Topfield's customizing API (TAP) for TF5000PVR series

Version : 1.22
2005-06-07

Contents

1. Overview

1.1 Introduction

The Topfiled's customizing Application Program Interface (TAP) is a suite for customizing your own TF5000PVR as what you want. Topfield offers the API and you can make some applications using it.

1.2 Background

After launching the TF4000PVR, we have got many helps from many end-users. We owe the enhancement of TF5000PVR to them. Many good advices rush till now. But, we found that it is hard to employ some of them because of the hardware restrictions of current TF4000PVR/TF5000PVR platform or someone's requests conflict with others.
We think that there will be many professional end-users who are willing to customize their own TF5000PVR and we would like to rely on them. Actually, the TAP is also available in TF4000PVR but we did not release its specifications. We just released some simple TAP applications. Now, we open TAP specifications in TF5000PVR and users can make more powerful applications than TF4000PVR's with high performance CPU and plenty of memory space.

1.3 Cautions

The TAP applications are saved in the HDD of TF5000PVR. It is highly possible that you have to format the HDD or do factory setting due to unexpected process after execution of TAP application.
Topfield has no responsibility for any kind of system error with modifying the sample code or your own TAP applications.

1.4 Restrictions

You may feel that it is insufficient to make good applications from the current API functions and this document. We will enrich the API functions and documents step by step.
The TAP applications will be executed in SDRAM even though they are saved in HDD.
So, the size of TAP application may be limited by the remained HEAP memory. In TF5000PVR, you can use 30MB of HEAP memory approximately.

2. How and what to install

The interpreter of TAP application is designed to support it based on GCC compiler for TAP. So, you should install the following compiler and tools.

2.1 What to install

1. CYGWIN
CYGWIN is “GCC for Windows” and you should install it AT FIRST because it supports the environment for using “GCC for TAP”.
You can get the latest version and install method from www.cygwin.com

2. GCC for TAP
It is a cross compiler for TAP and generates the execution code for TAP platform.
You can download it from www.i-topfield.com. But, it will be posted in some helping site for the time being. This is offered in ZIP file format.
After downloading you have to decompress it in directory that you installed CYGWIN. If you installed CYGWIN in c:\cygwin, you have to decompress it in c:\cygwin\usr\. If you encounter some dialog box asking overwrite while decompressing it, overwrite them.

3. Topfield’s customizing API (TAP) Library
It offers the interface to build the TAP application. You can find them in the root directory of the sample tap applications.

2.2 After installation

After installation, you have to set the path and parameter for compiler.
Let me assume that you have installed GCC at c:\cygwin and TAP at c:\work\tap. Then, you have to add following parameters in autoexec.bat (Win98)

```
SET path=%PATH%;C:\CTGWIN\USR\LOCAL\BIN

SET C_INCLUDE_PATH=C:\WORK\TAP;C:\CYGWIN\INCLUDE; C:\CYGWIN\USR\INCLUDE
SET CPLUS_INCLUDE_PATH=C:\WORK\TAP;C:\CYGWIN\INCLUDE; C:\CYGWIN\USR\INCLUDE
```

2.3 Description of TAP files

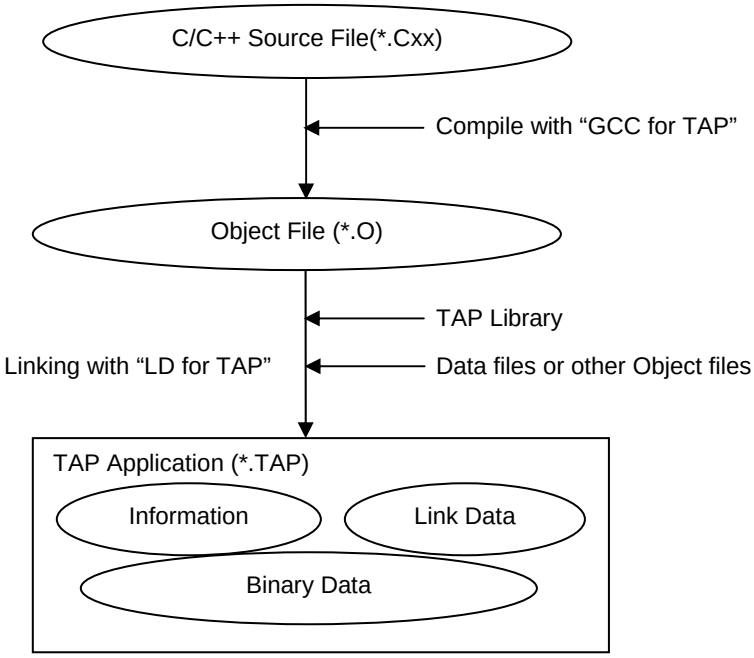
TAP\addpath.bat	Batch file for setting PATH parameters
TAP\libtap.a	TAP Library
TAP\tap.ld	Link definition file
TAP\gif.c	Sample source file of GIF file decoder
TAP\line.c	Sample source file of drawing LINE
TAP\tap.h	Header file of TAP main
TAP\font.h	Header file of OSD(On Screen Display) and Font
TAP\hdd.h	Header file of HDD related functions
TAP\key.h	Header file of remote controller functions

TAP\type.h	Header file of Type definition
TAP\win.h	Header file of Window
TAP\gif.h	Header file of GIF functions
TAP\FIRE	Directory of sample source code (Fire Clock)
TAP\GIF	Directory of sample source code (GIF Viewer)
TAP\LOGO	Directory of sample source code (LOGO)
TAP\MENUSND	Directory of sample source code (MENU Sound)
TAP\USBTEST	Directory of sample source code (USB Test)
TAP\EPG	Directory of sample source code (Quick EPG)
TAP\DirectOSD	Directory of sample source code (Direct OSD)
TAP\Plasma	Directory of sample source code (Plasma)
TAP\PACMAN	Directory of sample application (Game - Pacman)
TAP\ TimerList	Directory of sample source code (Timer)
TAP\ Background Slide Show	Directory of sample source code (Slide Show)
TAP\ ScreenCapture	Directory of sample source code (Screen Capture)
TAP\ How to Access Array of Pointer	Directory of sample source code (Access of pointer array)
TAP\MyOSD	Directory of sample source code (OSD)

3. How to build TAP application

The TAP applications are built and executed by the following procedures.

3.1 Building TAP Application



3.2 Download TAP Application to TF5000PVR series

You can download the TAP application file (*.TAP) to TF5000PVR series with Altair.exe to the Program Files directory. Download the TAP applications to be executed automatically after booting the HDD in the **Auto Start** directory.

3.3 Auto Start

The sub-directory of **Auto Start** in Program Files has the TAP applications to be executed automatically after booting the HDD. If you don’t want to execute the TAP applications in **Auto Start**, please press ‘ **0** ’ key after booting TF5000PVR series.

3.4 Building sample TAP applications

The following program will be called to make the TAP application. You can refer to build.bat of each sample source code directory. No makefile will be offered.

1. Mips-gcc.exe
   It makes object file through compiling the source.
   The following compile options have to be adopted.

   -mtap -mlong-calls -msoft-float
   (Refer to gcc.bat)

2. Mips-ld.exe
   It makes elf file through linking object files and TAP library
   On this time you have to use tap.ld
   It defines the memory structure and the path to search the library file at linking. It is recommended not to alter other part than SEARCH_DIR(…) because it may cause application error.

