**VATEK*****MDK API Structure***

B-series

Copyright © 2021 by VATEK

Content

[STRUCTURES 6](#_Toc95900326)

[video\_input\_parm 6](#_Toc95900327)

[audio\_input\_parm 8](#_Toc95900328)

[logo\_input\_parm 9](#_Toc95900329)

[video\_encoder\_parm 10](#_Toc95900330)

[audio\_encode\_parm 11](#_Toc95900331)

[encoder\_mux\_parm 12](#_Toc95900332)

[encoder\_quality\_parm 13](#_Toc95900333)

[tsmux\_iso13818\_parm 14](#_Toc95900334)

[tsmux\_pure\_parm 15](#_Toc95900335)

[tsmux\_default\_parm 16](#_Toc95900336)

[psi\_default\_iso13818\_channel /psi\_default\_iso13818\_program 17](#_Toc95900337)

[vatek\_string 18](#_Toc95900338)

[psi\_default\_dvb\_channel / psi\_default\_dvb\_program 19](#_Toc95900339)

[psi\_default\_arib\_channel / psi\_default\_arib\_program 20](#_Toc95900340)

[psi\_default\_abnt\_channel / psi\_default\_abnt\_program 21](#_Toc95900341)

[psi\_default\_psip\_channel / psi\_default\_psip\_program 22](#_Toc95900342)

[modulator\_base\_parm 23](#_Toc95900343)

[modulator\_dvbt\_parm 25](#_Toc95900344)

[modulator\_j83a\_parm 27](#_Toc95900345)

[modulator\_atsc\_parm 28](#_Toc95900346)

[modulator\_j83b\_parm 29](#_Toc95900347)

[modulator\_dtmb\_parm 30](#_Toc95900348)

[modulator\_isdbt\_parm 32](#_Toc95900349)

[modulator\_j83c\_parm 34](#_Toc95900350)

[modulator\_dvbt2\_parm 35](#_Toc95900351)

[modulator\_sinewave\_parm 39](#_Toc95900352)

[psitablelist\_parm 40](#_Toc95900353)

[userdata\_cc\_parm 41](#_Toc95900354)

[phy\_video\_info 42](#_Toc95900355)

[phy\_audio\_info 44](#_Toc95900356)

[r2\_board\_handle 45](#_Toc95900357)

[r2\_calibrate 46](#_Toc95900358)

[VATEK MAIN API (broadcast) 47](#_Toc95900359)

[vatek\_broadcast\_create 47](#_Toc95900360)

[vatek\_broadcast\_destroy 48](#_Toc95900361)

[vatek\_broadcast\_start 49](#_Toc95900362)

[vatek\_broadcast\_stop 50](#_Toc95900363)

[vatek\_broadcast\_chipstatus 51](#_Toc95900364)

[vatek\_broadcast\_bcstatus 52](#_Toc95900365)

[vatek\_broadcast\_encoder\_setinputparm\_phy 53](#_Toc95900366)

[vatek\_broadcast\_encoder\_setinputparm\_logo 54](#_Toc95900367)

[vatek\_broadcast\_encoder\_setencodeparm 55](#_Toc95900368)

[vatek\_broadcast\_encoder\_setmuxparm 56](#_Toc95900369)

[vatek\_broadcast\_encoder\_setqualityparm 57](#_Toc95900370)

[vatek\_broadcast\_tsmux\_setparm 58](#_Toc95900371)

[vatek\_broadcast\_modulator\_setparm 59](#_Toc95900372)

[vatek\_broadcast\_psitable\_register 60](#_Toc95900373)

[vatek\_broadcast\_psitable\_insert 61](#_Toc95900374)

[vatek\_broadcast\_userdata\_insert\_cc 62](#_Toc95900375)

[VATEK PERIPHERAL API (PHY) 63](#_Toc95900376)

[vatek\_phy\_create 63](#_Toc95900377)

[vatek\_phy\_destroy 64](#_Toc95900378)

[vatek\_phy\_enable 65](#_Toc95900379)

[vatek\_phy\_status 66](#_Toc95900380)

[vatek\_phy\_getvideoinfo 67](#_Toc95900381)

[vatek\_phy\_getaudioinfo 68](#_Toc95900382)

[vatek\_phy\_setvideoinfo 69](#_Toc95900383)

[vatek\_phy\_getccdata 70](#_Toc95900384)

[VATEK PERIPHERAL API (RF) 71](#_Toc95900385)

[vatek\_rf\_create 71](#_Toc95900386)

[vatek\_rf\_destroy 72](#_Toc95900387)

[vatek\_rf\_start 73](#_Toc95900388)

[vatek\_rf\_stop 74](#_Toc95900389)

[vatek\_rf\_getstatus 75](#_Toc95900390)

[vatek\_rf\_setcalibrate 76](#_Toc95900391)

[VATEK PORTING API (I2C) 77](#_Toc95900392)

[vatek\_porting\_i2c\_set\_speed 77](#_Toc95900393)

[vatek\_porting\_i2c\_start 78](#_Toc95900394)

[vatek\_porting\_i2c\_write 79](#_Toc95900395)

[vatek\_porting\_i2c\_read 80](#_Toc95900396)

[vatek\_porting\_i2c\_stop 81](#_Toc95900397)

[VATEK PORTING API (GPIO) 82](#_Toc95900398)

[vatek\_porting\_gpio\_write 82](#_Toc95900399)

[vatek\_porting\_gpio\_read 83](#_Toc95900400)

[VATEK PORTING API (UART) 84](#_Toc95900401)

[vatek\_porting\_uart\_receive 84](#_Toc95900402)

[vatek\_porting\_uart\_transmit 85](#_Toc95900403)

[VATEK PORTING API (SYSTEM) 86](#_Toc95900404)

[vatek\_porting\_system\_gettick 86](#_Toc95900405)

[vatek\_porting\_system\_delay 87](#_Toc95900406)

[vatek\_porting\_system\_mainreset 88](#_Toc95900407)

[vatek\_porting\_system\_phyreset 89](#_Toc95900408)

STRUCTURES

video\_input\_parm

Front end image input parameters, used to set VI (Video Interface).

|  |
| --- |
| typedef struct \_video\_input\_parm  {  video\_input\_resolution resolution;  video\_input\_aspectrate aspectrate;  uint32\_t offset\_x;  uint32\_t offset\_y;  uint8\_t buswidth\_16;  uint8\_t separated\_sync;  uint8\_t clk\_inverse;  uint8\_t hsync\_inverse;  uint8\_t vsync\_inverse;  uint8\_t field\_inverse;  uint8\_t ext\_half\_fps;  }video\_input\_parm, \*Pvideo\_input\_parm; |

**Members**

*resolution*  
Input the resolution of the video.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| vi\_resolution\_1080p60 | The resolution is 1080P，60FPS |
| vi\_resolution\_1080p59\_94 | The resolution is 1080P，59.94FPS |
| vi\_resolution\_1080p50 | The resolution is 1080P，50FPS |
| vi\_resolution\_1080p30 | The resolution is 1080P，30FPS |
| vi\_resolution\_1080p25 | The resolution is 1080P，25FPS |
| vi\_resolution\_1080p24 | The resolution is 1080P，24FPS |
| vi\_resolution\_1080p23\_97 | The resolution is 1080P，23.97FPS |
| vi\_resolution\_1080i60 | The resolution is 1080I，30FPS |
| vi\_resolution\_1080i59\_94 | The resolution is 080I，29.97FPS |
| vi\_resolution\_1080i50 | The resolution is 1080I，25FPS |
| vi\_resolution\_720p60 | The resolution is 720P，60FPS |
| vi\_resolution\_720p59\_94 | The resolution is 720P，59.94FPS |
| vi\_resolution\_720p50 | The resolution is 720P，50FPS |
| vi\_resolution\_576p50 | The resolution is 576P，50FPS |
| vi\_resolution\_480p60 | The resolution is 480P，60FPS |
| vi\_resolution\_480p59\_94 | The resolution is 480P，59.94FPS |
| vi\_resolution\_576i50 | The resolution is 576I，25FPS |
| vi\_resolution\_576p25 | The resolution is 576P，25FPS |
| vi\_resolution\_480i60 | The resolution is 480I，30FPS |
| vi\_resolution\_480i59\_94 | The resolution is 480I，29.97FPS |
| vi\_resolution\_480p30 | The resolution is 480P，30FPS |
| vi\_resolution\_480p29\_97 | The resolution is 480P，29.97FPS |

*aspectrate*

Input the aspect rate of video.

|  |  |
| --- | --- |
| parameter | meaning |
| vi\_aspectrate\_16\_9 | aspect rate is 16:9 |
| vi\_aspectrate\_4\_3 | aspect rate is 4:3 |

*offset\_ x*

Enter the x-axis displacement of the image.

