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

A-series

Copyright © 2021 by VATEK

Contents

[STRUCTURES 5](#_Toc95900207)

[ts\_input\_parm 5](#_Toc95900208)

[tsp\_filter\_parm 6](#_Toc95900209)

[tsmux\_iso13818\_parm 7](#_Toc95900210)

[tsmux\_pure\_parm 8](#_Toc95900211)

[modulator\_base\_parm 9](#_Toc95900212)

[modulator\_dvbt\_parm 11](#_Toc95900213)

[modulator\_j83a\_parm 13](#_Toc95900214)

[modulator\_atsc\_parm 14](#_Toc95900215)

[modulator\_j83b\_parm 15](#_Toc95900216)

[modulator\_dtmb\_parm 16](#_Toc95900217)

[modulator\_isdbt\_parm 18](#_Toc95900218)

[modulator\_j83c\_parm 20](#_Toc95900219)

[modulator\_dvbt2\_parm 21](#_Toc95900220)

[modulator\_sinewave\_parm 25](#_Toc95900221)

[psitablelist\_parm 26](#_Toc95900222)

[enum\_list 27](#_Toc95900223)

[enum\_stream 28](#_Toc95900224)

[demod\_lockparm 30](#_Toc95900225)

[tuner\_board\_handle 31](#_Toc95900226)

[tuner\_lockparm 32](#_Toc95900227)

[r2\_board\_handle 33](#_Toc95900228)

[r2\_calibrate 34](#_Toc95900229)

[VATEK MAIN API (TRANSFORM) 35](#_Toc95900230)

[vatek\_transform\_create 35](#_Toc95900231)

[vatek\_transform\_destroy 36](#_Toc95900232)

[vatek\_transform\_start 37](#_Toc95900233)

[vatek\_transform\_stop 38](#_Toc95900234)

[vatek\_transform\_chipstatus 39](#_Toc95900235)

[vatek\_transform\_bcstatus 40](#_Toc95900236)

[vatek\_transform\_tsp\_setinputparm\_ts 41](#_Toc95900237)

[vatek\_transform\_tsp\_setfilterparm 42](#_Toc95900238)

[vatek\_transform\_tsmux\_setparm 43](#_Toc95900239)

[vatek\_transform\_modulator\_setparm 44](#_Toc95900240)

[vatek\_transform\_psitable\_register 45](#_Toc95900241)

[vatek\_transform\_psitable\_insert 46](#_Toc95900242)

[vatek\_transform\_enum\_getlist 47](#_Toc95900243)

[VATEK PERIPHERAL API (DEMOD) 48](#_Toc95900244)

[vatek\_demod\_create 48](#_Toc95900245)

[vatek\_demod\_destroy 49](#_Toc95900246)

[vatek\_demod\_i2cbypass 50](#_Toc95900247)

[vatek\_demod\_lock 51](#_Toc95900248)

[vatek\_demod\_getstatus 52](#_Toc95900249)

[VATEK PERIPHERAL API (TUNER) 53](#_Toc95900250)

[vatek\_tuner\_create 53](#_Toc95900251)

[vatek\_tuner\_destroy 54](#_Toc95900252)

[vatek\_tuner\_lock 55](#_Toc95900253)

[vatek\_tuner\_getstatus 56](#_Toc95900254)

[VATEK PERIPHERAL API (RF) 57](#_Toc95900255)

[vatek\_rf\_create 57](#_Toc95900256)

[vatek\_rf\_destroy 58](#_Toc95900257)

[vatek\_rf\_start 59](#_Toc95900258)

[vatek\_rf\_stop 60](#_Toc95900259)

[vatek\_rf\_getstatus 61](#_Toc95900260)

[vatek\_rf\_setcalibrate 62](#_Toc95900261)

[VATEK PORTING API (I2C) 63](#_Toc95900262)

[vatek\_porting\_i2c\_set\_speed 63](#_Toc95900263)

[vatek\_porting\_i2c\_start 64](#_Toc95900264)

[vatek\_porting\_i2c\_write 65](#_Toc95900265)

[vatek\_porting\_i2c\_read 66](#_Toc95900266)

[vatek\_porting\_i2c\_stop 67](#_Toc95900267)

[VATEK PORTING API (GPIO) 68](#_Toc95900268)

[vatek\_porting\_gpio\_write 68](#_Toc95900269)