3. Mips-objcopy.exe
   It makes binary file with ELF format.
   This is the TAP application that you can execute in TF5000PVR
   The extension must be "**.TAP**".

3.5 Samples

A. Fire Clock
This example is designed to be executed such as TSR application and you can get the following hints from this example.
- Using OSD(On Screen Display) functions
- Getting system time
- Executing TSR mode

Executing the build.bat makes the binary file. Refer the applied options to the build.bat. After executing, pressing 'sleep' key will activate clock.

B. GIF Browser
You can get the following hints from this application.
- File read/write
- Using the functions related Window
- Getting the remote controller key data

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

C. Logo
You can get the following hints from this application.
- Using OSD(On Screen Display) functions
- Including the image data with a file

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

D. MENU Sound
You can get the following hints from this application.
- How to play PCM sound.

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

E. USB Test
You can get the following hints from this application.
- Communication with PC using USB.

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

F. Quick EPG
You can get the following hints from this application.
- How to get event information in TAP.
- How to handle key event.

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

G. Direct OSD

You can get the following hints from this application.

- How to access the OSD directly.

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

H. Plasma

You can get the following hints from this application.

- How to use floating-point operation.

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

I. TimerList

You can get the following hints from this application.

- How to control the timers.

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

J. Background Slide Show

You can get the following hints from this application.

- How to change the background image in MP3 play and radio.

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

K. ScreenCapture

You can get the following hints from this application.

- How to capture the live screen.

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

L. How to Access Array of Pointer

You can get the following hints from this application.

- How to access the array of pointers.

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

M. How to change infobox and volume bar (MyOSD)

You can get the following hints from this application.

- How to change infobox and volume bar

Executing the build.bat makes the binary file. Refer the applied options to the build.bat.

4. Writing TAP application

4.1 TAP_Main( )

The function of TAP_Main( ) is the first started function in TAP application. So, it should exist.

4.2. TAP_EventHandler(word event, dword param1, dword param2)

This function is event handler to process various events, for example remote controller key. This function must be in TAP and will be called every time if the events occurred .

word event : event type
dword param1, dword param2 : result of each event

event	comment	param1	param2	return value
EVT_IDLE	idle time	none	none	no effect
EVT_KEY	key input	key value	none	key value
EVT_UART	uart input	data value	none	no effect
EVT_RBACK	radio/mp3 background change	0 = Radio / MP3 Background drawing start 1 = Radio / MP3 Background clear	none	0 = default background draw. 1 = default clear 0xff = handled in TAP
EVT_SVCSTATUS	Change of Service Decoding status	Refer to the below table. (param1_of_EVT_SVCSTATUS)	none	no effect
EVT_VIDEO_CONV	Conversion of NTSC to/from PAL	Video Type 0 = PAL 1 = NTSC	Event Position 0 = Before the conversion 1 = After the conversion	

% Remarks
To prevent OSD problem for NTSC to/from PAL video mode, you have to erase the entire OSD before the conversion and re-draw the OSD after conversion.

param1_of_EVT_SVCSTATUS

bit	31	30 ~ 25	24	23~14	12~0
description	0 = Sub , 1 = Main	not used	0 = No Error, 1 = Error	not used	PID

The return value will be used as result of event in lower priority TAP or Firmware.
And then you have to return if you don't want to change it.

The following is basic form of the TAP_EventHandler

```
dword TAP_EventHandler( word event, dword param1, dword param2 )
{
        return param1
}
```

4.3 TAP_ID(ID_USERTAP)

Every TAP application needs unique ID and TAP allocations that have same ID can't execute at same time.
ID consists of DWORD(4byte) and the MSB bit should be set by 1.
The following is how to use in source codes.

```
#define ID_USERTAP 0x80000123   // user defined id , id = 0x0000123
TAP_ID( ID_USERTAP );
```

4.4 Program and Author information

Every TAP application has to include Program name, Author name, and Program description as following manners in source code.

```
TAP_PROGRAM_NAME("HELLOWORLD");              // maximum length is 64 byte

TAP_AUTHOR_NAME("MyName");                   // maximum length is 64 byte

TAP_DESCRIPTION("My first tap application");  // maximum length is 128 byte

TAP_ETCINFO( _DATE_ );
```

4.5 Running mode

There are 2 modes in running TAP applications. One is normal application mode and the other is TSR mode. The selection of running mode is decided by the return value of TAP_Main( ).

A. Application mode

The return value of TAP_Main( ) should be 0.
This mode means that the application ends at the end point of TAP_Main( ). The loaded application will be freed from memory and all the allocated resources will be returned when the application stops. It is easier to make a stable application than in TSR mode.

B. TSR mode

The return value of TAP_Main( ) should be 1.
This mode means that the application starts at the end point of TAP_Main( ). After returning from the TAP_Main( ), the TAP application shares the resources with the firmware of TF5000PVR. So, the TAP application should not occupy too much operation time and resources to cause the malfunction of TF5000PVR's firmware.
TAP_EventHandler( ) is called with EVT_IDLE event at every IDLE time at this mode.
To exit the TAP application from this mode, call TAP_Exit( ).

Important Note

Static initialized variables of the function pointer or the variable pointer cannot be permitted due to complier BUG.

For example, the below pArray will not be initialize.

```
char pA[] = { 0, 1, 2, 3 };
char pB[] = { 4, 5, 6, 7 };
```

```
void *pArray[] =
{
        pA,
        pB,
};
```

Also the member pointer of structure cannot be initialized in declaration part. In this case, you have to make the function to initialize the pointer variable.

5. Descriptions of TAP functions

5.1 System Function

TAP_SystemProc

void TAP_SystemProc(void)

• It SHOULD be called continuously to get the system resources in application mode.

It should be called repeatedly to process the RS232 data or remote controllers that are firmware's resources. If it is not called in application mode, the TAP application cannot use the RS232 data or remote controller.

It is HIGHLY recommended NOT to call this function in TSR mode because all the resources can be available even without this function in TSR mode.

TAP_GetState

void TAP_GetState (dword *state, dword *subState)

• It gets the current status of firmware.

state : the state of firmware

subState : detailed state of firmware

TAP_ExitNormal

void TAP_ExitNormal(void)

• It makes the firmware to exit from the normal state.

The normal state means that the system can display the info box, volume bar or any other OSD. All the OSD regions are initialized and cleared after this function.

This function SHOULD be called to use OSD functions in TAP application.

TAP_EnterNormal

void TAP_EnterNormal(void)

• It makes the firmware to enter the normal state.

The firmware can display info box or any other OSD after this function.

TAP_Exit

void TAP_Exit(void)

• It ends the TAP application.

This function SHOULD be called to finish the TAP application in TSR mode.

TAP_KbHit

byte TAP_KbHit(void)

• It checks if there is a new data in RS232 port.

It returns 0 if there is no data. If a new data is available, it returns non-zero value.

TAP_GetCh

byte TAP_GetCh(void)

- It returns one byte data from RS232 port.

TAP_PutCh

void TAP_PutCh(byte ch)

- It puts one byte data to RS232 port.

TAP_Print

void TAP_Print(const void *fmt, ...)

- It puts a formatted string data to RS232 port.

It will be useful in case of debugging the TAP application.

*TAP_GetSystemVar

int (*TAP_GetSystemVar)(int varId)

- It returns the system stting information.

varId : Identification number of the following system variables.
* The Identification number is defined in the **TAP.H**.

For the detailed information, refer the **TAP.H**.