*offset\_ y*

Enter the y-axis displacement of the image.

*buswidth\_ sixteen*

The data width of front-end image input (8 bits or 16 bits).

*separated\_ sync*

The synchronization status of front-end image input (embedded synchronization or separated synchronization).

*clk\_ inverse*

The front-end image clock signal is reversed.

*hsync\_ inverse*

The front-end image H synchronization signal is reversed.

*vsync\_ inverse*

Front end video V synchronization signal inversion.

*field\_ inverse*

Front end image field signal inversion.

*ext\_ half\_ fps*

FPS halved.

audio\_input\_parm

Front end sound input parameters, used to set AI (Audio Interface).

|  |
| --- |
| typedef struct \_audio\_input\_parm  {  audio\_input\_samplerate samplerate;  } audio\_input\_parm, \*Paudio\_input\_parm; |

**Members**

*samplerate*

Input the sample rate.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| ai\_samplerate\_48K | The sample rate is 48K Hz |
| ai\_samplerate\_44\_1K | The sample rate is 44.1K Hz |
| ai\_samplerate\_32K | The sample rate is 32K Hz |

logo\_input\_parm

Logoinput parameters, used to set bootlogo and colorbar.

|  |
| --- |
| typedef struct \_logo\_input\_parm  {  logo\_type type;  uint32\_t logoidx;  } logo\_input\_parm, \*Plogo\_input\_parm; |

**Members**

*type*

Set logo type.

|  |  |
| --- | --- |
| parameter | meaning |
| logo\_type\_colorbar | To set the logo as colorbar |
| logo\_type\_bootlogo | To set customed logo |

*logoidx*

Customedlogo index.

video\_encoder\_parm

Parameters of video encode.

|  |
| --- |
| typedef struct \_video\_encoder\_parm  {  video\_encode\_type type;  uint8\_t interlaced\_frame;  uint8\_t progressive\_2\_i;  } video\_encoder\_parm, \*Pvideo\_encoder\_parm; |

**Members**

*type*

The category of video encode.

|  |  |
| --- | --- |
| parameter | Meaning |
| ve\_type\_mpeg2 | mpeg2 video encode |
| ve\_type\_h264 | h264 video encode |

*interlaced\_frame*

interlaced encode.

*progressive\_2\_i*

progressive to interlaced.

audio\_encode\_parm

Parameters of audio encode.

|  |
| --- |
| typedef struct \_audio\_encode\_parm  {  audio\_encode\_type type;  audio\_encode\_channel channel;  } audio\_encode\_parm, \*Paudio\_encode\_parm; |

**Members**

*type*

The category of audio encode.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| ae\_type\_mp1\_l2 | mpeg1-l2 audio encode |
| ae\_type\_aac\_lc\_adts | aac-lc-adts audio encode |
| ae\_type\_ac\_3 | ac3 audio encode |
| ae\_type\_aac\_lc\_latm | aac-lc-latm audio encode |

*channel*

The category of audio channel.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| ae\_channel\_mute | Turn off audio channel |
| ae\_channel\_stereo | Stereo channel |
| ae\_channel\_mono\_l | Left audio track |
| ae\_channel\_mono\_r | Right audio track |
| ae\_channel\_stereo\_mono\_l | Dual channel, ch1 is the left channel |
| ae\_channel\_stereo\_mono\_r | Dual channel, ch1 is the right channel |

encoder\_mux\_parm

Parameters of encoder package.

|  |
| --- |
| typedef struct \_encoder\_mux\_parm  {  uint32\_t video\_pid;  uint32\_t audio\_pid;  }encoder\_mux\_parm, \*Pencoder\_mux\_parm; |

**Members**

*video\_pid*

Image stream PID. The range is 0 ~ 0x1FFF.

*audio\_pid*

Sound stream PID. The range is 0 ~ 0x1FFF.

encoder\_quality\_parm

Parameter of encode quality.

|  |
| --- |
| typedef struct \_encoder\_quality\_parm  {  quality\_rcmode rcmode;  uint32\_t minq;  uint32\_t maxq;  uint32\_t gop;  uint32\_t latency;  uint32\_t bitrate;  }encoder\_quality\_parm, \*Pencoder\_quality\_parm; |

**Members**

*rcmode*

The mode of encode quality.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| q\_rcmode\_vbr | VBR mode, adjust coding Q value according to the dynamic code rate. |
| q\_rcmode\_fixedq | Fixed coding Q value mode. |
| q\_rcmode\_auto | Automatic mode, adjust coding Q value according to the dynamic code rate. |

*minq*

The minimum value of Q value range. The range is 0 ~ 32 and the default is 3.

*maxq*

The maximum value of Q value range. The range is 0 ~ 32 and the default is 10.

*gop*

Group of pictures. The default is 16.

*latency*

Latency of video and audio in milliseconds. The default is 500 milliseconds.

*bitrate*

Upper limit of video and audio stream rate. The default is 19m.

tsmux\_iso13818\_parm

Transport stream parameters of iso13818.

|  |
| --- |
| typedef struct \_tsmux\_iso13818\_parm  {  uint32\_t pcr\_pid;  uint32\_t padding\_pid;  uint32\_t pmtpid;  uint32\_t tsid;  uint32\_t program\_num;  }tsmux\_iso13818\_parm, \*Ptsmux\_iso13818\_parm; |

**Members**

*pcr\_pid*

PCR PID. The range is 0 ~ 0x1fff.

*padding\_pid*

Padding PID. The range is 0 ~ 0x1fff..

*pmtpid*

PID of PMT. The range is 0 ~ 0x1fff.

*tsid*

Transmission stream identification. The range is 0 ~ 0xFFFF.

*program\_num*

Program channel number. The range is 0 ~ 0xFFFF.

tsmux\_pure\_parm

Parameters of TSMux pure.

|  |
| --- |
| typedef struct \_tsmux\_pure\_parm  {  uint32\_t pcr\_pid;  uint32\_t padding\_pid;  }tsmux\_pure\_parm, \*Ptsmux\_pure\_parm; |

**Members**

*pcr\_pid*

PCR PID. The range is 0 ~ 0x1fff.

*padding\_pid*

Padding PID. The range is 0 ~ 0x1fff.

tsmux\_default\_parm

Parameter TS parameter.

|  |
| --- |
| typedef struct \_tsmux\_default\_parm  {  uint32\_t pcr\_pid;  uint32\_t padding\_pid;  }tsmux\_default\_parm, \*Ptsmux\_default\_parm; |

**Members**

*pcr\_pid*

PCR serial flow PID. The range is 0 ~ 0x1FFF.

*padding\_pid*

Padding PID. The range is 0~0x1FFF.

psi\_default\_iso13818\_channel /psi\_default\_iso13818\_program

When setting the PSI table of iso13818 in default mode, you need to set channel and program through this parameter.

|  |
| --- |
| typedef struct \_psi\_default\_iso\_channel  {  uint16\_t transport\_stream\_id;  }psi\_default\_iso13818\_channel, \*Ppsi\_default\_iso13818\_channel;  typedef struct \_psi\_default\_iso\_program  {  uint16\_t program\_number;  }psi\_default\_iso13818\_program, \*Ppsi\_default\_iso13818\_program; |

**Members**

*transport\_stream\_id*

Set TS ID.

*program\_number*

Set program number.

vatek\_string

You need to use this parameter to set information about the table in default mode.

|  |
| --- |
| typedef struct \_vatek\_string  {  uint32\_t len;  uint8\_t\* text;  }vatek\_string, \*Pvatek\_string; |

**Members**

*len*

Set string length.

