Library Reference

Index

```
int setBaudRate(unsigned long br);
int setIOMode(io_mode_t mode);
int resetIO(uint8_t *ios=0, uint8_t len=1);
int setPulseWidth(uint8_t level);
int setAutoLoad(uint8_t *records=0, uint8_t len = 0);
int disableAutoLoad();
int restoreSystemSettings();
int checkSystemSettings(uint8_t* buf);
int recognize(uint8_t *buf, int timeout = VR_DEFAULT_TIMEOUT);
int train(uint8_t *records, uint8_t len=1, uint8_t *buf = 0);
int train(uint8_t record, uint8_t *buf = 0);
int trainWithSignature(uint8_t record, const void *buf, uint8_t len=0, uint8_t *retbuf = 0);
int load(uint8_t *records, uint8_t len=1, uint8_t *buf = 0);
int load(uint8_t record, uint8_t *buf = 0);
int clear();
int setSignature(uint8_t record, const void *buf=0, uint8_t len=0);
int deleteSignature(uint8_t record);
int checkSignature(uint8_t record, uint8_t *buf);
int checkRecognizer(uint8_t *buf);
int checkRecord(uint8_t *buf, uint8_t *records = 0, uint8_t len = 0);

int setGroupControl(uint8_t ctrl);
int checkGroupControl();
int setUserGroup(uint8_t grp, uint8_t *records, uint8_t len);
int checkUserGroup(uint8_t grp, uint8_t *buf);
int loadSystemGroup(uint8_t grp, uint8_t *buf=0);
int loadUserGroup(uint8_t grp, uint8_t *buf=0);

void send_pkt(uint8_t *buf, uint8_t len);
void send_pkt(uint8_t cmd, uint8_t *buf, uint8_t len);
void send_pkt(uint8_t cmd, uint8_t subcmd, uint8_t *buf, uint8_t len);
int receive(uint8_t *buf, int len, uint16_t timeout = VR_DEFAULT_TIMEOUT);
int receive_pkt(uint8_t *buf, uint16_t timeout = VR_DEFAULT_TIMEOUT);
```

Details

```
/**
    @brief VR class constructor.
    @param receivePin --> software serial RX
           transmitPin --> software serial TX
*/
VR::VR(uint8_t receivePin, uint8_t transmitPin) : SoftwareSerial(receivePin, transmitPin)

/**
    @brief VR class constructor.
    @param buf --> return data .
              buf[0]  -->  Group mode(FF: None Group, 0x8n: User, 0x0n:System
           buf[1]  -->  number of record which is recognized.
           buf[2]  -->  Recognizer index(position) value of the recognized record.
           buf[3]  -->  Signature length
           buf[4]~buf[n] --> Signature
           timeout --> wait time for receiving packet.
    @retval length of valid data in buf. 0 means no data received.
*/
int VR :: recognize(uint8_t *buf, int timeout)
```

```
/**
    @brief train records, at least one.
    @param records --> record data buffer pointer.
           len --> number of records.
         buf --> pointer of return value buffer, optional.
           buf[0]  -->  number of records which are trained successfully.
          buf[2i+1]  -->  record number
          buf[2i+2]  -->  record train status.
            00 --> Trained
            FE --> Train Time Out
            FF --> Value out of range"
          (i = 0 ~ len-1 )
    @retval '>0' --> length of valid data in buf.
          0 --> success, and no data received.
          '<0' --> failed.
            -1 --> data format error.
            -2 --> train timeout.
*/
int VR :: train(uint8_t *records, uint8_t len, uint8_t *buf)

/**
    @brief train one record.
    @param records --> record data buffer pointer.
           len --> number of records.
         buf --> pointer of return value buffer, optional.
           buf[0]  -->  number of records which are trained successfully.
          buf[2i+1]  -->  record number
          buf[2i+2]  -->  record train status.
            00 --> Trained
            FE --> Train Time Out
            FF --> Value out of range"
          (i = 0 ~ len-1 )
    @retval '>0' --> length of valid data in buf.
          0 --> success, and no data received.
          '<0' --> failed.
            -1 --> data format error.
            -2 --> train timeout.
*/
int VR :: train(uint8_t record, uint8_t *buf)
```

```
/**
    @brief train record and set a signature(alias) for this record.
    @param record --> record value.
         buf --> signature string/data pointer.
           len --> lenght of buf.
           retbuf --> return data .
              retbuf[0]  -->  number of records which are trained successfully.
           retbuf[1]  -->  record number.
           retbuf[2]  -->  record train status.
              00 --> Trained
              F0 --> Trained, signature truncate
              FE --> Train Time Out
              FF --> Value out of range"
           retbuf[3] ~ retbuf[retval-1] --> Signature.(retval means return value)
    @retval '>0' --> length of valid data in buf.
          0 --> success, and no data received.
          '<0' --> failed.
              -1 --> data format error.
              -2 --> train with signature timeout.