SYSVAR_Vol	Volome
SYSVAR_OsdLan	Menu Language
SYSVAR_SubLan	Subtitle Language
SYSVAR_AudLan	Audio Language
SYSVAR_TvType	Tv Type
SYSVAR_VideoOut	Video Output
SYSVAR_ScartType	Scart Type
SYSVAR_TvRatio	Tv Aspect Ratio
SYSVAR_16_9_Display	16:9 Display Format
SYSVAR_SoundMode	Sound Mode
SYSVAR_UhfType	UHF Type
SYSVAR_UhfCh	UHF Channel
SYSVAR_OsdAlpha	OSD Transparency
SYSVAR_IBoxTime	Info Box Display Time
SYSVAR_IBoxPos	Info Box Position
SYSVAR_Timeshift	Time shifting

*TAP_SetSystemVar

int (*TAP_SetSystemVar)(int varId, int value)

- It sets the system variable.

It returns the set value.

varId : Identification number of the following system variables.
value : the value to be set to the system variable.

* The Identification number is defined in the **TAP.H**.
For the detailed information, refer the **TAP.H**.

SYSVAR_Vol	Volome
SYSVAR_OsdLan	Menu Language
SYSVAR_SubLan	Subtitle Language
SYSVAR_AudLan	Audio Language
SYSVAR_TvType	Tv Type
SYSVAR_VideoOut	Video Output
SYSVAR_ScartType	Scart Type
SYSVAR_TvRatio	Tv Aspect Ratio
SYSVAR_16_9_Display	16:9 Display Format
SYSVAR_SoundMode	Sound Mode
SYSVAR_UhfType	UHF Type
SYSVAR_UhfCh	UHF Channel
SYSVAR_OsdAlpha	OSD Transparency
SYSVAR_IBoxTime	Info Box Display Time
SYSVAR_IBoxPos	Info Box Position
SYSVAR_Timeshift	Time shifting

*TAP_WriteSystemVar

```
int     (*TAP_WriteSystemVar)( void )
```

• It saves the value in the EEPROM.
If it is not called after changing the system variables by TAP_SetSystemVar( ), the variables cannot be saved.
return value : 0 for Success,
other value for ERROR

*TAP_GenerateEvent

```
void    (*TAP_GenerateEvent)( word evtCode, dword param1, dword param2 )
```

• It makes an event such as input from the RCU.
EvtCode : event (only the EVT_KEY is supported now.)
param1, param2 : The event value

TAP_SetBaudRate

```
void    TAP_SetBaudRate( dword baudRate )
```

• It sets the baud rate of serial port.
baudRate : 150 / 300 / 600 / 1200 / 2400 / 4800 / 9600 / 19200 / 28800 / 38400 / 56000 / 57600 / 76800 / 115200 / 128000 (bps)

TAP_Vfd_GetStatus

int TAP_Vfd_GetStatus(void)

• It detects the availability of VFD.

return value : The status of VFD

return value	comment
VFD_NotSupport (0)	No VFD
VFD_LedEmulMode (1)	VFD works in LED emulation mode. It does not support TAP.
VFD_VfdMode (2)	VFD works in VFD Mode.

TAP_Vfd_Control

void TAP_Vfd_Control(bool underTapCtrl)

• It gets the control over the VFD.

underTapCtrl : TRUE = Controlled by TAP, FALSE = Controlled by firmware

TAP_Vfd_SendData

void TAP_Vfd_SendData(byte *data, byte dataLen)

• It outputs the data to the VFD.

data : The data to be sent to the VFD

dataLen : The length of data

Remarks – The VFD module which is employed in TF5500PVR is GU140x16G-7002 from NORITAKE ITRON . You may find its specifications from the search engine.
http://www.google.com/search?hl=en&q=GU140x16G+filetype%3Apdf&btnG=Google+Search

TAP_PrintSys

void TAP_PrintSys(const void *fmt, ...);

• It puts a formatted string data to RS232 port as TAP_Print does.

Contrary to TAP_Print it does not insert '\r' in front of '\n'.

Remarks : Some compiler options may insert '\r' in front of '\n' automatically. So, you have to change the options in this case.

TAP_PutByte

void TAP_PutByte(byte c);

• It puts one byte data to RS232 port as TAP_PutCh does.

Contrary to TAP_PutCh it does not insert '\r' in front of '\n'.

5.2 Common Function

TAP_SetSoundLevel

word TAP_SetSoundLevel(byte soundLevel)

- It adjusts the sound volume.

  soundLevel : 0 ~ 31

TAP_GetTime

word TAP_GetTime(word *mjd, byte *hour, byte *min, byte *sec)

- It gets the current time.

  mjd : date information in MJD format

  hour : hour (0 ~ 23)

  min : minute (0 ~ 59)

  sec : second (0 ~ 59)

TAP_MemInfo

void TAP_MemInfo(dword *heapSize, dword *freeHeapSize, dword *availHeapSize)

- It finds the usable HEAP information.

  heapSize : total HEAP size

  freeHeapSize : free HEAP size

  availHeapSize : available HEAP size

TAP_MemAlloc

void* TAP_MemAlloc(dword size)

- It allocates the memory at the HEAP.

  size : the memory size to be allocated

  return value : the allocated memory address

CAUTION : This function SHOULD be preceded by the TAP_MemInfo( ) to check the memory size.

TAP_MemFree

byte TAP_MemFree(const void *addr)

- It frees the allocated memory at the HEAP.

  *addr : the start address to be freed

TAP_GetTick

dword TAP_GetTick(void)

- It returns the counter value from the power on.

The counter value is increased by 1 at every 10ms.

TAP_Delay

void TAP_Delay(dword dm10)

- It delays the TAP application by the specified time.
  dm10 : the delay time in 10ms unit

TAP_ExtractMjd

byte TAP_ExtractMjd(word mjd, word *year, byte *month, byte *day, byte *weekDay)

- It gets the year, month, day and weekday from the date information of MJD format.
  mjd : date information in MJD format
  year : year
  month : month
  day : day
  weekDay : weekday
  return value : 1 if the input MJD data is normal. If it is abnormal value, it returns 0

TAP_MakeMjd

word TAP_MakeMjd(word year, byte month, byte day)

- It makes the MJD format from the year, month, and day.
  return value : date information in MJD format

TAP_Sin

long TAP_Sin(long mul, long th)

- It calculates the value of sine function.
  mul : parameter for multiplying
  th : angle degree between 0 to 360
  return value : mul * sin(th)

TAP_Cos

long TAP_Cos(long mul, long th)

- It calculates the value of cosine function.
  mul : parameter for multiplying
  th : angle degree between 0 to 360
  return value : mul * cos(th)

TAP_SPrint

void TAP_SPrint(void *str, const void *fmt, ...)

- It puts the formatted string to str.

TAP_GetSignalLevel

int TAP_GetSignalLevel (void)

- It returns the signal level of current transponder.
  return value : signal level (0 ~ 100 %)

TAP_GetSignalQuility

int TAP_GetSignalQuility (void)

•It returns the signal quality of current transponder.

return value : signal quility (0 ~ 100 %)

TAP_SetInfoboxTime

int TAP_SetInfoboxTime(int sec)

• It controls the display time of Infobox.

sec : displaying time (0 ~ 30)

TAP_PlayPCM

int TAP_PlayPCM(void *pcmdata, int size, int freq, void (*Callback)(void))

• It plays PCM data.

pcmdata : pointer of PCM data

size : size of PCM data

freq : frequency of PCM

Callback : It will be called after out the PCM data .