*text*

Set string.

psi\_default\_dvb\_channel / psi\_default\_dvb\_program

When setting DVB in default mode, you need to set channel and program through this parameter.

|  |
| --- |
| typedef struct \_psi\_default\_dvb\_channel  {  uint16\_t transport\_stream\_id;  uint16\_t network\_id;  Pvatek\_string network\_name;  }psi\_default\_dvb\_channel, \*Ppsi\_default\_dvb\_channel;  typedef struct \_psi\_default\_dvb\_program  {  uint16\_t original\_network\_id;  uint16\_t program\_number;  uint16\_t channel\_no;  Pvatek\_string service\_name;  }psi\_default\_dvb\_program, \*Ppsi\_default\_dvb\_program; |

**Members**

*transport\_stream\_id*

*Network\_id*

*Network\_name*

*original\_network\_id*

*program\_number*

channel\_no

*Service\_name*

psi\_default\_arib\_channel / psi\_default\_arib\_program

When setting ARIB in default mode, you need to set channel and program through this parameter.

|  |
| --- |
| typedef struct \_psi\_default\_arib\_channel  {  uint8\_t region\_id;  uint8\_t broadcaster\_id;  uint8\_t remote\_control\_key\_id;  Pvatek\_string network\_name;  }psi\_default\_arib\_channel, \*Ppsi\_default\_arib\_channel;  typedef struct \_psi\_default\_arib\_program  {  uint8\_t service\_no;  mux\_iso\_lang main\_lang;  mux\_iso\_lang sub\_lang;  arib\_abnt\_copy copy\_flag;  Pvatek\_string service\_name;  Pvatek\_string ts\_name;  }psi\_default\_arib\_program, \*Ppsi\_default\_arib\_program; |

**Members**

*Region\_id*

*Broadcaster\_id*

*Remote\_control\_key\_id*

*Network\_name*

*Service\_no*

*Main\_lang*

*Sub\_lang*

*Copy\_flag*

*Service\_name*

*Ts\_name*

|  |  |
| --- | --- |
| parameter | index |
| arib\_abnt\_free | 0 |
| arib\_abnt\_copyonce | 1 |
| arib\_abnt\_forbidden | 2 |

psi\_default\_abnt\_channel / psi\_default\_abnt\_program

When setting ABNT in default mode, you need to set channel and program through this parameter.

|  |
| --- |
| typedef struct \_psi\_default\_abnt\_channel  {  uint8\_t remote\_control\_key\_id;  uint8\_t original\_network\_id;  uint16\_t area\_code;  Pvatek\_string network\_name;  }psi\_default\_abnt\_channel, \*Ppsi\_default\_abnt\_channel;  typedef struct \_psi\_default\_abnt\_program  {  uint8\_t service\_no;  mux\_iso\_lang main\_lang;  mux\_iso\_lang sub\_lang;  arib\_abnt\_copy copy\_flag;  Pvatek\_string service\_name;  Pvatek\_string ts\_name;  }psi\_default\_abnt\_program, \*Ppsi\_default\_abnt\_program; |

**Members**

*Remote\_control\_key\_id*

*Original\_network\_id*

*Area\_code*

*Network\_name*

*Service\_no*

*Main\_lang*

*Sub\_lang*

*Copy\_flag*

|  |  |
| --- | --- |
| **parameter** | **index** |
| arib\_abnt\_free | 0 |
| arib\_abnt\_copyonce | 1 |
| arib\_abnt\_forbidden | 2 |

*Service\_name*

*TS\_name*

psi\_default\_psip\_channel / psi\_default\_psip\_program

To set PSIP in default mode, you need to set channel and program through this parameter.

|  |
| --- |
| typedef struct \_psi\_default\_psip\_channel  {  uint32\_t psip\_flags;  uint16\_t transport\_stream\_id;  uint8\_t gps\_utc\_offset;  uint16\_t daylight\_saving;  psip\_cc\_mode cc\_mode;  Pvatek\_string short\_name;  }psi\_default\_psip\_channel, \*Ppsi\_default\_psip\_channel;  typedef struct \_psi\_default\_psip\_program  {  uint16\_t program\_number;  uint16\_t source\_id;  uint16\_t channel\_major;  uint16\_t channel\_minor;  Pvatek\_string long\_name;  }psi\_default\_psip\_program, \*Ppsi\_default\_psip\_program; |

**Members**

*Psip\_flags*

*Transport\_stream\_id*

*Gps\_utc\_offset*

*Daylight\_saving*

*cc\_mode*

|  |  |
| --- | --- |
| parameter | index |
| Cc\_mode\_disable | 0 |
| Cc\_mode\_608 | 1 |

*Short\_name*

*Program\_number*

*Source\_id*

*Channel\_major*

*Channel\_minor*

*Long\_name*

modulator\_base\_parm

Basic parameters of Digital modulation.

|  |
| --- |
| typedef struct \_modulator\_base\_parm  {  modulator\_type type;  modulator\_ifmode ifmode;  uint32\_t iffreq;  uint32\_t dacgain;  uint32\_t bw\_sb;  }modulator\_base\_parm, \*Pmodulator\_base\_parm; |

**Members**

*type*

Modulations.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| m\_type\_dvb\_t | dvb-t modulation |
| m\_type\_j83a | j83a modulation |
| m\_type\_atsc | atsc modulation |
| m\_type\_j83b | j83b modulation |
| m\_type\_dtmb | dtmb modulation |
| m\_type\_isdb\_t | isdb-t modulation |
| m\_type\_j83c | j83c modulation |
| m\_type\_dvb\_t2 | dvb-t2 modulation |
| m\_type\_sinewave | signal tone |

*ifmode*

Modulation mode, preset IF frequency mode.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| m\_ifmode\_disable | Disable |
| m\_ifmode\_if\_freq | IF mode |
| m\_ifmode\_iq\_offset | IQ offset mode |
| m\_ifmode\_if\_invfreq | IF inversion mode |
| m\_ifmode\_iq\_invfreq | IQ shift + IF inversion mode |

*iffreq*

IF frequency.

*dacgain*

Digital to analog converter amplitude.

*bw\_sb*

Frequency bandwidth.

modulator\_dvbt\_parm

Parameters of DVB-T modulation.

|  |
| --- |
| typedef struct \_modulator\_dvbt\_parm  {  dvbt\_constellation constellation;  dvbt\_fft fft;  dvbt\_guardinterval guardinterval;  dvbt\_coderate coderate;  }modulator\_dvbt\_parm, \*Pmodulator\_dvbt\_parm; |

**Members**

*constellation*

Set constellation.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt\_constellation\_qpsk | 4-bit quadrature phase shift modulation |
| dvbt\_constellation\_qam16 | 16-bit quadrature amplitude modulation |
| dvbt\_constellation\_qam64 | 64-bit quadrature amplitude modulation |

*fft*

Set **F**ast **F**ourier **T**ransform.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt\_fft\_2k | The sampling bandwidth is 2 kHz |
| dvbt\_fft\_4k | The sampling bandwidth is 4 kHz |
| dvbt\_fft\_8k | The sampling bandwidth is 8 kHz |

*guardinterval*

Cyclic prefix and guard interval.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt\_guardinterval\_1\_32 | The interval is 1 / 32 of the symbol length. |
| dvbt\_guardinterval\_1\_16 | The interval is 1 / 16 of the symbol length. |
| dvbt\_guardinterval\_1\_8 | The interval is 1 / 8 of the symbol length. |
| dvbt\_guardinterval\_1\_4 | The interval is 1 / 4 of the symbol length. |

*coderate*

Code rate.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt\_coderate\_1\_2 | There is 1 redundant bit for every 2 bits |
| dvbt\_coderate\_2\_3 | There is 1 redundant bit for every 3 bits |
| dvbt\_coderate\_3\_4 | There is 1 redundant bit for every 4 bits |
| dvbt\_coderate\_5\_6 | There is 1 redundant bit for every 6 bits |
| dvbt\_coderate\_7\_8 | There is 1 redundant bit for every 8 bits |

modulator\_j83a\_parm

Parameters of j83a modulation.

|  |
| --- |
| typedef struct \_modulator\_j83a\_parm  {  j83a\_constellation constellation;  }modulator\_j83a\_parm, \*Pmodulator\_j83a\_parm; |

**Members**

*constellation*

Set constellation.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| j83a\_constellation\_qam16 | 16-bit quadrature amplitude modulation |
| j83a\_constellation\_qam32 | 32-bit quadrature amplitude modulation |
| j83a\_constellation\_qam64 | 64-bit quadrature amplitude modulation |
| j83a\_constellation\_qam128 | 128bit quadrature amplitude modulation |
| j83a\_constellation\_qam256 | 256-bit quadrature amplitude modulation |

modulator\_atsc\_parm

Parameters of ATSC modulation.

|  |
| --- |
| typedef struct \_modulator\_atsc\_parm  {  atsc\_constellation constellation;  }modulator\_atsc\_parm, \*Pmodulator\_atsc\_parm; |

**Members**

*constellation*

Set constellation.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| atsc\_constellation\_8vsb | 8-level residual sideband modulation |

modulator\_j83b\_parm

Parameters of j83b modulation.

