundredths_Of_Second);  Out
```

Time_Hundredths_Of_Seconds is not supported by all systems.

See also: Get_Date.

proc_0225 * Tpt_Read_Transponder

```
Tpt_Read_Transponder (  Tpt_Adr,              In
```

```
                        Transponder_Number,    In
                        Transponder_Nid,       Out
                        Transponder_Tid,       Out
                        Transponder_Channels,  Out
                        Transponder_Buf_Adr,   In
                        Error_Code);           Out
```

Transponder_Buf_Adr contains the address in the memory zone where the following information is found:

```
0   FREQUENCY        (4 byte)
4   SYMBOL_RATE      (1 byte)
5   FEC              (1 byte)  0 = 1/2, 1 = 2/3, 2 = 3/4, 3 = 5/6, 4 = 7/8
6   POLARITY         (1 byte)  0 = Horizontal, 1 = Vertical
7   CMD_AUX          (1 byte)  0 = Low Band, 1 = High Band
8   ALIM_LNB         (1 byte)
9   CAR_TYP          (1 byte)
10  IQ_MODE          (1 byte)  0 = ?, 1 = ?
11  RF_SWITCH        (1 byte)  0 = Lnb A, 1 = Lnb B, 2..7 = ?
```

proc_0226 * Tpt_Search_Transponder

```
Tpt_Search_Transponder (  Tpt_Adr,             In
                          Transponder_Number,  Out
                          Transponder_Nid,     In
                          Transponder_Tid,     In
                          Error_Code);         Out
```

proc_0227 * Tpt_Write_Transponder

```
Tpt_Write_Transponder (  Tpt_Adr,              In
                         Transponder_Number,   In
                         Transponder_Nid,      In
                         Transponder_Tid,      In
                         Transponder_Channels, In
                         Transponder_Buf_Adr,  In
                         Error_Code);          Out
```

Transponder_Buf_Adr contains the address in the memory zone where the following information is found:

```
0   FREQUENCY        (4 byte)
4   SYMBOL_RATE      (1 byte)
5   FEC              (1 byte)  0 = 1/2, 1 = 2/3, 2 = 3/4, 3 = 5/6, 4 = 7/8
6   POLARITY         (1 byte)  0 = Horizontal, 1 = Vertical
7   CMD_AUX          (1 byte)  0 = Low Band, 1 = High Band
8   ALIM_LNB         (1 byte)
9   CAR_TYP          (1 byte)
10  IQ_MODE          (1 byte)  0 = ?, 1 = ?
11  RF_SWITCH        (1 byte)  0 = Lnb A, 1 = Lnb B, 2..7 = ?
```

proc_0228 * Write_Trace

```
Write_Trace (  Trace_Number,   In
               Trace_String);  In
```

proc_0231 * Wait_Event

Wait for a single event.

```
Wait_Event (  Event_Category,  In
              Event_Code,      In
              Timeout,         In
              Error_Code);     Out

Timeout =   -1  Infinite
        >   0   Finite (milliseconds)

Error_Code =    0  Event occurred
           <>   0  Timeout expired
```

See also: Wait_Multiple_Event, Get_On_Event_Info;

proc_0232 * Wait_Multiple_Events

Wait for a single event of a multiple events list.

```
Wait_Multiple_Events (  Events_List,   In
                        Events_Count,  In
                        Timeout,       In
                        Evt_Category,  Out
                        Evt_Code);     Out

Timeout =   -1  Infinite
        >   0   Finite (milliseconds)

Evt_Category =    0  Event occurred
             <>   0  Timeout expired
```

Events_Lists must be a *N x 2* matrix contains the following data:

```
Events_Lists[0][0] = Evt0_Category
Events_Lists[0][1] = Evt0_Code
Events_Lists[1][0] = Evt1_Category
Events_Lists[1][1] = Evt1_Code
Events_Lists[2][0] = Evt2_Category
Events_Lists[2][1] = Evt2_Code
```

...and so on.

See also: Wait_Event, Get_On_Event_Info;

proc_023B * Window_Draw_Line

Draw a single pixel width line.

```
Window_Draw_Line (  Window_id,  In
                    X1,         In
                    Y1,         In
                    X2,         In
                    Y2);        In
```

See also: Window_Set_Attr.

proc_023C * Window_Draw_Pixel

```
Window_Draw_Pixel (  Window_id,  In
                     X,          In
                     Y);         In
```

See also: Window_Set_Attr.

proc_023D * Window_Draw_Rect

Draw a rectangular outline.

```
Window_Draw_Rect (  Window_id,  In
                    Left,       In
                    Top,        In
                    Width,      In
                    Height);    In
```

Outline width is specified by WINDOW_ATTR_DRAW_SIZE attribute.

See also: Window_Fill_Rect, Window_Set_Attr.

proc_023E * Window_Draw_Box

Draw a rounded rectangular outline.