*TAP_CaptureScreen

int (*TAP_CaptureScreen)(int mainSub, TYPE_VideoBank *videoBank)

• It captures the picture.

mainSub : 1 – main screen, 0 – sub screen

videoBank : The pointer of TYPE_VideoBank that has the information for captured picture.

return value : 0 for Success,
other value for ERROR

The structure of the TYPE_VideoBank is as follows.

```
typedef struct
{
        byte *yAddress
        byte *cAddress
        word width
        word height
} TYPE_VideoBank
```

(Refer the **ScreenCapture** among the examples.)

5.3 OSD (On Screen Display) Function

Screen Resolution and Color

0 <= x (width) < 720
0 <= y (height) < 576
16bit Colors (ARGB = 1555)

Common Parameters of OSD or OSD String functions

word rgn : Region Index
dword x : the start value in X coordinate
dword y : the start value in Y coordinate
dword w : Width
dword h : Height
byte color : Color Index
byte lutIdx : LUT Index (only used in 256 color data)

TAP_Osd_Create

word TAP_Osd_Create(dword x, dword y, dword w, dword h, byte lutIdx, int flag)

- It creates a region.

x, y : starting poison of the region. (upper, left)
w, h : the width and height of the region.
LutIdx : The LUT number used in the region. LUT is assigned by TAP_Osd_SetLut( ).
flag : Osd creation flag.
return value : region number of created region.

flag	comment
OSD_Flag_MemRgn	make virtual region in memory instead of screen
OSD_Flag_256	make 256 color region. default is 16bit color
OSD_Flag_Plane2	make region in Plane2 instead of Plane1

TAP_Osd_Delete

word TAP_Osd_Delete(word rgn)

- It deletes the region.

TAP_Osd_Move

word TAP_Osd_Move(word rgn, dword x, dword y)

- It moves the region to the new position.

TAP_Osd_FillBox

word TAP_Osd_FillBox(word rgn, dword x, dword y, dword w, dword h, dword color)

- It draws the filled box with the assigned color.

TAP_Osd_SetLut

word TAP_Osd_SetLut(byte lutIdx, byte lut[][4])

- It assigns the Color LUT.

lutIdx : LUT Index
lut : the start address of LUT contents
return value : If 0, it is successful. Else, there is an error.

CAUTION : The size of lut must be 256*4.

TAP_Osd_SetTransparency

word TAP_Osd_SetTransparency(word rgn, char rt)

- It sets the transparency level of a region.

rt : transparency level between 0 to 63
return value : If 0, it is successful. Else, there is an error.

CAUTION : In TF5000PVR, you cannot set each transparency in the individual region. Only the total transparency can be changed with this function.

TAP_Osd_DrawRectangle

word TAP_Osd_DrawRectangle(word rgn, dword x, dword y, dword w, dword h, dword t, dword color)

- It draws a rectangle.

TAP_Osd_DrawPixmap

word TAP_Osd_DrawPixmap(word rgn, dword x, dword y, dword w, dword h, void *pixmap, bool sprite, byte dataFormat)

- It draws a 16bit (32768 color) or 8bit (256 color) pixel image.

*pixmap : the start address of image data
sprite : FALSE - All pixel data will be drawn on the region.
TRUE - The pixel data with color index 0 will not be drawn on the region.
dataFormat : pixel data format (OSD_1555 or OSD_256)
return value : If 0, it is successful. Else, there is an error.

TAP_Osd_PutGd

int TAP_Osd_PutGd(word rgn, int x, int y, TYPE_GrData * gd, bool sprite)

- It displays the pixel data that compressed with TYPE_GrData type.

This data type can be made using MakeGd.exe. It can compress 70% approximately The MakeGd.exe will be released at the next time.

gd : pixel data of TYPE_GrData type
sprite : FALSE - All pixel data will be drawn on the region.
TRUE - The pixel data with color index 0 will not be drawn on the region.
dataFormat : pixel data format (OSD_1555 or OSD_256)
return value : If 0, it is successful. Else, there is an error.

CAUTION : The member pointer of structure should not be initialized in the declaration part. So, you have to make the sub-routine that initializes TYPE_GrData and member of this structure. (Please refer to Logo.c in sample source codes)

TAP_Osd_Copy

word TAP_Osd_Copy(word srcRgnNum, word dstRgnNum, dword srcX, dword srcY, dword w, dword h, dword dstX, dword dstY, bool sprite)

- It copies some portion of a region to other region or to any other position in the same region.

srcRgnNum : the index of source region
dstRgnNum : the index of destination region
srcX : the start position in X coordinate of source to be copied
srcY : the start position in Y coordinate of source to be copied
dstX : the start position in X coordinate of destination to copy
dstY : the start position in Y coordinate of destination to copy
sprite : FALSE - All pixel data will be drawn on the region.
TRUE - The pixel data with color index 0 will not be drawn on the region.

TAP_Osd_SaveBox

byte* TAP_Osd_SaveBox(word rgn, dword x, dword y, dword w, dword h)

- It saves some portion of a region to memory.

return value : the start address of memory to save. The memory SHOULD be freed after using it with TAP_MemFree( ).

TAP_Osd_RestoreBox

void TAP_Osd_RestoreBox(word rgn, dword x, dword y, dword w, dword h, byte *data)

- It displays the data to the specified position.

The data can be saved by the function TAP_Osd_SaveBox( ).

TAP_Osd_GetPixel

word TAP_Osd_GetPixel(word rgn, dword x, dword y, byte *pix)

- It reads the pixel value.

TAP_Osd_PutPixel

word TAP_Osd_PutPixel(word rgn, dword x, dword y, byte pix)

- It writes the pixel value.

TAP_SysOsdControl

void TAP_SysOsdControl(TYPE_TapSysOsdId osdId, bool ctrl)

- It controls the system OSD such as infobox, volume bar, PVR info bar and service status bar.

osdId : ID number of the specific system OSD object.
0 : SYSOSD_InfoBox Information Box
1 : SYSOSD_PvrInfo PVR Information Bar
2 : SYSOSD_VolumeBar Volume Bar

3 : SYSOSD_ServiceStatus Service Status Bar
ctrl : TRUE = show, FALSE = hide

TAP_Osd_GetBaseInfo

int TAP_Osd_GetBaseInfo(TYPE_OsdBaseInfo *info);

• It gets the physical information to access the real memory of OSD. The OSD can be drawn by accessing this address.

*info : OSD base information
Return value : 0

```
typedef struct
{
    void *eAddr;            // even address (the address of top field)
    void *oAddr;            // odd address (the address of bottom field)
    word width;
    word height;
    word bytePerPixel;      // 256 color = 1, ARGB1555 = 2, ARGB8888 = 4
    word dataFormat;
    word reserved[4];
} TYPE_OsdBaseInfo;
```

TAP_Osd_GetBox

int TAP_Osd_GetBox(word rgn, dword x, dword y, dword w, dword h, void *data);

• It copies the memory content of the specified region to “data”.

Return value : If 0, it is successful. Else, there is an error.
Remarks : The *data should be allocated to the same size of the region.

TAP_Osd_PutBox

int TAP_Osd_PutBox(word rgn, dword x, dword y, dword w, dword h, const void *data, bool sprite, byte dataFormat);

• It copies the content of “data” to the real memory of the specified region.

sprite : FALSE – All pixel data will be drawn on the region.
TRUE – The pixel data with color index 0 will not be drawn on the region.
dataFormat : pixel data format (OSD_1555 or OSD_256)
Return value : If 0, it is successful. Else, there is an error.