|  |
| --- |
| typedef struct \_modulator\_j83b\_parm  {  j83b\_constellation constellation;  }modulator\_j83b\_parm, \*Pmodulator\_j83b\_parm; |

**Members**

*constellation*

Set constellation.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| j83b\_constellation\_qam64 | 64-bit quadrature amplitude modulation |
| j83b\_constellation\_qam256 | 256-bit quadrature amplitude modulation |

modulator\_dtmb\_parm

Parameters of DTMB modulation.

|  |
| --- |
| typedef struct \_modulator\_dtmb\_parm  {  dtmb\_constellation constellation;  dtmb\_time\_interleaved ti;  dtmb\_coderate coderate;  dtmb\_carriermode carriermode;  dtmb\_syncframe syncframe;  }modulator\_dtmb\_parm, \*Pmodulator\_dtmb\_parm; |

**Members**

*constellation*

Set constellation.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dtmb\_constellation\_qpsk | 4-bit quadrature phase shift modulation |
| dtmb\_constellation\_qam4\_nr | 4-bit quadrature amplitude modulation\_NR |
| dtmb\_constellation\_qam16 | 16-bit quadrature amplitude modulation |
| dtmb\_constellation\_qam32 | 32-bit quadrature amplitude modulation |
| dtmb\_constellation\_qam64 | 64-bit quadrature amplitude modulation |

*ti*

Time interleaves length setting.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dtmb\_time\_interleaved\_disable | Disablw |
| dtmb\_time\_interleaved\_240 | 240 milliseconds |
| dtmb\_time\_interleaved\_720 | 720 milliseconds |

*coderate*

Code rate.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dtmb\_coderate\_0\_4 | There are 3 redundant bits for every 5 bits |
| dtmb\_coderate\_0\_6 | There are 2 redundant bits for every 5 bits |
| dtmb\_coderate\_0\_8 | There are 1 redundant bits for every 5 bits |

*carriermode*

Carrier mode.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dtmb\_carriermode\_3780 | 3780 mode |
| dtmb\_carriermode\_1 | Single mode |

*syncframe*

Sync frame length. (Unit: symbol)

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dtmb\_syncframe\_420 | The length is 420 symbols |
| dtmb\_syncframe\_595 | The length is 595 symbols |
| dtmb\_syncframe\_945 | The length is 945 symbols |

modulator\_isdbt\_parm

Parameters of ISDBT modulation.

|  |
| --- |
| typedef struct \_modulator\_isdbt\_parm  {  isdbt\_constellation constellation;  isdbt\_fft fft;  isdbt\_guardinterval guardinterval;  isdbt\_coderate coderate;  isdbt\_time\_interleaved ti;  uint8\_t ac1\_en;  uint8\_t ac2\_en;  uint8\_t emergency\_en;  }modulator\_isdbt\_parm, \*Pmodulator\_isdbt\_parm; |

**Members**

*constellation*

Set constellation.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| isdbt\_constellation\_dqpsk | 4-bit relative phase shift modulation |
| isdbt\_constellation\_qpsk | 4-bit quadrature phase shift modulation |
| isdbt\_constellation\_qam16 | 16-bit quadrature amplitude modulation |
| isdbt\_constellation\_qam64 | 64-bit quadrature amplitude modulation |

*fft*

Set **F**ast **F**ourier **T**ransform.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| isdbt\_fft\_2k | The sampling bandwidth is 2 KHz |
| isdbt\_fft\_4k | The sampling bandwidth is 4 KHz |
| isdbt\_fft\_8k | The sampling bandwidth is 8 KHz |

*guardinterval*

Cyclic prefix and guard interval.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| isdbt\_guardinterval\_1\_32 | The interval is 1 / 32 of the symbol length. |
| isdbt\_guardinterval\_1\_16 | The interval is 1 / 16 of the symbol length. |
| isdbt\_guardinterval\_1\_8 | The interval is 1 / 8 of the symbol length. |
| isdbt\_guardinterval\_1\_4 | The interval is 1 / 4 of the symbol length. |

*coderate*

Code rate.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| isdbt\_coderate\_1\_2 | There is 1 redundant bit for every 2 bits |
| isdbt\_coderate\_2\_3 | There is 1 redundant bit for every 3 bits |
| isdbt\_coderate\_3\_4 | There is 1 redundant bit for every 4 bits |
| isdbt\_coderate\_5\_6 | There is 1 redundant bit for every 6 bits |
| isdbt\_coderate\_7\_8 | There is 1 redundant bit for every 8 bits |

*ti*

Time interleaves mode setting.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| isdbt\_time\_interleaved\_disable | Disable |
| isdbt\_time\_interleaved\_mode1 | Mode 1 |
| isdbt\_time\_interleaved\_mode2 | Mode 2 |
| isdbt\_time\_interleaved\_mode3 | Mode 3 |

*ac1\_en*

AC1 encode.

*ac2\_en*

AC2 encode.

*emergency\_en*

Emergency encode.

modulator\_j83c\_parm

Parameters of j83c modulation.

|  |
| --- |
| typedef struct \_modulator\_j83c\_parm  {  j83c\_constellation constellation;  }modulator\_j83c\_parm, \*Pmodulator\_j83c\_parm; |

**Members**

*constellation*

Set constellation.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| j83c\_constellation\_qam64 | 64-bit quadrature amplitude modulation |
| j83c\_constellation\_qam256 | 256-bit quadrature amplitude modulation |

modulator\_dvbt2\_parm

Parameters of DVB-T2 modulation.

|  |
| --- |
| typedef struct \_modulator\_dvbt2\_parm  {  dvbt2\_version version;  dvbt2\_issy issy;  dvbt2\_nti nti;  dvbt2\_l1\_constellation l1\_constellation;  dvbt2\_plp\_constellation plp\_constellation;  dvbt2\_fft fft;  dvbt2\_coderate coderate;  dvbt2\_guardinterval guardinterval;  dvbt2\_pilotpattern pilotpattern;  dvbt2\_ffc ffc;  uint32\_t nid;  uint32\_t sid;  uint32\_t fecbn;  uint32\_t sbn;  uint8\_t extend\_carrier\_mode\_en;  uint8\_t constellation\_rotation\_en;  uint8\_t input\_ts\_hem\_en;  uint8\_t delete\_null\_packet\_en;  uint8\_t vbr\_coding\_en;  uint8\_t time\_interval\_en;  }modulator\_dvbt2\_parm, \*Pmodulator\_dvbt2\_parm; |

**Members**

*version*

dvbt2 modulation version.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt2\_version\_1\_1\_1 | Version 1.1.1 |
| dvbt2\_version\_1\_1\_1\_lite | Version 1.1.1-Lite profile |
| dvbt2\_version\_1\_2\_1 | Version 1.2.1 |
| dvbt2\_version\_1\_2\_1\_lite | Version 1.2.1-Lite profile |
| dvbt2\_version\_1\_3\_1 | Version 1.3.1 |
| dvbt2\_version\_1\_3\_1\_lite | Version 1.3.1-Lite profile |

*issy*

**I**nput **S**tream **Sy**nchronizer.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt2\_issy\_disable | Disable |
| dvbt2\_issy\_short | Short |
| dvbt2\_issy\_long | Long |

*nti*

The length of time.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt2\_nti\_disable | Disable |
| dvbt2\_nti\_1 | 1 |
| dvbt2\_nti\_2 | 2 |
| dvbt2\_nti\_3 | 3 |

*l1\_constellation*

First stage constellation setting.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt2\_l1\_constellation\_bpsk | 2-bit quadrature phase shift modulation |
| dvbt2\_l1\_constellation\_qpsk | 4-bit quadrature phase shift modulation |
| dvbt2\_l1\_constellation\_qam16 | 16 bit quadrature amplitude modulation |
| dvbt2\_l1\_constellation\_qam64 | 64 bit quadrature amplitude modulation |

*plp\_constellation*

Set **P**hysical **L**ayer **P**ipe.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt2\_plp\_constellation\_qpsk | 4-bit quadrature phase shift modulation |
| dvbt2\_plp\_constellation\_qam16 | 16 bit quadrature amplitude modulation |
| dvbt2\_plp\_constellation\_qam64 | 64 bit quadrature amplitude modulation |
| dvbt2\_plp\_constellation\_qam256 | 256 bit quadrature amplitude modulation |