*/
int VR :: trainWithSignature(uint8_t record, const void *buf, uint8_t len, uint8_t * retbuf)

/**
   @brief Load records to recognizer.
   @param records --> record data buffer pointer.
         len --> number of records.
         buf --> pointer of return value buffer, optional.
           buf[0]    -->  number of records which are load successfully.
           buf[2i+1]  -->  record number
           buf[2i+2]  -->  record load status.
              00 --> Loaded
              FC --> Record already in recognizer
              FD --> Recognizer full
              FE --> Record untrained
              FF --> Value out of range"
           (i = 0 ~ '(retval-1)/2' )
    @retval '>0' --> length of valid data in buf.
          0 --> success, buf=0, and no data returned.
          '<0' --> failed.
*/
int VR :: load(uint8_t *records, uint8_t len, uint8_t *buf)
```

```
/**
   @brief Load one record to recognizer.
   @param record --> record value.
         buf --> pointer of return value buffer, optional.
          buf[0]   -->  number of records which are load successfully.
          buf[2i+1]  -->  record number
          buf[2i+2]  -->  record load status.
             00 --> Loaded
             FC --> Record already in recognizer
             FD --> Recognizer full
             FE --> Record untrained
             FF --> Value out of range"
          (i = 0 ~ '(retval-1)/2' )
    @retval '>0' --> length of valid data in buf.
          0 --> success, buf=0, and no data returned.
          '<0' --> failed.
*/
int VR :: load(uint8_t record, uint8_t *buf)

/**
   @brief set signature(alias) for a record.
   @param record --> record value.
         buf --> signature buffer.
         len --> length of buf.
   @retval 0 --> success, buf=0, and no data returned.
          '<0' --> failed.
*/
int VR :: setSignature(uint8_t record, const void *buf, uint8_t len)

/**
   @brief delete signature(alias) of a record.
   @param record --> record value.
   @retval  0 --> success
          -1 --> failed
*/
int VR :: deleteSignature(uint8_t record)
```

```
/**
   @brief check the signature(alias) of a record.
   @param record --> record value.
         buf --> signature, return value buffer.
   @retval '>0' --> length of valid data in buf.
          0 --> success, buf=0, and no data returned.
          '<0' --> failed.
*/
int VR :: checkSignature(uint8_t record, uint8_t *buf)

/**
   @brief clear recognizer.
   @retval  0 --> success
          -1 --> failed
*/
int VR :: clear()

/**
   @brief clear recognizer.
   @param buf --> return value buffer.
           buf[0]    -->  Number of valid voice records in recognizer
           buf[i+1]   -->  Record number.(0xFF: Not loaded(Nongroup mode), or not set (Group
mode))
             (i= 0, 1, ... 6)
           buf[8]    -->  Number of all voice records in recognizer
           buf[9]    -->  Valid records position indicate.
           buf[10]   -->  Group mode indicate(FF: None Group, 0x8n: User, 0x0n:System
   @retval '>0' --> success, length of data in buf
          -1 --> failed
*/
int VR :: checkRecognizer(uint8_t *buf)

/**
   @brief check record train status.
   @param buf --> return value
           buf[0]    -->  Number of checked records
           buf[2i+1]  -->  Record number.
           buf[2i+2]  -->  Record train status. (00: untrained, 01: trained, FF: record
value out of range)
           (i = 0 ~ buf[0]-1 )
   @retval Number of trained records
*/
int VR :: checkRecord(uint8_t *buf, uint8_t *records, uint8_t len)
```

```
/****************************************************************************/
/****************************** GROUP CONTROL ******************************/
/**
   @brief set group control by external IO function
   @param ctrl --> group control by external IO
             0 --> disable group control by external IO
             1 --> user group control by external IO
             2 --> system group control by external IO
   @retval  0 --> success
          -1 --> failed
*/
int VR :: setGroupControl(uint8_t ctrl)

/**
   @brief check group control by external IO function
   @param ctrl --> group control by external IO
   @retval  0 --> group control by external IO disabled
           1 --> user group control by external IO status
           2 --> system group control by external IO status
          -1 --> failed
*/
int VR :: checkGroupControl()

/**
   @brief set user gruop content.
   @param grp --> user group number.
         records --> pointer of records buffer.
         len --> length of reocrds
   @retval  0 --> success
          -1 --> failed
*/
int VR :: setUserGroup(uint8_t grp, uint8_t *records, uint8_t len)
```

```
/**
   @brief check user gruop content.
   @param grp --> user group number.
         buf --> return value
          buf[8i]   -->  group number.
          buf[8i+1]  -->  group position 0 status.
          buf[8i+2]  -->  group position 1 status.
            ...              ...
          buf[8i+6]  -->  group position 5 status.
          buf[8i+7]  -->  group position 6 status.
          (i = 0 ~ @retval)
   @retval '>0' --> number of checked user group
          '<0' --> failed
*/
int VR :: checkUserGroup(uint8_t grp, uint8_t *buf)

/**
   @brief load system gruop content to recognizer.