TAP_Osd_Zoom

int TAP_Osd_Zoom(int xzoom, int yzoom);

• It adjusts the zoom factor in the OSD plane.

xzoom : The zoom factor in x-axis
yzoom : The zoom factor in y -axis
Return value : 0

5.4 OSD String Function

TAP_Osd_PutS

dword TAP_Osd_PutS(dword rgn, dword x, dword y, dword maxX, const char *str, dword fcolor, dword bcolor, byte fntType, byte fntSize, byte bDot, byte align)

- It prints the string on the screen.

maxX : the maximum value in X coordinate of output area. width = maxX – X + 1
fcolor : character color. If 0xff, it will be transparent.
bcolor : background color. If 0xff, it will be transparent.
fntType : font type. It SHOULD be 0.
fntSize : font size. FNT_Size_1419, FNT_Size_1622 or FNT_Size_1926
bDot : It decides to print "…" in case that the string is over the maxX.
align : align method. ALIGN_LEFT, ALIGN_CENTER or ALIGN_RIGHT

TAP_Osd_GetW

dword TAP_Osd_GetW(const char *str, byte fntType, byte fntSize)

- It calculates the output width with the specific font.

TAP_Osd_PutString

void TAP_Osd_PutString(dword rgn, dword x, dword y, dword maxX, const char *str, dword fcolor, dword bcolor, byte fntType, byte fntSize, byte nextLine)

- It prints the string on the screen.

This function is used to display the special characters.

maxX : the maximum value in X coordinate of output area. width = maxX – X + 1
fcolor : character color. If 0xff, it will be transparent.
bcolor : background color. If 0xff, it will be transparent.
fntType : font type. It SHOULD be 0.
fntSize : font size. FNT_Size_1419, FNT_Size_1622 or FNT_Size_1926
nextLine : It decides to print the new line character('\n').

TAP_Osd_DrawString

void TAP_Osd_DrawString(const char *str, dword dstWidth, dword color, byte *dest, dword maxWidth, byte fntType, byte fntSize)

- It prints the string on the memory.

dstWidth : the width of the memory
color : character color
*dest : the start address of memory
maxWidth : the maximum width to print

TAP_Osd_PutStringAf

void TAP_Osd_PutStringAf(dword rgn, dword x, dword y, dword maxX, const char *str, dword fcolor, dword bcolor, byte fntType, byte fntSize, byte nextLine, byte lut[][4])

- It draws the string with anti-flickering color on the screen.
  maxX : the maximum value in X coordinate of output area. width = maxX – X + 1
  fcolor : character color. If 0xff, it will be transparent.
  bcolor : background color. If 0xff, it will be transparent.
  fntType : font type. It SHOULD be 0.
  fntSize : font size. FNT_Size_1419, FNT_Size_1622 or FNT_Size_1926
  nextLine : It decides to print the new line character('\n').
  lut : the LUT for anti-flickering

5.5 HDD Function

TAP_Hdd_TotalSize

```
dword    TAP_Hdd_TotalSize( void )
```

- It gets the total capacity of HDD.

TAP_Hdd_FreeSize

```
dword    TAP_Hdd_FreeSize( void )
```

- It gets the usable capacity of HDD.

TAP_Hdd_Fopen

```
TYPE_File*        TAP_Hdd_Fopen( char *name )
```

- It opens the file whose name is the string pointed to by name
  name : file name
  return value : pointer to the object controlling the file. In case of error, it returns 0.

TAP_Hdd_Fread

```
dword    TAP_Hdd_Fread( void *buf, dword blksize, dword blk, TYPE_File *fp )
```

- It reads blk elements of blksize bytes each from the file specified by fp into the buffer specified by buf.
  buf : pointer of the buffer for reading.
  blksize : the data size per block.
  blk : the number of block to be read.
  *fp : pointer to the object controlling the file.
  Total reading bytes = blksize * blk

TAP_Hdd_Fwrite

```
dword    TAP_Hdd_Fwrite( void *buf, dword blksize, dword blk, TYPE_File *fp )
```

- It writes blk elements of blksize bytes each to the file specified by fp.
  buf : pointer of the buffer for writing.
  blksize : the data size per block.
  blk : the number of block to be written.
  *fp : pointer to the object controlling the file.
  Total writing bytes = blksize * blk

TAP_Hdd_Fclose

```
void       TAP_Hdd_Fclose( TYPE_File *fp )
```

- It closes the file fp.

TAP_Hdd_Flen

```
dword    TAP_Hdd_Flen( TYPE_File *fp )
```

- It returns the number of bytes in the opened file specified by fp.

TAP_Hdd_Ftell

```
dword    TAP_Hdd_Ftell( TYPE_File *fp )
```

• It returns the current read/write position of the file specified by fp.
This position defines the character that will be read or written by the next I/O operation on the file.

TAP_Hdd_Fseek

```
dword    TAP_Hdd_Fseek( TYPE_File *fp, long pos, long where )
```

• It changes the read/write position of the file specified by fp.
This position defines the character that will be read or written on the next I/O operation on the file**.**
pos : offset from the file location given by 'where'
where : file location. The value are as follows.
SEEK_SET : File beginning
SEEK_CUR : Current file pointer position
SEEK_END : End of file

TAP_Hdd_Delete

```
word     TAP_Hdd_Delete( char *filename )
```

• It deletes the file or folder whose name is the string pointed to by filename.

*TAP_Hdd_Rename

```
bool     (*TAP_Hdd_Rename)( char *oldName, char *newName )
```

• It changes the file name.
return value : TRUE = Success, FALSE = False

TAP_Hdd_Exist

```
bool     TAP_Hdd_Exist( char *filename )
```

• It checks the existence of the specified file.

TAP_Hdd_Create

```
dword    TAP_Hdd_Create( char *name, byte attr )
```

• It creates a file or a folder
name : file name
attr : ATTR_NORMAL : data file
ATTR_PROGRAM : program file
ATTR_FOLDER : folder

TAP_Hdd_ChangeDir

```
word     TAP_Hdd_ChangeDir( char *dir )
```

• It changes the current directory into the specified directory.
The path must be relative to the current directory.
dir : directory name. In case of "..", it is the parent directory. For "/", it is ROOT directory.

TAP_Hdd_FindFirst

```
dword    TAP_Hdd_FindFirst( TYPE_File *file )
```

• It starts to scan the file in the current directory.
file : the pointer of TYPE_File which the file information is saved to.
return value : total files in the current directory

TAP_Hdd_FindNext

```
dword    TAP_Hdd_FindNext( TYPE_File *file )
```

• It continues to scan the file in the current directory.
file : the pointer of TYPE_File which the file information is saved to.
return value : If 0, there is no file. If 1, there is a file.