*fft*

Set **F**ast **F**ourier **T**ransform.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt2\_fft\_1k | The sampling bandwidth is 1 kHz |
| dvbt2\_fft\_2k | The sampling bandwidth is 2 kHz |
| dvbt2\_fft\_4k | The sampling bandwidth is 4 kHz |
| dvbt2\_fft\_8k | The sampling bandwidth is 8 kHz |
| dvbt2\_fft\_16k | The sampling bandwidth is 16 kHz |
| dvbt2\_fft\_32k | The sampling bandwidth is 32 kHz |

*coderate*

Code rate.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt2\_coderate\_1\_3 | There are 2 redundant bits for every 3 bits |
| dvbt2\_coderate\_2\_5 | There are 3 redundant bits for every 5 bits |
| dvbt2\_coderate\_1\_2 | There is 1 redundant bit for every 2 bits |
| dvbt2\_coderate\_3\_5 | There are 2 redundant bits for every 5 bits |
| dvbt2\_coderate\_2\_3 | There is 1 redundant bit for every 3 bits |
| dvbt2\_coderate\_3\_4 | There is 1 redundant bit for every 4 bits |
| dvbt2\_coderate\_4\_5 | There is 1 redundant bit for every 5 bits |
| dvbt2\_coderate\_5\_6 | There is 1 redundant bit for every 6 bits |

*guardinterval*

Cyclic prefix and guard interval.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt2\_guardinterval\_1\_128 | The interval is 1 / 128 of the symbol length |
| dvbt2\_guardinterval\_1\_32 | The interval is 1 / 32 of the symbol length |
| dvbt2\_guardinterval\_1\_16 | The interval is 1 / 16 of the symbol length |
| dvbt2\_guardinterval\_19\_256 | The interval is 19 / 256 of the symbol length |
| dvbt2\_guardinterval\_19\_128 | The interval is 19 / 128 of the symbol length |
| dvbt2\_guardinterval\_1\_8 | The interval is one eighth of the symbol length |
| dvbt2\_guardinterval\_1\_4 | The interval is 1 / 4 of the symbol length |

*pilotpattern*

Pilot pattern.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt2\_pilotpattern\_1 | Model 1 |
| dvbt2\_pilotpattern\_2 | Model 2 |
| dvbt2\_pilotpattern\_3 | Model 3 |
| dvbt2\_pilotpattern\_4 | Model 4 |
| dvbt2\_pilotpattern\_5 | Model 5 |
| dvbt2\_pilotpattern\_6 | Model 6 |
| dvbt2\_pilotpattern\_7 | Model 7 |
| dvbt2\_pilotpattern\_8 | Model 8 |

*fec*

**F**orward **E**rror **C**orrection.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dvbt2\_fec\_16200 | 16200 bit |
| dvbt2\_fec\_64800 | 64800 bit |

*nid*

**N**etwork **ID**entifier

*sid*

**S**ervice **ID**entifier

*fecbn*

**F**orward **E**rror **C**orrection **B**lock **N**umbers

*sbn*

**S**ymbol **B**lock **N**umbers

*extend\_carrier\_mode\_en*

Extended carrier mode.

*constellation\_rotation\_en*

The constellation rotation.

*input\_ts\_hem\_en*

High efficiency mode of Transport Stream.

*delete\_null\_packet\_en*

Delete null packet.

*vbr\_coding\_en*

**V**ariable **B**it **R**ate.

*time\_interval\_en*

Time interval.

modulator\_sinewave\_parm

Parameter of signal tone.

|  |
| --- |
| typedef struct \_modulator\_sinewave\_parm  {  uint32\_t freq;  }modulator\_sinewave\_parm, \*Pmodulator\_sinewave\_parm; |

**Members**

*freq*

Frequency. The unit is kHz.

psitablelist\_parm

Parameter of PSI table list.

|  |
| --- |
| typedef struct \_psitable\_parm  {  uint16\_t interval\_ms;  uint16\_t tspacket\_num;  uint8\_t \*tspackets;  }psitable\_parm, \*Ppsitable\_parm;  typedef struct \_psitablelist\_parm  {  uint32\_t table\_num;  psitable\_parm table[MAX\_PSITABLE\_NUM];  uint8\_t psip\_en;  }psitablelist\_parm, \*Ppsitablelist\_parm; |

**Members**

*table\_num*

The maximum number of PSI table lists is 16 (MAX\_PSITABLE\_NUM)。

*table*

PSI table.

*interval\_ms*

The interval of repeated transmission of PSI table, in milliseconds.

*tspacket\_num*

The number of TS packets in the PSI table.

*tspackets*

The data address of the PSI form.

*psip\_en*

The special **P**rogram and **S**ystem **I**nformation **P**rotocol of ATSC.

The PID(0x1FFB)is specially processed to prevent the occurrence of **C**ontinues **C**ount **E**rror.

userdata\_cc\_parm

Closed caption packet parameters.

|  |
| --- |
| typedef struct  {  uint8\_t attr;  uint8\_t ccdata1;  uint8\_t ccdata2;  }userdata\_cc\_parm, \*Puserdata\_cc\_parm; |

**Members**

*attr*

Closed caption properties.

*ccdata1*

Closed caption data byte 1.

*ccdata2*

Closed caption data byte 2.

phy\_video\_info

Parameter of front end image information.

|  |
| --- |
| typedef struct \_phy\_video\_info  {  video\_input\_resolution resolution;  video\_input\_aspectrate aspectrate;  }phy\_video\_info, \*Pphy\_video\_info; |

**Members**

*resolution*

Enter the resolution of the image.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| vi\_resolution\_1080p60 | The resolution is 1080P，60FPS. |
| vi\_resolution\_1080p59\_94 | The resolution is 1080P，59.94FPS |
| vi\_resolution\_1080p50 | The resolution is 1080P，50FPS |
| vi\_resolution\_1080p30 | The resolution is 1080P，30FPS |
| vi\_resolution\_1080p25 | The resolution is 1080P，25FPS |
| vi\_resolution\_1080p24 | The resolution is 1080P，24FPS |
| vi\_resolution\_1080p23\_97 | The resolution is 1080P，23.97FPS |
| vi\_resolution\_1080i60 | The resolution is 1080I，30FPS |
| vi\_resolution\_1080i59\_94 | The resolution is 1080I，29.97FPS |
| vi\_resolution\_1080i50 | The resolution is 1080I，25FPS |
| vi\_resolution\_720p60 | The resolution is 720P，60FPS |
| vi\_resolution\_720p59\_94 | The resolution is 720P，59.94FPS |
| vi\_resolution\_720p50 | The resolution is 720P，50FPS |
| vi\_resolution\_576p50 | The resolution is 576P，50FPS |
| vi\_resolution\_480p60 | The resolution is 480P，60FPS |
| vi\_resolution\_480p59\_94 | The resolution is 480P，59.94FPS |
| vi\_resolution\_576i50 | The resolution is 576I，25FPS |
| vi\_resolution\_576p25 | The resolution is 576P，25FPS |
| vi\_resolution\_480i60 | The resolution is 480I，30FPS |
| vi\_resolution\_480i59\_94 | The resolution is 480I，29.97FPS |
| vi\_resolution\_480p30 | The resolution is 480P，30FPS |
| vi\_resolution\_480p29\_97 | The resolution is 480P，29.97FPS |

*aspectrate*

Enter the aspect rate of the image.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| vi\_aspectrate\_16\_9 | aspect rate is 16:9. |
| vi\_aspectrate\_4\_3 | aspect rate is 4:3. |

phy\_audio\_info

Front end sound information parameters.

|  |
| --- |
| typedef struct \_phy\_audio\_info  {  audio\_input\_samplerate samplerate;  }phy\_audio\_info, \*Pphy\_audio\_info; |

**Members**

*samplerate*

The sample rate of the input sound.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| ai\_samplerate\_48K | The sampling rate is 48K Hz |
| ai\_samplerate\_44\_1K | The sampling rate is 44.1K Hz |
| ai\_samplerate\_32K | The sampling rate is 32K Hz |

r2\_board\_handle

The back-end RF system R2 special handler.

|  |
| --- |
| typedef struct \_gpio\_pin  {  void \* port;  uint16\_t index;  }gpio\_pin, \*Pgpio\_pin;  typedef struct \_r2\_board\_handle  {  gpio\_pin r2\_cs;  gpio\_pin r2\_clk;  gpio\_pin r2\_mosi;  gpio\_pin r2\_miso;  }r2\_board\_handle, \*Pr2\_board\_handle; |

**Members**

*r2\_cs*

GPIO pin, simulate SPI CS (SS) pin.

*r2\_clk*

GPIO pin, simulate SPI SCK (SCLK) pin.

*r2\_mosi*

GPIO pin, analog SPI SDI (MOSI) pin.