   @param grp --> syestem group number.
         buf  -->  return value.
          buf[0]    -->  Number of valid voice records in recognizer.
          buf[i+1]   -->  Record number.(0xFF: Not loaded(Nongroup mode), or not set (Group
mode))
             (i= 0, 1, ... 6)
          buf[8]    -->  Number of all voice records in recognizer
          buf[9]    -->  Valid records position indicate.
          buf[10]   -->  Group mode indicate(FF: None Group, 0x8n: User, 0x0n:System
          (i = 0 ~ @retval)
   @retval '>0' --> length of buf
          '<0' --> failed
*/
int VR :: loadSystemGroup(uint8_t grp, uint8_t *buf)
```

```
/**
   @brief load user gruop content to recognizer.
   @param grp --> user group number.
         buf  -->  return value.
           buf[0]    -->  Number of valid voice records in recognizer.
           buf[i+1]   -->  Record number.(0xFF: Not loaded(Nongroup mode), or not set (Group
mode))
             (i= 0, 1, ... 6)
           buf[8]    -->  Number of all voice records in recognizer
           buf[9]    -->  Valid records position indicate.
           buf[10]   -->  Group mode indicate(FF: None Group, 0x8n: User, 0x0n:System)
           (i = 0 ~ @retval)
   @retval '>0' --> length of buf
          '<0' --> failed
*/
int VR :: loadUserGroup(uint8_t grp, uint8_t *buf)

/**
   @brief reset system setting to default
   @retval  0 --> success
          -1 --> failed
*/

int VR :: restoreSystemSettings()

/**
   @brief check system settings
   @param buf --> return value
          buf[0] --> baud rate. (0-9600 1-2400 2-4800 3-9600 4-19200 5-38400)
          buf[1] --> output io mode(0-pulse 1-toggle 2-clear 3-set)
          buf[2] --> pulse width level
          buf[3] --> auto load(0,0xFF-disable 1-enable)
          buf[4] --> Group control by external IO(0-disable 1-system group 2-user group)
   @retval '>0' --> buf length
          -1 --> failed
*/
int VR :: checkSystemSettings(uint8_t* buf)
```

```
/**
   @brief set module baud rate.
   @param br --> module baud rate.(0-9600 1-2400 2-4800 3-9600 4-19200 5-38400)
   @retval 0 --> success
          -1 --> failed
*/
int VR :: setBaudRate(unsigned long br)

/**
   @brief set module output IO mode.
   @param mode --> module output IO mode.(must be PULSE, TOGGLE, SET, CLEAR)
   @retval 0 --> success
          -1 --> failed
*/
int VR :: setIOMode(io_mode_t mode)

/**
   @brief resset module output IO.
   @param ios --> output IO buffer.
         len --> length of ios.
   @retval 0 --> success
          -1 --> failed
*/
int VR :: resetIO(uint8_t *ios, uint8_t len)

/**
   @brief set module pulse width(PULSE mode).
   @param level --> pulse width level.(LEVEL0~LEVEL15)
         len --> length of ios.
   @retval 0 --> success
          -1 --> failed
*/
int VR :: setPulseWidth(uint8_t level)

/**
   @brief set autoload.
   @param records --> record buffer.
         len --> records length.
   @retval 0 --> success
          -1 --> failed
*/
int VR :: setAutoLoad(uint8_t *records, uint8_t len)
```

```
/**
   @brief disable autoload.
   @param records --> record buffer.
         len --> records length.
   @retval 0 --> success
          -1 --> failed
*/
int VR :: disableAutoLoad()

/**
   @brief send data packet in Voice Recognition module protocol format.
   @param cmd --> command
         subcmd --> subcommand
         buf --> data area
         len --> length of buf
*/
void VR :: send_pkt(uint8_t cmd, uint8_t subcmd, uint8_t *buf, uint8_t len)

/**
   @brief send data packet in Voice Recognition module protocol format.
   @param cmd --> command
         buf --> data area
         len --> length of buf
*/
void VR :: send_pkt(uint8_t cmd, uint8_t *buf, uint8_t len)

/**
   @brief send data packet in Voice Recognition module protocol format.
   @param buf --> data area
         len --> length of buf
*/
void VR :: send_pkt(uint8_t *buf, uint8_t len)

/**
   @brief receive a valid data packet in Voice Recognition module protocol format.
   @param buf --> return value buffer.
         timeout --> time of reveiving
   @retval '>0' --> success, packet lenght(length of all data in buf)
          '<0' --> failed
*/
int VR :: receive_pkt(uint8_t *buf, uint16_t timeout)
```

```
/**
   @brief receive data .
   @param buf --> return value buffer.
         len --> length expect to receive.
         timeout --> time of reveiving
   @retval number of received bytes, 0 means no data received.
*/
int VR::receive(uint8_t *buf, int len, uint16_t timeout)
```