[vatek\_porting\_gpio\_read 69](#_Toc95900270)

[VATEK PORTING API (UART) 70](#_Toc95900271)

[vatek\_porting\_uart\_receive 70](#_Toc95900272)

[vatek\_porting\_uart\_transmit 71](#_Toc95900273)

[VATEK PORTING API (SYSTEM) 72](#_Toc95900274)

[vatek\_porting\_system\_gettick 72](#_Toc95900275)

[vatek\_porting\_system\_delay 73](#_Toc95900276)

[vatek\_porting\_system\_mainreset 74](#_Toc95900277)

[vatek\_porting\_system\_phyreset 75](#_Toc95900278)

STRUCTURES

ts\_input\_parm

Input parameters of transport stream analysis module.

|  |
| --- |
| typedef struct \_ts\_input\_parm  {  uint8\_t serial;  uint8\_t clk\_inverse;  uint8\_t valid\_inverse;  uint8\_t sync\_inverse;  }ts\_input\_parm, \*Pts\_input\_parm; |

**Members**

*serial*

Serial transmission setting, 0 is parallel transmission.

*clk\_inverse*

TS Clock signal reverses.

*valid\_inverse*

TS Valid signal reverses.

*sync\_inverse*

TS Sync signal reverses.

tsp\_filter\_parm

Filter parameters of transport stream analysis module.

|  |
| --- |
| typedef struct \_tsp\_filter  {  uint16\_t pid;  uint8\_t replace;  uint16\_t replace\_pid;  }tsp\_filter, \*Ptsp\_filter;  typedef struct \_tsp\_filter\_parm  {  uint16\_t filter\_num;  tsp\_filter filter[MAX\_TSP\_FILTER\_NUM];  }tsp\_filter\_parm, \*Ptsp\_filter\_parm; |

**Members**

*filter\_num*

Number of filters.

*filter*

Filter parameters.

*pid*

Identification code of the transmission stream packet to be filtered.

*replace*

Replace the transport stream PID.

*replace\_pid*

The transport stream PID to be replaced.

tsmux\_iso13818\_parm

iso13818 transport stream parameters.

|  |
| --- |
| 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 stream PID. The range is 0~0x1FFF.

*padding\_pid*

Padding PID. The range is 0~0x1FFF.

*pmtpid*

PMT’s PID. The range is 0~0x1FFF.

*tsid*

TSID. The range is 0~0xFFFF.

*program\_num*

Channel program number. The range is 0~0xFFFF.

tsmux\_pure\_parm

pure transport stream parameters.

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

**Members**

*pcr\_pid*

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

*padding\_pid*

Padding PID. The range is 0~0x1FFF.

modulator\_base\_parm

Modulator base parameters.

|  |
| --- |
| 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*

Modulation Standard

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

*ifmode*

Modulation mode, preset IF 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

Modulator DVB-T parameters.

|  |
| --- |
| 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*

Constellation setting.

|  |  |
| --- | --- |
| **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*

**F**ast **F**ourier **T**ransform setting.

|  |  |
| --- | --- |
| **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*

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*

Constellation setting.

|  |  |
| --- | --- |
| **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*

Constellation setting.

|  |  |
| --- | --- |
| **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*

Constellation Setting.

|  |  |
| --- | --- |
| **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*

Constellation setting.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dtmb\_constellation\_qpsk | 4-bit quadrature phase offset 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 interleave length setting.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dtmb\_time\_interleaved\_disable | Disable |
| dtmb\_time\_interleaved\_240 | 240 MS |
| dtmb\_time\_interleaved\_720 | 720 MS |

*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. 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*

Constellation setting.

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

*fft*

**F**ast **F**ourier **T**ransform setting.

|  |  |
| --- | --- |
| **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*

Constellation setting.

|  |  |
| --- | --- |
| **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*

vbt2 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.

enum\_list

List the program list parameters.

|  |
| --- |
| typedef struct \_enum\_program  {  uint32\_t pcr\_pid;  uint32\_t pmt\_pid;  uint32\_t stream\_num;  Penum\_stream stream;  }enum\_program, \*Penum\_program;  typedef struct \_enum\_list  {  uint32\_t program\_num;  Penum\_program program;  }enum\_list, \*Penum\_list; |

**Members**

*program\_num*

Number of programs.

*program*

Program content parameters.

*pcr\_pid*

PCR PID.