*r2\_miso*

GPIO pin, analog SPI SDO (MISO) pin.

*port*

The handler of GPIO port.

*index*

GPIO pin.

r2\_calibrate

Backend RF system R2 and R2\_via\_Calibration parameters of VATek.

When using the RF calibration tool, it is necessary to set the RF chip through this parameter structure.

|  |
| --- |
| typedef struct \_r2\_calibrate  {  uint32\_t i\_offset;  uint32\_t q\_offset;  uint32\_t image\_offset;  uint32\_t phase\_offset;  uint32\_t gpio;  uint32\_t gain;  }r2\_calibrate, \*Pr2\_calibrate; |

**Members**

*i\_offset*

I direction offset parameter.

*q\_offset*

Q direction offset parameter.

*image\_offset*

Image offset parameter.

*phase\_offset*

Phase offset parameter.

*gpio*

This value is determined by the board design.

*gain*

Amplifier setting, which is determined by the board design.

VATEK MAIN API (broadcast)

vatek\_broadcast\_create

Establish a television broadcasting system.

|  |
| --- |
| vatek\_result vatek\_broadcast\_create (  [In] Pboard\_handle hboard,  [Out] Phbroadcast \*handle  ); |

**Function Arguments**

*hboard*

The Handler of user environment I2C.

The Handler will be passed in while SDK calls vatek\_porting.

*handle*

The Handler of TV broadcasting system.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter |
| vatek\_result\_memfail | Malloc failed |
| vatek\_result\_timeout | Reset VATek chip timeout (5 seconds) |
| vatek\_result\_success | The television broadcasting system was successfully established and sent back to the Handler. |

vatek\_broadcast\_destroy

Remove the television broadcasting system.

|  |
| --- |
| vatek\_result vatek\_broadcast\_destroy (  [In] Phbroadcast handle  ); |

**Function Arguments**

*handle*

The handler of TV broadcasting system.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_success | TV broadcast system removed successfully. |

vatek\_broadcast\_start

Turn on the TV broadcasting system.

|  |
| --- |
| vatek\_result vatek\_broadcast\_start (  [In] Phbroadcast handle  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_timeout | Broadcast start timeout (5 seconds). |
| vatek\_result\_hwfail | Broadcast start failed. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Successful start of TV broadcasting system. |

vatek\_broadcast\_stop

Stop the TV broadcasting system and reset the VATek chip.

|  |
| --- |
| vatek\_result vatek\_broadcast\_stop (  [In] Phbroadcast handle  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_timeout | Reset VATlek chip timeout (5 seconds). |
| vatek\_result\_hwfail | Broadcast stop failed. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Successfully stop the TV broadcasting system and reset the VATek chip. |

vatek\_broadcast\_chipstatus

Read the status of VATek chip.

|  |
| --- |
| vatek\_result vatek\_broadcast\_chipstatus (  [In] Phbroadcast handle,  [Out] chip\_status \*status  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*status*

Status of VATek Chip.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| chip\_status\_unknown | Other unknown status. |
| chip\_status\_idle | Status in transition. |
| chip\_status\_wait\_command | The chip is ready, waiting for instructions. |
| chip\_status\_running | TV broadcast in progress. |
| chip\_status\_fail | Broadcast failure status. |
| chip\_status\_badfw | Unknown broadcast failure status. |

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Successfully read the status of VATek chip. |

vatek\_broadcast\_bcstatus

Read the state of VATek chip.

|  |
| --- |
| vatek\_result vatek\_broadcast\_bcstatus (  [In] Phbroadcast handle,  [Out] broadcast\_status \*status  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*status*

Status of TV broadcast.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| bc\_status\_unknown | Other unknown status. |
| bc\_status\_idle | Waiting for broadcast status. |
| bc\_status\_wait\_source | Waiting for the front source to prepare. |
| bc\_status\_broadcast | Broadcast execution status. |
| bc\_status\_fail\_unknown | Unknown fail. |
| bc\_status\_fail\_source | Front end source fail. |
| bc\_status\_fail\_timeout | Broadcast timeout fail. |
| bc\_status\_fail\_codecdrop | Encoder drop fail. |
| bc\_status\_fail\_buffer | Buffer fail. |
| bc\_status\_fail\_muxer | Muxer fail. |
| bc\_status\_fail\_encode | Encode fail. |
| bc\_status\_fail\_media | Multimedia error. |

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Successfully read broadcast status. |

vatek\_broadcast\_encoder\_setinputparm\_phy

Set the front-end input parameters of the encoder.

|  |
| --- |
| vatek\_result vatek\_broadcast\_encoder\_setinputparm\_phy (  [In] Phbroadcast handle,  [In] video\_input\_parm vparm,  [In] audio\_input\_parm aparm  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*vparm*

Please refer to [video\_input\_parm](#video_input_parm).

*aparm*

Please refer to [audio\_input\_parm](#audio_input_parm).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | The input parameter is over range. |
| vatek\_result\_busy | The broadcast system is busy. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Parameters set successfully. |

vatek\_broadcast\_encoder\_setinputparm\_logo

Set the Logo input parameters of encoder.

|  |
| --- |
| vatek\_result vatek\_broadcast\_encoder\_setinputparm\_logo (  [In] Phbroadcast handle,  [In] logo\_input\_parm parm,  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*parm*

Please refer to [logo\_input\_parm](#logo_input_parm).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | The input parameter is over range. |
| vatek\_result\_busy | The broadcast system is busy. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Parameters set successfully. |

vatek\_broadcast\_encoder\_setencodeparm

Set the encoding parameters of encoder.

|  |
| --- |
| vatek\_result vatek\_broadcast\_encoder\_setencodeparm (  [In] Phbroadcast handle,  [In] video\_encode\_parm vparm,  [In] audio\_encode\_parm aparm,  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*vparm*

Please refer to [video\_encode\_parm](#video_encoder_parm).

*aparm*

Please refer to [audio\_encode\_parm](#audio_encode_parm).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | The input parameter is over range. |
| vatek\_result\_busy | The broadcast system is busy. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Parameters set successfully. |

vatek\_broadcast\_encoder\_setmuxparm

Set the PES / TS packet parameters of encoder.

|  |
| --- |
| vatek\_result vatek\_broadcast\_encoder\_setmuxparm (  [In] Phbroadcast handle,  [In] encoder\_mux\_parm parm  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*parm*

Please refer to [encoder\_mux\_parm](#encoder_mux_parm).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | The input parameter is over range. |
| vatek\_result\_busy | The broadcast system is busy. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Parameters set successfully. |

vatek\_broadcast\_encoder\_setqualityparm

Set the coding quality parameters of encoder.

|  |
| --- |
| vatek\_result vatek\_broadcast\_encoder\_setqualityparm (  [In] Phbroadcast handle,  [In] encoder\_quality\_parm parm  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*parm*

Please refer to [encoder\_quality\_parm](#encoder_quality_parm).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | The input parameter is over range. |
| vatek\_result\_busy | The broadcast system is busy. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Parameters set successfully. |

vatek\_broadcast\_tsmux\_setparm

Set the parameters of the TSMux.

|  |
| --- |
| vatek\_result vatek\_broadcast\_tsmux\_setparm (  [In] Phbroadcast handle,  [In] tsmux\_type type,  [In] Ptsmux\_parm parm  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*type*

TSMux type.

|  |  |
| --- | --- |
| **parameter** | **Meaning** |
| tsmux\_type\_pure | No PSI spec. |
| tsmux\_type\_iso13818 | Simple PAT and PMT spec. |
| tsmux\_type\_spec | Full PSI spec. |

*parm*

According to the transmission stream mode, different parameter structures are input.

For pure mode, please refer to [tsmux\_pure\_parm](#tsmux_pure_parm).

For iso13818mode, please refer to [tsmux\_iso13818\_parm](#tsmux_iso13818_parm).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | The input parameter is over range. |
| vatek\_result\_busy | The broadcast system is busy. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Parameters set successfully. |

vatek\_broadcast\_modulator\_setparm

Set the parameters of modulator module.

|  |
| --- |
| vatek\_result vatek\_broadcast\_modulator\_setparm (  [In] Phbroadcast handle,  [In] modulator\_base\_parm parm\_base,  [In] Pmodulator\_advanced\_parm parm\_adv  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*parm\_base*

Basic parameters of modulator, including modulation mode.

Please refer to [modulator\_base\_parm](#modulator_base_parm).