*TAP_Hdd_PlayTs

```
int        (*TAP_Hdd_PlayTs)( char *name )
```

• It plays the recorded files.
The active folder should be set to the folder that contains the recorded files to be played back.
(To move the active folder, use TAP_Hdd_ChangeDir function.)
name : filename of recorded service file.
return value : 0 : Success, other value : ERROR

*TAP_Hdd_StopTs

```
int        (*TAP_Hdd_StopTs)( void)
```

• It stops the playback.
return value : 0 : Success, other value : ERROR

*TAP_Hdd_PlayMp3

```
 int        (*TAP_Hdd_PlayMp3)( char *name )
```

• It plays the MP3 files.
The active folder should be set to the folder, which contains the MP3 files to be played.
(To move the active folder, use TAP_Hdd_ChangeDir function.)
name : filename of MP3 file.
return value : 0 : Success, other value : ERROR

*TAP_Hdd_StopMp3

```
int        (*TAP_Hdd_StopMp3)( void )
```

• It stops the MP3 files.
return value : 0 : Success, other value : ERROR

TAP_Hdd_StartRecord

```
bool       TAP_Hdd_StartRecord( void )
```

• It records the current watching service. (Instant recording)
return value : TRUE = Success, FALSE = Failure

TAP_Hdd_StopRecord

```
bool    TAP_Hdd_StopRecord( byte recSlot )
```

• It stops the recording.

recSlot : the recording slot number to be stopped

return value : TRUE = Success, FALSE = Failure

TAP_Hdd_GetRecInfo

```
bool    TAP_Hdd_GetRecInfo( byte recSlot, TYPE_RecInfo *recInfo )
```

• It gets the recording information.

recSlot : the recording slot number

recInfo : pointer of TYPE_RecInfo where the recording information is saved.

For the more information about TYPE_RecInfo, please refer the **TAP.H**.

return value : TRUE = Success, FALSE = Failure

TAP_Hdd_GetPlayInfo

```
bool    TAP_Hdd_GetPlayInfo(TYPE_PlayInfo *playInfo )
```

• It gets the playback information.

playInfo : pointer of TYPE_PlayInfo where the playback information is saved.

For the more information about TYPE_ PlayInfo, please refer the **TAP.H**.

return value : TRUE = Success, FALSE = Failure

TAP_Hdd_GetHddID

```
bool    TAP_Hdd_GetHddID(char *modelNo, char *serialNo);
```

• It gets the model number and serial number of HDD.

Return value : TRUE=success, FALSE=fail

Remarks : The modelNo should be allocated to the size of 40 bytes and serialNo should be 20 bytes.

TAP_Hdd_ChangePlaybackPos

```
bool    TAP_Hdd_ChangePlaybackPos( dword blockPos );
```

• It changes the playback position for the specified point. The total block can be gotten from TAP_Hdd_GetPlayInfo().

blockPos : The new position

Return value : TRUE – Success, FALSE - Fail

TAP_Hdd_GotoBookmark

```
bool    TAP_Hdd_GotoBookmark( void );
```

• It changes the playback position for the bookmark position which is the nearest one to the right direction.

Return value : TRUE – Success, FALSE - Fail

TAP_Hdd_SetBookmark

```
bool    TAP_Hdd_SetBookmark( void );
```

• It sets the bookmark at the current playback position.

blockPos : The new position to be set as bookmark
Return value : TRUE – Success, FALSE - Fail

5.6 Window Function

Common Parameters of Window functions

*win : The structure pointer of TYPE_Window
byte fntType : Font type. FNT_Type_Ar or FNT_Type_Ba
byte fntSize : Font Size. FNT_Size_wwhh
dword x : the start position in X coordinate
dword y : the start position in Y coordinate
dword w : Width
dword h : Height
bytecolor : Color

TAP_Win_Create

```
void TAP_Win_Create( TYPE_Window *win, word rgn, dword x, dword y, dword w, dword h, byte save, byte bScr )
```

- It sets the basic parameters of window and draws it on the screen.

save : It decides to save the background image before the window is drawn.
bScr : It decides to show the scroll bar.

TAP_Win_SetTitle

```
void TAP_Win_SetTitle( TYPE_Window *win, const char *str, byte fntType, byte fntSize )
```

- It draws the title of window.

TAP_Win_SetColor

```
void TAP_Win_SetColor( TYPE_Window *win, byte titleBack, byte titleText, byte bodyBack, byte bodyText, byte border, byte shadow, byte dark, byte light )
```

- It sets the color of window.

titleBack : the background color of Title Bar
titleText : the text color of Title Bar
bodyBack : the background color of Body
bodyText : the text color of Body
shadow : It decides to show the shadow effect of the window.
dark : the color of the dark side to effect 3D
light : the color of the bright side to effect 3D

CAUTION : This function SHOULD be called before TAP_Win_Create( ) is called.

TAP_Win_SetDefaultColor

```
void TAP_Win_SetDefaultColor( TYPE_Window *win )
```

- It sets the color LUT of window to default one.

CAUTION : This function SHOULD be called before TAP_Win_Create( ) is called if you want the default CLUT.

TAP_Win_Draw

```
void    TAP_Win_Draw( TYPE_Window *win )
```

- It draws the window.

TAP_Win_Delete

```
dword   TAP_Win_Delete( TYPE_Window *win )
```

- It deletes the window.

TAP_Win_SetFont

```
void    TAP_Win_SetFont( TYPE_Window *win, byte fntType, byte fntSize )
```

- It selects the font for list mode.

TAP_Win_AddItem

```
void    TAP_Win_AddItem( TYPE_Window *win, char *str )
```

- It adds the optional strings after changing to the list mode.

TAP_Win_GetSelection

```
dword   TAP_Win_GetSelection( TYPE_Window *win )
```

- It gets the selected item number in the list mode.
  return value : the index of the selected item

TAP_Win_SetSelection

```
void    TAP_Win_SetSelection( TYPE_Window *win, dword pos )
```

- It changes the item to be selected in the list mode.
  pos : the index of the item to be selected

TAP_Win_Action

```
void    TAP_Win_Action( TYPE_Window *win, dword key )
```

- The window works by the remote controller key in the list mode.
  key : remote controller key

5.7 Channel Function

TAP_Channel_GetTotalNum

```
int     TAP_Channel_GetTotalNum( int *nTvSvc, int *nRadioSvc )
```

- It gets the number of TV and Radio services.
  - *nTvSvc : number of TV services
  - *nRadioSvc : number of Radio services
  - Return value : 0

TAP_Channel_GetInfo

```
int     TAP_Channel_GetInfo( int chType, int chNum, TYPE_TapChInfo *chInfo )
```

- It gets the information of the specified channel.
  - chType : Type of channel. It is defined in Tap.h.
  - chNum : Channel number
  - *chInfo : Channel information.
  - Return value : : If 0, there is no errors.

TAP_Channel_GetCurrent

```
int     TAP_Channel_GetCurrent( int *tvRadio, int *chNum )
```

- It gets the service type and number of the current service.
  - *tvRadio : Channel type
  - *chNum : Channel Number
  - Return value : 0

TAP_Channel_Start

```
Int     TAP_Channel_Start( int mainSub, int chType, int chNum )
```

- It changes the current service to the specified channel.
  - mainSub : 1 – main screen, 0 – sub screen
  - chType : Channel type
  - chNum : Channel number
  - Return value : : If 0, there is no errors.

TAP_Channel_Scale

```
int     TAP_Channel_Scale( int mainSub, long x, long y, long w, long h, bool anim )
```

- It changes the size of screen.
  - mainSub : 1 – main screen, 0 – sub screen
  - x, y : upper-left coordinate of the screen
  - w,h : width and height of the screen.
  - anim : if TRUE, it is scaled smoothly.
  - Return value : : If 0, there is no errors.

TAP_Channel_IsPlayable

int TAP_Channel_IsPlayable(int mainSub, int chType, int chNum)

- It checks the specified channel whether it is possible to decode

  mainSub : 1 – main screen, 0 – sub screen

  chType : Channel type

  chNum : Channel number

  Return value : : If 0, there is no errors.