*pmt\_pid*

PMT PID.

*stream\_num*

The number of data streams.。

*stream*

For stream parameters, please refer to [enum\_stream](#enum_stream).

enum\_stream

List the data stream parameters of the program.

|  |
| --- |
| typedef struct \_enum\_stream  {  stream\_type type;  uint32\_t encode\_type;  uint32\_t stream\_pid;  uint32\_t stream\_type;  uint32\_t esinfo\_len;  uint32\_t \*esinfo;  }enum\_stream, \*Penum\_stream; |

**Members**

*type*

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| stream\_type\_video | The data stream is video. |
| stream\_type\_audio | The data stream is audio. |
| stream\_type\_data | Other. |

*encode\_type*

Depending on the encode data stream, there are two parameter types.

*video\_encode\_type*

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| ve\_type\_mpeg2 | mpeg2 |
| ve\_type\_h264 | h264 |

*audio\_encode\_type*

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

*stream\_pid*

Stream PID.

*stream\_type*

The type of stream.

*esinfo\_len*

ES information length, attached to the original PMT form.

*esinfo*

ES information length, attached to the original PMT form.

demod\_lockparm

Lock parameter of demodulator.

|  |
| --- |
| typedef struct \_demod\_lockparm  {  demod\_modulator\_type type;  uint32\_t bandwidth;  }demod\_lockparm, \*Pdemod\_lockparm; |

**Members**

*type*

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| dm\_type\_dvbc | DVBC television broadcast signal |
| dm\_type\_dvbtx | DVBTx television broadcast signal |
| dm\_type\_dvbsx | DVBSx television broadcast signal |
| dm\_type\_isdbt | ISDBT television broadcast signal |

*bandwidth*

Lock bandwidth, unit: kHz.

tuner\_board\_handle

The initialization parameter of the tuner.

|  |
| --- |
| typedef struct \_tuner\_board\_handle  {  Pboard\_handle hboard;  void\* hdemod;  fpdemod\_i2cbypass i2cbypass;  }tuner\_board\_handle, \*Ptuner\_board\_handle; |

**Members**

*hboard*

The Handler of user environment I2C.

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

*hdemod*

The Handler of Demodulator needs to be set when the tuner is controlled by Demodulator.

*i2cbypass*

This Demodulator bypass I2C API will be called before operating the tuner action.

tuner\_lockparm

The lock parameter of the tuner.

|  |
| --- |
| typedef struct \_tuner\_lockparm  {  tuner\_modulator\_type type;  uint32\_t bandwidth;  uint32\_t freq;  }tuner\_lockparm, \*Ptuner\_lockparm; |

**Members**

*type*

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| tm\_type\_dvbc | DVBC television broadcast signal |
| tm\_type\_dvbtx | DVBTx television broadcast signal |
| tm\_type\_dtmb | DTMB television broadcast signal |
| tm\_type\_atsc | ATSC television broadcast signal |
| tm\_type\_isdbt | ISDBT television broadcast signal |
| tm\_type\_j83b | J83B television broadcast signal |

*bandwidth*

Lock bandwidth, unit: kHz.

*freq*

Lock frequency, unit: kHz.

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 (TRANSFORM)

vatek\_transform\_create

Establish a television broadcasting system.

|  |
| --- |
| vatek\_result vatek\_transform\_create (  [In] Pboard\_handle hboard,  [Out] Phtransform \*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\_transform\_destroy

Remove the television broadcasting system.

|  |
| --- |
| vatek\_result vatek\_transform\_destroy (  [In] Phtransform 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\_transform\_start

Turn on the TV broadcasting system.

|  |
| --- |
| vatek\_result vatek\_transform\_start (  [In] Phtransform 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\_transform\_stop

Stop the TV broadcasting system and reset the VATek chip.

|  |
| --- |
| vatek\_result vatek\_transform\_stop (  [In] Phtransform 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\_transform\_chipstatus

Read the status of VATek chip.

|  |
| --- |
| vatek\_result vatek\_transform\_chipstatus (  [In] Phtransform 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\_transform\_bcstatus

Read the state of VATek chip.