*parm\_adv*

The advanced parameters of modulator, input different parameter structures according to the modulation mode in the basic parameters.

dvb\_t standard, please refer to [modulator\_dvbt\_parm](#modulator_dvbt_parm).

j83a standard, please refer to [modulator\_j83a\_parm](#modulator_j83a_parm).

atsc standard, please refer to [modulator\_atsc\_parm](#modulator_atsc_parm).

j83b standard, please refer to [modulator\_j83b\_parm](#modulator_j83b_parm).

dtmb standard, please refer to [modulator\_dtmb\_parm](#modulator_dtmb_parm).

isdb\_t standard, please refer to [modulator\_isdbt\_parm](#modulator_isdbt_parm).

j83c standard, please refer to [modulator\_j83c\_parm](#modulator_j83c_parm).

dvb\_t2 standard, please refer to [modulator\_dvbt2\_parm](#modulator_dvbt2_parm).

sinewave mode, please refer to [modulator\_sinewave\_parm](#modulator_sinewave_parm).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | The input parameter is over range. |
| vatek\_result\_busy | The broadcast system is busy. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Parameters set successfully. |

vatek\_broadcast\_psitable\_register

Register PSI table list parameter (before TV broadcast system start).

|  |
| --- |
| vatek\_result vatek\_broadcast\_psitable\_register (  [In] Phbroadcast handle,  [In] Ppsitablelist\_parm parm  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*parm*

Please refer to [psitablelist\_parm](#psitablelist_parm).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | The length of PSI input is over range. |
| vatek\_result\_badstatus | An bad status. |
| vatek\_result\_busy | The broadcast system is busy. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Parameters set successfully. |

vatek\_broadcast\_psitable\_insert

Plug in the PSI table immediately.

|  |
| --- |
| vatek\_result vatek\_broadcast\_psitable\_insert (  [In] Phbroadcast handle,  [In] uint16\_t tspacket\_num,  [In] uint8\_t \*tspackets  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*tspacket\_num*

The number of TS packets in the PSI table.

*tspackets*

Data address of PSI table.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_badstatus | An bad status. |
| vatek\_result\_bufoverflow | There is not enough space to insert PSI table. |
| vatek\_result\_idle | Television has not yet started. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Parameters set successfully. |

vatek\_broadcast\_userdata\_insert\_cc

Insert **C**losed **C**aption Information to the video packet (after the start of the television broadcast system).

|  |
| --- |
| vatek\_result vatek\_broadcast\_userdata\_insert\_cc (  [In] Phbroadcast handle,  [In] uint16\_t cc\_num,  [In] Puserdata\_cc\_parm cc\_parm  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*cc\_num*

The maximum number of closed caption packets is 12. (MAX\_USERDATA\_CC\_NUM).

*cc\_parm*

Closed caption package, please refer to [userdata\_cc\_parm](#userdata_cc_parm).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | Number of packets over range. |
| vatek\_result\_busy | The previous CC data has not been processed. |
| vatek\_result\_unsupport | Video coding does not support CC. |
| vatek\_result\_idle | Television has not yet started. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Parameters set successfully. |

VATEK PERIPHERAL API (PHY)

vatek\_phy\_create

Establish the front-end data receiving system.

|  |
| --- |
| vatek\_result vatek\_phy\_create (  [In] Pboard\_handle hboard,  [In] phy\_type type,  [Out] Phphy \*handle  ); |

**Function Arguments**

*hboard*

The Handler of user environment I2C.

The Handler will be passed in while SDK calls vatek\_porting.

*type*

The Driver of the front-end data receiving system.

|  |  |
| --- | --- |
| **Parameter** | **Meaning** |
| phy\_type\_ep9555e | ep9555e front end data receiving system |
| phy\_type\_adv7182a\_cvbs | adv7182a front end data receiving system (CVBS mode) |
| phy\_type\_adv7182a\_ypbpr | adv7182a front end data receiving system (YPbPr mode) |

*handle*

Handler of front-end data receiving system.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | Unsupported Driver category. |
| vatek\_result\_memfail | Malloc failed. |
| vatek\_result\_badstatus | Initialization failed. |
| vatek\_result\_success | The front-end data receiving system is successfully established and sent back to the Handler. |

vatek\_phy\_destroy

Remove the front end data receiving system.

|  |
| --- |
| vatek\_result vatek\_phy\_destroy (  [In] Phphy handle  ); |

**Function Arguments**

*handle*

Handler of front-end data receiving system.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_success | Successfully removed the front-end data receiving system |

vatek\_phy\_enable

Turn on or off the front end data system.

|  |
| --- |
| vatek\_result vatek\_phy\_enable (  [In] Phphy handle,  [In] uint8\_t enable  ); |

**Function Arguments**

*handle*

Handler of front-end data receiving system.

*enable*

Turn on or off.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_unsupport | The handle is not supported by this API. |
| vatek\_result\_badstatus | A bad status. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Set successfully. |

vatek\_phy\_status

Read the front-end data system status.

|  |
| --- |
| vatek\_result vatek\_phy\_status (  [In] Phphy handle,  [Out] phy\_status \*status  ); |

**Function Arguments**

*handle*

Handler of front-end data receiving system.

*status*

Front end data system status.

|  |  |
| --- | --- |
| **Parameter** | **Meaning** |
| phy\_status\_lost | There is no connection signal. |
| phy\_status\_active | Normal operation. |
| phy\_status\_fail | An error occurred. |

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_unsupport | The handle is not supported by this API. |
| vatek\_result\_badstatus | A bad status. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Set successfully. |

vatek\_phy\_getvideoinfo

Read the front-end image information parameters.

|  |
| --- |
| vatek\_result vatek\_phy\_getvideoinfo (  [In] Phphy handle,  [Out]Pphy\_video\_info info  ); |

**Function Arguments**

*handle*

Handler of front-end data receiving system.

*info*

Front-end image information, please refer to [phy\_video\_info](#phy_video_info).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_badstatus | A bad status. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_unsupport | The handle is not supported by this API. |
| vatek\_result\_success | Set successfully. |

vatek\_phy\_getaudioinfo

Read the front-end sound information parameters.

|  |
| --- |
| vatek\_result vatek\_phy\_getaudioinfo (  [In] Phphy handle,  [Out] Pphy\_audio\_info info  ); |

**Function Arguments**

*handle*

Handler of front-end data receiving system.

*info*

For front-end sound information, please refer to [phy\_audio\_info](#phy_audio_info).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_badstatus | A bad status. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_unsupport | 此API不支援該handle |
| vatek\_result\_success | Set successfully. |

vatek\_phy\_setvideoinfo

Set the front-end image information parameters.

|  |
| --- |
| vatek\_result vatek\_phy\_setvideoinfo (  [In] Phphy handle,  [In] Pphy\_video\_info info  ); |

**Function Arguments**

*handle*

Handler of front-end data receiving system.

*info*

For front-end sound information, please refer to [phy\_video\_info](#phy_video_info).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_unsupport | The handle is not supported by this API. |
| vatek\_result\_badstatus | A bad status. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Set successfully. |

vatek\_phy\_getccdata

Read the closed caption information of the front-end image.

|  |
| --- |
| vatek\_result vatek\_phy\_getccdata (  [In] Phphy handle,  [Out] Puserdata\_cc\_parm cc\_parm  ); |

**Function Arguments**

*handle*

Handler of front-end data receiving system.

*cc\_parm*

For closed caption information, please refer to [userdata\_cc\_parm](#userdata_cc_parm).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_unsupport | The handle is not supported by this API. |
| vatek\_result\_badstatus | A bad status. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Set successfully. |

VATEK PERIPHERAL API (RF)

vatek\_rf\_create

Establish the back-end RF system.

|  |
| --- |
| vatek\_result vatek\_rf\_create (  [In] Pboard\_handle hboard,  [In] rf\_type type,  [Out] Phrf \*handle  ); |

**Function Arguments**

*hboard*

The Handler of the user environment, the Handler will be passed in while the SDK calls vatek\_porting.