TAP_Channel_Move

int TAP_Channel_Move(int mainSub, int dir)

- It moves the current channel.

  mainSub : 1 – main screen, 0 – sub screen

  dir : direction of zapping. (-1 = decrease channel number, 1 = increase channel number)

  Return value : : If 0, there is no errors.

TAP_Channel_Stop

int TAP_Channel_Stop(int mainSub)

- It stops the current channel.

  mainSub : 1 – main screen, 0 – sub screen

  Return value : : If 0, there is no errors.

TAP_Channel_GetTotalAudioTrack

Int TAP_Channel_GetTotalAudioTrack(void)

- It gets the number of audio tracks.

  Return value : the number of audio tracks.

*TAP_Channel_GetAudioTrackName

char *TAP_Channel_GetAudioTrackName(int trkNum, char *trkName)

- It gets the name of specified audio track.

  *trkName : the pointer of specified audio track name. It must be allocated or NULL. In case of NULL, only return value has the pointer of the audio track name.

  Return value : the pointer of specified audio track name

TAP_Channel_SetAudioTrack

Int TAP_Channel_SetAudioTrack(int trkNum)

- It changes the current audio track.

  trkNum : audio track number

  Return value : : If 0, there is no errors.

TAP_Channel_GetTotalSubtitleTrack

int TAP_Channel_GetTotalSubtitleTrack(void)

• It gets the number of subtitle track.
Return value : the number of subtitle track.

*TAP_Channel_GetSubtitleTrackName

```
char	*TAP_Channel_GetSubtitleTrackName( int trkNum, char *trkName )
```

• It gets the name of specified subtitle track.
*trkName : the pointer of specified subtitle track name. It must be allocated or NULL. In case of NULL, only return value has the pointer of the subtitle track name.
Return value : the pointer of specified subtitle track name

TAP_Channel_SetSubtitleTrack

```
int	TAP_Channel_SetSubtitleTrack( int trkNum )
```

• It change the current subtitle track.
trkNum : subtitle track number.
Return value : : If 0, there is no errors.

TAP_Channel_IsStarted

```
bool	TAP_Channel_IsStarted( int mainSub );
```

• It checks whether the service started or not.
mainSub : 1 – main screen, 0 – sub screen
return value : TRUE = Service started, FALSE = Not started

TAP_Channel_GetCurrentSubTV

```
int	TAP_Channel_GetCurrentSubTV( int *svcNum );
```

• It gets the service number of sub-screen in PIP.
Return value : 0

TAP_Channel_GetFirstInfo

```
int	TAP_Channel_GetFirstInfo( int svcType, TYPE_TapChInfo *chInfo );
```

• It gets the channel information of the first service in the current TP.
svcType : 1 – Radio service, 0 – TV service
*chInfo : Channel information
Return value : Total number of services in the current TP.

TAP_Channel_GetNextInfo

```
int	TAP_Channel_GetNextInfo( TYPE_TapChInfo *chInfo );
```

• It gets the channel information of the next service in the current TP.
*chInfo : Channel information
Return value : If 0, there is no more service. If 1, there is a service.

5.8 EPG Function

TAP_GetEvent

```
TYPE_TapEvent*          TAP_GetEvent( int chType, int chNum, int *eventNum )
```

- It gets all the event information of the specified channel.
  chType : Channel type.
  chNum : Channel number.
  *eventNum : number of event.
  Return value : pointer of array that stores events.

TAP_GetCurrentEvent

```
TYPE_TapEvent*          TAP_GetCurrentEvent( int chType, int chNum )
```

- It gets the current event information of the specified channel.
  chType : Channel type.
  chNum : Channel number.
  Return value : pointer of event information.

TAP_ControlEit

```
word    TAP_ControlEit( bool enable )
```

- It controls EIT data loading to enable or disable.
  enable: Control flag for EIT Data Loading.
  TRUE = Enable (Default)
  FALSE = DIsable
  Return value : 0 = Success, Other Value = Fail.

TAP_PutEvent

```
bool    TAP_PutEvent( byte *eitData )
```

- It sends the EIT data to firmware. The EPG information is made from this EIT data.
  eitData : EID data which is specified in “ETSI EN 300 468".
  return value : TRUE = Success, FALSE = Fail

TAP_UpdateEvent

```
bool    TAP_EPG_UpdateEvent(  int chType, int chNum, dword evtid, dword runningStatus )
```

- It changes the runningStatus of EPG event.
  chType : Type of channel. It is defined in Tap.h.
  chNum : Channel number
  evtid : event id
  runningStatus : 0 = undefined, 1= not running, 4 = running
  return value : TRUE = Success, FALSE = Fail

TAP_DeleteEvent

```
bool      TAP_EPG_DeleteEvent(  int chType, int chNum, dword evtid, dword runningStatus )
```

- It deletes the event event.

  chType : Type of channel. It is defined in Tap.h.
  chNum : Channel number
  evtid : event id
  return value : TRUE = Success, FALSE = Fail

TAP_EPG_GetExtInfo

```
byte      *TAP_EPG_GetExtInfo ( TYPE_TapEvent *tapEvtInfo )
```

- It gets the extended information of EPG.

  tapEvtInfo : The pointer of event of which extended information is to be gotten.
  return value : Extended information string
  % Notice – The string from this function should be freed by **TAP_MemFree**.

5.9 USB Function

TAP_Usb_Read

```
void    TAP_Usb_Read( byte *pBuff, word size, void (*pFunc)(word size) )
```

- It reads the data from USB port, ie. from PC.
  pBuff : destination pointer
  size : data size
  pFunc : It will be called after reading the data.

TAP_Usb_Write

```
void    TAP_Usb_Write( byte *pBuff, word size, void (*pFunc)(word size) )
```

- It writes the data to USB port, ie. to PC.
  pBuff : source pointer
  size : data size
  pFunc : It will be called after writing data.

TAP_Usb_PacketRead

```
void    TAP_Usb_PacketRead( byte *pBuff, word size, word (*pFunc)(word size) )
```

- It reads the data from USB port, ie. from PC.
  pBuff : destination pointer
  size : data size
  pFunc : It will be called after reading the data. If 0 is returned from this callback function, the read data is processed by TAP application. If not, the read data is processed in the firmware.

The structure of read data will be released by the coming specifications.

TAP_Usb_PacketWrite

```
void    TAP_Usb_PacketWrite( byte *pBuff, word size, void (*pFunc)(word size), dword cmd )
```

- It writes the data to USB port, ie. to PC.
  pBuff : source pointer
  size : data size
  pFunc : It will be called after writing data.
  cmd : packet command

The structure of written data should be followed by the specifications of Topfield.

TAP_Usb_Cancel

```
void TapUsbCancel( void );
```

- It cancels the data transfer of USB.

5.10 Timer Function

*TAP_Timer_GetTotalNum

```
int     (*TAP_Timer_GetTotalNum)( void )
```

• It gets the total number of used timers.

return value : the number of used timer

*TAP_Timer_GetInfo

```
bool    (*TAP_Timer_GetInfo)( int entryNum, TYPE_TimerInfo *info )
```

• It gets the information of used timer.

entryNum : the timer number to be got its information

info : the pointer of the TYPE_TimerInfo which gets the information.

return value : TRUE = Success, FALSE = False

The structure of TYPE_TimerInfo is as follows.

```
typedef struct
{
        byte isRec;// 1 = Recording, 0 = VCR Timer
        byte tuner; // 0 = Tuner1 , 1 = Tuner2
        byte svcType;
        byte reserved;
        word svcNum;
        byte reservationType;
        byte nameFix; // 1 = fileName is fixed
        word duration;
        dword startTime;
        char fileName[];
} TYPE_TimerInfo
```

*TAP_Timer_Add

```
int     (*TAP_Timer_Add)( TYPE_TimerInfo *info )
```

• It adds a timer.

info : information of the timer to be added.

return value : 0 : Success
1 : ERROR : Invalid Entry or No resource available.
2 : ERROR : Invalid Tuner
0xFFFFxxxx : xxxx = There is conflict with the xxxx Timer.