|  |
| --- |
| vatek\_result vatek\_transform\_bcstatus (  [In] Phtransform 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 states |
| bc\_status\_idle | Waiting for broadcast status |
| bc\_status\_wait\_source | Waiting for the front source to prepare |
| bc\_status\_finish | Enumeration complete |
| bc\_status\_broadcast | Broadcast execution status |
| bc\_status\_fail\_unknown | Unknown error |
| bc\_status\_fail\_source | Front end source error |
| bc\_status\_fail\_timeout | Broadcast timeout error |
| bc\_status\_fail\_codecdrop | Encoder drop error |
| bc\_status\_fail\_buffer | Buffer error |
| bc\_status\_fail\_muxer | Muxer error |
| bc\_status\_fail\_encode | Coding error |
| bc\_status\_fail\_media | Multimedia error |
| bc\_status\_fail\_demux | Stream parsing 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\_transform\_tsp\_setinputparm\_ts

Set the input parameters of the TS analysis module.

|  |
| --- |
| vatek\_result vatek\_transform\_tsp\_setinputparm\_ts (  [In] Phtransform handle,  [In] ts\_input\_parm parm  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*parm*

Input parameters of transport stream analysis module, please refer to [ts\_input\_parm](#ts_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\_transform\_tsp\_setfilterparm

Set the filter parameters of the TS analysis module.

|  |
| --- |
| vatek\_result vatek\_transform\_tsp\_setfilterparm (  [In] Phtransform handle,  [In] tsp\_filter\_parm parm  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*parm*

Please refer to [tsp\_filter\_parm](#tsp_filter_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\_transform\_tsmux\_setparm

Set the filter parameters of the transport stream analysis module.

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

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*type*

Transport stream mode.

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

*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\_transform\_modulator\_setparm

Set the parameters of modulator module.

|  |
| --- |
| vatek\_result vatek\_transform\_modulator\_setparm (  [In] Phtransform 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\_transform\_psitable\_register

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

|  |
| --- |
| vatek\_result vatek\_transform\_psitable\_register (  [In] Phtransform 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\_transform\_psitable\_insert

Plug in the PSI table immediately.

|  |
| --- |
| vatek\_result vatek\_transform\_psitable\_insert (  [In] Phtransform 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 | A 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\_transform\_enum\_getlist

According to PAT and PMT table, the content of TS program will be listed.

|  |
| --- |
| vatek\_result vatek\_transform\_enum\_getlist (  [In] Phtransform handle,  [Out] Penum\_list \*list  ); |

**Function Arguments**

*handle*

The Handler of TV broadcasting system.

*list*

Listed program content, please refer to [enum\_list](#enum_list).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **Meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_hwfail | Enumeration failure. |
| vatek\_result\_timeout | Enumeration timeout (10 seconds). |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Parameters set successfully. |

VATEK PERIPHERAL API (DEMOD)

vatek\_demod\_create

Establish the front-end demodulator.

|  |
| --- |
| vatek\_result vatek\_demod\_create (  [In] Pboard\_handle hboard,  [In] demod\_type type,  [Out] Phdemod \*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** |
| demod\_type\_avl68xx | Demodulator avl68xx |

*handle*

Handler of front-end demodulator.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | Unsupported Driver category. |
| vatek\_result\_memfail | Malloc failed. |
| vatek\_result\_hwfail | Demodulator failed. |
| vatek\_result\_badstatus | Initialization failed. |
| vatek\_result\_success | The demodulator is successfully established, and sent back to the Handler. |

vatek\_demod\_destroy

Remove the front-end demodulator.

|  |
| --- |
| vatek\_result vatek\_demod\_destroy (  Phdemod handle  ); |

**Function Arguments**

*handle*

Handler of front-end demodulator.

**Result**

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

vatek\_demod\_i2cbypass

Bypass I2C, so that I2C can control more front-end Turner.

|  |
| --- |
| vatek\_result vatek\_demod\_i2cbypass (  [In] Phdemod handle,  [In] uint8\_t bypasson  ); |

**Function Arguments**

*handle*

Handler of front-end demodulator.

*bypasson*

Start I2C bypass.

**Result**

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

vatek\_demod\_lock

Lock the front demodulator signal.

|  |
| --- |
| vatek\_result vatek\_demod\_lock (  [In] Phdemod handle,  [In] Pdemod\_lockparm parm  ); |

**Function Arguments**

*handle*

Handler of front-end demodulator.

*parm*

Front-end demodulator lock parameters, please refer to [demod\_lockparm](#demod_lockparm).