```
Window_Draw_Box (  Window_id,  In
                   Left,       In
                   Top,        In
                   Width,      In
                   Height);    In
```

Outline width is specified by WINDOW_ATTR_DRAW_SIZE attribute.

See also: Window_Fill_Box, Window_Set_Attr.

proc_023F * Window_Draw_Text

Draw a text box.

```
Window_Draw_Text (  Window_id,  In
                    X,          In
                    Y,          In
                    Text);      In
```

Text vertical justify is spefified by WINDOW_ATTR_TEXT_VERTICAL_JUSTIFY attribute (0 = Top, 1 = Bottom).

See also: Get_Text_Rect, Window_Set_Attr.

proc_0242 * Get_Display_Lut

```
Get_Display_Lut (  Display_Num,  In
                   Lut_Adr);     In
```

proc_0243 * Window_Get_Attr

```
Window_Get_Attr (  Window_id,    In
                   Window_Attr,  In
                   Attr_Value);  Out
```

See also: Window_Set_Attr.

proc_0244 * Get_Font_Id

```
Get_Font_Id (  Font_Name,  In
               Fond_id);   Out

Font_Name =   "fudemi", "videotex", ...
```

proc_0247 * Get_Text_Rect

```
Get_Text_Rec (  Font_id,       In
                Font_Size,     In
                Text,          In
                Text_Width,    Out
                Text_Height);  Out
```

See also: Get_Font_Id.

proc_0248 * Window_Disable_Drawing

```
Window_Disable_Drawing (  Window_id);  In
```

See also: Window_Enable_Drawing.

proc_0249 * Xor_Mem

```
Xor_Mem (  Buf_Adr,      In
           Buf_Ofs,      In
           Length,       In
           Xor_Result);  Out
```

proc_024A * Window_Fill_Rect

Draw a squared rectangular box.

```
Window_Fill_Rect (  Window_id,  In
                    Left,       In
                    Top,        In
                    Width,      In
                    Height);    In
```

See also: Window_Draw_Rect, Window_Set_Attr.

proc_024B * Window_Fill_Box

Draw a rounded rectangular box.

```
Window_Fill_Box (  Window_id,  In
                   Left,       In
                   Top,        In
                   Width,      In
                   Height);    In
```

See also: Window_Draw_Box, Window_Set_Attr.

proc_024D * Set_Display_Lut

```
Set_Display_Lut (  Display_Num,  In
                   Lut_Flag,     In
                   Lut_Adr);     In
```

```
Lut_Flag =   0  Set default application Lut
             1  Set specified Lut
```

proc_024E * Window_Set_Attr

```
Window_Set_Attr (  Window_id,    In
                   Window_Attr,  In
                   Attr_Value);  Out
```

See also: Window_Get_Attr.

proc_024F * Window_Set_Font

```
Window_Set_Font (  Window_id,   In
                   Font_id,     In
                   Font_Size);  In
```

See also: Get_Font_Id.

proc_0250 * **Window_Enable_Drawing**

```
Window_Enable_Drawing (  Window_id);  In
```

See also: Window_Disable_Drawing.

proc_0252 * Flash_Record_Delete

Delete a specified record in flash.

```
Flash_Record_Delete (  Record_Name,  In
                       Error_Code);  Out
```

See also: Flash_Clear.

proc_0268 * Volume_File_Info

```
Volume_File_Info (  Volume_id,    In
                    Module_Name,  In
                    File_Name,    In
                    Volume_id,    Out
                    File_Adr,     Out
```

```
                    File_Size,    Out
                    Error_Code);  Out
```

proc_02BC * Create_Region

```
Create_Region (  Region_id,    Out
                 Left,         In
                 Top,          In
                 Width,        In
                 Height,       In
                 Buf_Adr,      In
                 ?,            In
                 ?,            In
                 ?,            In
                 Error_Code);  Out
```

Buf_Adr contains the address in the memory zone where the following information is found:

```
0   0x00?   (2 byte)
2   0x04?   (2 byte)
4   LUT_ADR (4 byte)
```

proc_02BD * Free_Region

```
Free_Region (  Region_id,    In
               Error_Code);  Out
```

proc_02E5 * ?

```
? (  Transponder_Nid,  In
     Transponder_Tid,  In
     Channel_Sid,      In
     Error_Code);      In
```

proc_02E6 * Set_Ca_Filter

```
Set_Ca_Filter (  Ca_System_id,  In
                 Em_Adr,        In
                 Error_Code);   Out
```

Em_Adr contains the address in the memory zone where the following information is found:

```
0   OPI_TABLE_ADR  (4 byte)
4   MSD_TABLE_ADR  (4 byte)
8   DATE_TABLE_ADR (4 byte)
12  CA_SYSTEM_ID   (4 byte)
```

With null Em_Adr filters isn’t set.

proc_0303 * Tcs_Channel_Exists

```
Tcs_Channel_Exists (  Tcs_Adr,         In
                      Channel_Number,  In
                      Error_Code);     Out
```

proc_02AA * Device_Instance_List