*TAP_Timer_Modify

```
int     (*TAP_Timer_Modify)( int entryNum, TYPE_TimerInfo *info )
```

• It changes the information of the timer which is already set.

entryNum : the number of timer whose information will be changed.

info : new information of the timer to be changed.

return value : 0 : Success
1 : ERROR : Invalid Entry or No resource available.
2 : ERROR : Invalid Tuner
0xFFFFxxxx : xxxx = There is conflict with the xxxx Timer.

*TAP_Timer_Delete

```
bool (*TAP_Timer_Delete)( int entryNum )
```

- It deletes the timer.

entryNum : the number of timer to be deleted.
return value : TRUE = Success, FALSE = False

6. Revision History

Version 1.22

Released at 7th June 2005.

< Required Firmware version >

Since 5.11.85 (June 03 2005)

< Modified text in this document >

4.3 TAP_ID(ID_USERTAP) Contact TOPFIELD if you want certified ID. => Removed.

5.1 void TAP_GetStatus (dword *state, dword *subState)

=> void TAP_GetState (dword *state, dword *subState)

< Fixed Bugs >

1. The startTime and endTime of TYPE_TapEvent were UTC based. It is changed to local time based.

< New event >

1. EVT_VIDEO_CONV : For the conversion of NTSC to/from PAL video format

< Modified or new functions >

5.1 System Function

TAP_PrintSys	: new
TAP_PutByte	: new

5.3 OSD (On Screen Display) Function

TAP_Osd_GetBaseInfo	: new
TAP_Osd_GetBox	: new
TAP_Osd_PutBox	: new
TAP_Osd_Zoom	: new

5.5 HDD Function

TAP_Hdd_GetHddID	: new
TAP_Hdd_ChangePlaybackPos	: new
TAP_Hdd_GotoBookmark	: new
TAP_Hdd_SetBookmark	: new

5.7 Channel Function

TAP_Channel_GetCurrentSubTV	: new
TAP_Channel_GetFirstInfo	: new
TAP_Channel_GetNextInfo	: new

5.7 USB Function

TapUsbCancel	: new

Version 1.21

Released at 7th April 2005.

< Required Firmware version >

Since 5.11.55 (Mar 24 2005)

< Modified struct > - Please refer to the TAP.h

1. TYPE_TapChInfo : Satellite index, Original network ID, TS ID and Logical channel number are added.

```
typedef struct
{
    char    satName[ MAX_SatName ];
    char    chName[ MAX_SvcName ];
    word    flag;
    byte    tunerNum : 2;
    byte    polar : 1;
    byte    ttxAvailable:1;
    word    freq;
    word    sr;
    word    svcId;
    word    pmtPid;
    word    pcrPid;
    word    videoPid;
    word    audioPid;
    byte    dolbyTrack;
    byte    multifeed;

    byte    satIdx;             // satellite index
    word    orgNetId;           // Original network ID
    word    tsId;               // Transport stream ID
    word    logicalChNum;       // Logical channel number
} TYPE_TapChInfo;
```

Version 1.20

Released at 18th March 2005.

< Required Firmware version >

Since 5.11.51 (Mar 16 2005)

< Fixed Bugs >

1. TAP_Timer_Add : The bug of two timers for the different services in the same TP is fixed.
2. TYPE_PlayInfo : TYPE_PlayInfo *playInfo->file->name will not changed after recording.
3. The timer information will be stored in the flash memory in some cases
4. The multifeed information will be updated.

< New sample TAP application >

1. VFD Sample : It show how to control the VFD display (TF5500PVR only) in TAP application.
2. Ext Info Sample : It shows how to get the extended information of EPG.

< Modified struct > - Please refer to the TAP.h

1. TYPE_TapChInfo

```
    byte  ttxAvailable : 1;    //  Teletext availability
  2. TYPE_TapEvent
    byte  satIdx;      // Satellite index
    word  svcId;       // Service ID
    word  orgNetId;  // Original Network ID
    word  tsId;         // Transport Stream ID
  3. TYPE_PlayInfo
    byte  repeatStatus;  // Repeat status in playback information

    // Playback Repeat Status
    typedef enum
    {
        REPEAT_None,
        REPEAT_Region,
        REPEAT_Total,
        N_RepeatMode
    } TYPE_RepeatMode;
```

< Modified or new functions >

5.1 System Function

TAP_Vfd_GetStatus	: new
TAP_Vfd_Control	: new
TAP_Vfd_SendData	: new

5.7 Channel Function

TAP_Channel_IsStarted	: new

5.8 EPG Function

TAP_EPG_GetExtInfo	: new

Version 1.11

Released at 13th December 2004.

< Required Firmware version >

After 5.11.32 (Dec 06 2004)

< Modified or new functions >

5.1 System Function

TAP_SetBaudRate	: new

5.8 EPG Function

TAP_ControlEit	: new
TAP_PutEvent	: new
TAP_EPG_UpdateEvent	: new
TAP_EPG_DeleteEvent	: new

Version 1.10

Released at 07th December 2004.

< Required Firmware version >

After 5.10.99 (June 10 2004)

< New sample TAP application >

MyOSD : It show how to change infobox and volume bar.

< New utilities >

MkGd.exe : It converts the image file of PCX and BMP format into GD format which can be used in TAP.

a. If you use GD format, you can display the image easily with TAP_Osd_PutGd() function.
b. How to use
   (1) Execute MkGd.exe and press the button of “Find”.
   (2) Open the PCX or BMP file which you want to convert.
   (3) Press the button of “Conversion”.
c. You should check “1555” in case of TF5000PVR because it uses 16 bit colour.
d. This utility is in the folder of “Utilities”.

< New Event >

4.2 EVT_SVCSTATUS : new

< Modified or new functions >

5.3 OSD (On Screen Display) Function

TAP_SysOsdControl : new

5.4 HDD Function

TAP_Hdd_StartRecord : new
TAP_Hdd_StopRecord : new
TAP_Hdd_GetRecInfo : new
TAP_Hdd_GetPlayInfo : new

Version 1.00

Released at 23th April 2004.

< modified or new functions >

4.2 TAP_EventHandler : modified

5.1 System Function

*TAP_GetSystemVar : new
*TAP_SetSystemVar : new
*TAP_WriteSystemVar : new
*TAP_GenerateEvent : new

5.2 Common Function

*TAP_CaptureScreen : new

5.3 OSD (On Screen Display) Function

TAP_Osd_Create : new

5.4 HDD Function

*TAP_Hdd_PlayTs : new
*TAP_Hdd_StopTs : new
*TAP_Hdd_PlayMp3 : new
*TAP_Hdd_StopMp3 : new
*TAP_Hdd_Rename : new

5.10 Timer Function

*TAP_Timer_GetTotalNum : new
*TAP_Timer_GetInfo : new
*TAP_Timer_Add : new
*TAP_Timer_Modify : new
*TAP_Timer_Delete : new

Version 0.21

Released at 30th May 2003.