**Result**

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

vatek\_demod\_getstatus

Read the front-end demodulator status.

|  |
| --- |
| vatek\_result vatek\_demod\_getstatus (  [In] Phdemod handle,  [Out] demod\_status \*status  ); |

**Function Arguments**

*handle*

Handler of front-end demodulator.

*status*

The lock status of the front-end demodulator.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| demod\_status\_lock | Lock |
| demod\_status\_unlock | Unlock |

**Result**

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

VATEK PERIPHERAL API (TUNER)

vatek\_tuner\_create

Establish the front-end tuner.

|  |
| --- |
| vatek\_result vatek\_tuner\_create (  [In] Ptuner\_board\_handle hboard,  [In] tuner\_type type,  [Out] Phtuner \*handle  ); |

**Function Arguments**

*hboard*

Turner Handler for the consumer environment, please refer to [tuner\_board\_handle](#tuner_board_handle).

*type*

The Driver category of the front-end tuner.

|  |  |
| --- | --- |
| **Parameter** | **Meaning** |
| tuner\_type\_mxl608 | Demodulator mxl608 |

*handle*

The Handler of the front-end tuner.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_overrange | Unsupported Driver category. |
| vatek\_result\_memfail | Malloc failed. |
| vatek\_result\_hwfail | Tuner failed. |
| vatek\_result\_badstatus | Initialization failed. |
| vatek\_result\_success | The tuner is successfully established and returned to the handler. |

vatek\_tuner\_destroy

Remove the front-end tuner.

|  |
| --- |
| vatek\_result vatek\_tuner\_destroy (  [In] Phtuner handle  ); |

**Function Arguments**

*handle*

The Handler of the front-end tuner.

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_hwfail | Tuner failed. |
| vatek\_result\_success | Successfully remove tuner. |

vatek\_tuner\_lock

Lock the front tuner signal.

|  |
| --- |
| vatek\_result vatek\_tuner\_lock (  [In] Phtuner handle,  [In] Ptuner\_lockparm parm  ); |

**Function Arguments**

*handle*

The Handler of the front-end tuner.

*parm*

Front-end tuner lock parameters, please refer to [tuner\_lockparm](#tuner_lockparm).

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_unsupport | The handle is not supported by this API. |
| vatek\_result\_hwfail | Tuner failed. |
| vatek\_result\_success | Set successfully. |

vatek\_tuner\_getstatus

Read the front-end tuner status.

|  |
| --- |
| vatek\_result vatek\_tuner\_getstatus (  [In] Phtuner handle,  [Out] tuner\_status \*status  ); |

**Function Arguments**

*handle*

The Handler of the front-end tuner.

*status*

The lock status of the front-end tuner.

|  |  |
| --- | --- |
| **parameter** | **meaning** |
| tuner\_status\_lock | Lock |
| tuner\_status\_unlock | Unlock |

**Result**

|  |  |
| --- | --- |
| **vatek\_result** | **meaning** |
| vatek\_result\_invalidparm | Invalid input parameter. |
| vatek\_result\_unsupport | The handle is not supported by this API. |
| vatek\_result\_hwfail | Tuner failed |
| vatek\_result\_success | Read status 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 user environment I2C.

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

*type*

This Handler will be different depending on the Driver category.

|  |  |
| --- | --- |
| **Driver Catagory** |  |
| 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 backend 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 | The back-end RF system is successfully established and sent back to the handler. |

vatek\_rf\_destroy

Remove the back-end RF system.

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

**Function Arguments**

*handle*

The Handler of the 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*

The Handler of the 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 | Bad RF status (non idle). |
| vatek\_result\_timeout | RF control timeout (10 seconds). |
| vatek\_result\_hwfail | RF chip failed. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | Successful start RF system. |

vatek\_rf\_stop

Stop the back-end RF system.

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

**Function Arguments**

*handle*

The Handler of the 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 error. |
| vatek\_result\_i2cfail | Failed to use I2C. |
| vatek\_result\_success | RF system stopped successfully. |

vatek\_rf\_getstatus

Read the back-end RF system.

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

**Function Arguments**

*handle*

The Handler of the 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*

The Handler of the 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).

r2please 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 state.

**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 from 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

Stop 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, which 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

Read GPIO.

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

**Function Arguments**

*pin*

GPIO pin Handler, which 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 | Failed to operate UART. |
| 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 | Failed to operate UART. |
| 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** |
| 數值 | 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 restart 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. |