This Handler will be different depending on the Driver category.

|  |  |
| --- | --- |
| **Driver Category** |  |
| rf\_type\_r2\_via\_vatek | Environment I2C Handler. |
| rf\_type\_r2 | Environment simulation SPI Handler, please refer to [r2\_board\_handle](#r2_board_handle). |

*type*

The Driver category of the back-end RF system.

|  |  |
| --- | --- |
| **parameter** | **Meaning** |
| rf\_type\_r2\_via\_vatek | Establish R2 back-end system (controlled by VATEK chip). |
| rf\_type\_r2 | Establish R2 back-end system. |

*handle*

The Handler of the back-end RF system.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | Unsupported Driver category. |
| vatek\_result\_memfail | Malloc failed. |
| vatek\_result\_badstatus | Initialization failed. |
| vatek\_result\_success | Successfully removed the back-end RF system. |

vatek\_rf\_destroy

Remove the back-end RF system.

|  |
| --- |
| vatek\_result vatek\_rf\_destroy (  [In] Phrf handle  ); |

**Function Arguments**

*handle*

Handler of back-end RF system.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_success | Successfully removed the back-end RF system. |

vatek\_rf\_start

Start the back-end RF system.

|  |
| --- |
| vatek\_result vatek\_rf\_start (  [In] Phrf handle,  [In] uint32\_t freq  ); |

**Function Arguments**

*handle*

Handler of back-end RF system.

freq

RF frequency point. The unit is KHz.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_badstatus | Wrong RF status (non idle). |
| vatek\_result\_timeout | RF control timeout (10 seconds). |
| vatek\_result\_hwfail | RF chip fail. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Successful start of RF system. |

vatek\_rf\_stop

Stop the back-end RF system.

|  |
| --- |
| vatek\_result vatek\_rf\_stop (  [In] Phrf handle,  ); |

**Function Arguments**

*handle*

Handler of back-end RF system.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_badstatus | Wrong RF status (inactive). |
| vatek\_result\_timeout | RF control timeout (10 seconds). |
| vatek\_result\_hwfail | RF chip fail. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | RF system stopped successfully. |

vatek\_rf\_getstatus

Read back-end RF system status.

|  |
| --- |
| vatek\_result vatek\_rf\_getstatus (  [In] Phrf handle,  [Out] rf\_status \*status  ); |

**Function Arguments**

*handle*

Handler of back-end RF system.

*status*

Back-end RF system status.

|  |  |
| --- | --- |
| **parameter** | **Meaning** |
| rf\_status\_idle | The RF system is idle. |
| rf\_status\_active | The RF system is working. |
| rf\_status\_fail | RF system is in error state. |

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_i2cfail | Failed to use I2C.. |
| vatek\_result\_success | Read RF system status successfully. |

vatek\_rf\_setcalibrate

Set the calibration parameters of the back-end RF system.

It is used in RF automatic calibration tool.

|  |
| --- |
| vatek\_result vatek\_rf\_setcalibrate (  [In] Phrf handle,  [In] Prf\_calibrate parm  ); |

**Function Arguments**

*handle*

Handler of back-end RF system.

*parm*

RF calibration parameters, according to the type of RF Driver, input different parameter structure.

r2\_via\_vatekplease refer to [r2\_calibrate](#r2_calibrate).

r2 please refer to [r2\_calibrate](#r2_calibrate).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Set successfully. |

VATEK PORTING API (I2C)

vatek\_porting\_i2c\_set\_speed

Set the access speed of I2C.

|  |
| --- |
| vatek\_result vatek\_porting\_i2c\_set\_speed (  [In] Pboard\_handle hboard,  [In] uint32\_t speedkhz  ); |

**Function Arguments**

*hboard*

The user-defined Handler is registered when the VATek system is established.

*speedkhz*

I2C access speed, the unit is kHz.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_success | Set successfully. |

vatek\_porting\_i2c\_start

Start I2C control status.

|  |
| --- |
| vatek\_result vatek\_porting\_i2c\_start (  [In] Pboard\_handle hboard,  [In] uint8\_t devaddr,  [In] uint32\_t restart  ); |

**Function Arguments**

*hboard*

The user-defined Handler is registered when the VATek system is established.

*devaddr*

The address of the I2C device.

*restart*

Restart the I2C control status.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_success | Set successfully. |

vatek\_porting\_i2c\_write

Write data to I2C.

|  |
| --- |
| vatek\_result vatek\_porting\_i2c\_write (  [In] Pboard\_handle hboard,  [In] uint8\_t\* pbuf,  [In] uint32\_t len  ); |

**Function Arguments**

*hboard*

The user-defined Handler is registered when the VATek system is established.

*pbuf*

The address of the data.

*len*

The length of the data.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_success | Data written successfully. |

vatek\_porting\_i2c\_read

Read data for I2C.

|  |
| --- |
| vatek\_result vatek\_porting\_i2c\_read (  [In] Pboard\_handle hboard,  [Out] uint8\_t\* pbuf,  [In] uint32\_t len  ); |

**Function Arguments**

*hboard*

The user-defined Handler is registered when the VATek system is established.

*pbuf*

The address of the data.

*len*

The length of the data.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_success | Read data successfully. |

vatek\_porting\_i2c\_stop

End the I2C control status.

|  |
| --- |
| vatek\_result vatek\_porting\_i2c\_stop (  [In] Pboard\_handle hboard,  ); |

**Function Arguments**

*hboard*

The user-defined Handler is registered when the VATek system is established.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_success | Set successfully. |

VATEK PORTING API (GPIO)

vatek\_porting\_gpio\_write

Write GPIO.

|  |
| --- |
| vatek\_result vatek\_porting\_gpio\_write (  [In] gpio\_pin pin,  [In] uint8\_t val  ); |

**Function Arguments**

*pin*

GPIO pin Handler is registered when the VATek system is established.

*val*

The value is 0 (low) or 1 (high)..

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_success | Set successfully. |

vatek\_porting\_gpio\_read

Write GPIO.

|  |
| --- |
| vatek\_result vatek\_porting\_gpio\_read (  [In] gpio\_pin pin,  [Out] uint8\_t \*val  ); |

**Function Arguments**

*pin*

GPIO pin Handler is registered when the VATek system is established.

*val*

The value is 0 (low) or 1 (high)..

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_success | Set successfully. |

VATEK PORTING API (UART)

vatek\_porting\_uart\_receive

Receive UART data.

|  |
| --- |
| vatek\_result vatek\_porting\_uart\_receive (  [In] Pboard\_handle hboard,  [Out]uint8\_t\* pbuf,  [In] uint32\_t len,  [In] uint32\_t timeout  ); |

**Function Arguments**

*hboard*

The user-defined UART Handler is registered when the VATek system is established.

*pbuf*

Receiving data content.

*len*

The length of the data to be received.

*timeout*

Timeout setting.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_uartfail | Operation UART failed. |
| vatek\_result\_success | Set successfully. |

vatek\_porting\_uart\_transmit

Send UART data.

|  |
| --- |
| vatek\_result vatek\_porting\_uart\_receive (  [In] Pboard\_handle hboard,  [In] uint8\_t\* pbuf,  [In] uint32\_t len,  [In] uint32\_t timeout  ); |

**Function Arguments**

*hboard*

The user-defined UART Handler is registered when the VATek system is established.

*pbuf*

Transmit data content.

*len*

Length of data transmitted.

*timeout*

Timeout setting.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_uartfail | Operation UART failed. |
| vatek\_result\_success | Set successfully. |

VATEK PORTING API (SYSTEM)

vatek\_porting\_system\_gettick

Read the tick of the system.

|  |
| --- |
| uint32\_t vatek\_porting\_system\_gettick (  ); |

**Function Arguments**

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| Value | tick count |

vatek\_porting\_system\_delay

Set the system delay.

|  |
| --- |
| vatek\_result vatek\_porting\_system\_delay (  [In] uint32\_t msec  ); |

**Function Arguments**

*msec*

Delay time in milliseconds.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_success | Set successfully. |

vatek\_porting\_system\_mainreset

Hardware restart the VATek chip.

|  |
| --- |
| vatek\_result vatek\_porting\_system\_mainreset (  [In] Pboard\_handle hboard  ); |

**Function Arguments**

*hboard*

The user-defined Handler is registered when the VATek system is established.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_success | Set successfully. |

vatek\_porting\_system\_phyreset

Hardware restarts the front-end data receiving chip.

|  |
| --- |
| vatek\_result vatek\_porting\_system\_phyreset (  [In] Pboard\_handle hboard,  [In] phy\_type type  ); |

**Function Arguments**

*hboard*

The user-defined Handler is registered when the VATek system is established.

*type*

The Driver category of the front-end data receiving system.

|  |  |
| --- | --- |
| **Parameter** | **Meaning** |
| phy\_type\_ep9555e | EP9555e front-end data receiving system. |
| phy\_type\_adv7182a\_cvbs | ADV7182a front-end data receiving system (CVBS mode) |
| phy\_type\_adv7182a\_ypbpr | ADV7182a front-end data receiving system (YPbPr mode) |

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_success | Set successfully. |