**Riftek RF627 Laser Scanner Software Development Kit**

v. 18.10.17.0

**Class name:**

rf627

**Static methods**

static bool **init**();

Must be called once before using the library

Returns true on success

static void **cleanup**();

Must be called at the end of program to free resources allocated in init()

static rf627\_list **search**(bool \*ok = nullptr);

Perform search for sensors over network

ok: optional pointer to boolean which is set to false if error occurs

Returns a list of sensors found

static uint32\_t **version**();

Returns SDK version

**Public methods**

rf627(rf627\_device\_info\_t\* info, uint16\_t init\_msg\_count = 1);

Constructs rf627 object

info: a pointer to structure from rf627\_list returned by search method

bool **connect**();

Creates and configures UDP sockets for data exchange with scanner. Must be called before using of any methods below

Returns false on error

void **disconnect**();

Closes sockets opened by connect method

bool **read\_params**(rf627\_user\_params\_t\* pparams = nullptr);

Reads configuration parameters block from scanner memory

pparams: a pointer to configuration parameters structure to store

Returns false on error

bool **write\_params**(rf627\_user\_params\_t\* pparams = nullptr);

Writes configuration parameters block to scanner memory

pparams: a pointer to configuration parameters structure to write

Returns false on error

bool **read\_sysmon\_params**(rf627\_sysmon\_params\_t\* psysmon = nullptr);

Reads system monitor values from scanner memory. This is also being read by read\_params method as part of ref627\_user\_params structure

pparams: a pointer to sysmon structure to store

Returns false on error

bool **flush\_params**();

Saves configuration parameters to flash memory

Returns false on error

bool **reset\_params**();

Restores configuration parameters to factory set

Returns false on error

bool **reboot**();

Reboots scanner

Returns false on error

bool **get\_result**(rf627\_profile& profile);

Reads profile data from stream

profile: reference to the profile structure

Returns false on error

bool **get\_image**(uint8\_t\* ppixmap);

Requests and reads an image from scanner

ppixmap: a pointer to 8-bit pixel array. Array size is RF627\_IMAGE\_SIZE. Each byte represents a pixel brightness in range of 0-255 (black to bright)

Returns false on error

bool **write\_firmware\_image**(const char\* file\_name);

Writes firmware image into memory

file\_name: name of firmware image file (.rf627)

Returns false on error

bool **flush\_firmware\_image**();

Saves firmware image transferred by write\_firmware\_image to flash. Scanner will be rebooted on success to boot new firmware

Returns false on error

bool **read\_log**(uint32\_t nstart\_line, rf627\_log\_record\_t \*plog\_entries, int nlines);

Reads log entries from scanner memory

nstart\_line: number of first line to read

plog\_entries: a pointer to array of rf627\_log\_record\_t structures to store data read

nlines: number of lines to read. Maximum value is limited to 10 for now.

Returns false on error

uint32\_t **read\_log\_record\_count**();

Gets a total number of log entries from scanner

Returns number of entries or 0 on error

const char \***error\_msg**();

Gets a pointer to the last error message

**Data structures**

struct rf627\_point

{

double x;

double z;

};

Coordinates of a profile point (millimeters)

struct rf627\_profile

{

rf627\_stream\_msg\_t header;

std::vector<rf627\_point> points;

};

Profile read from stream

typedef enum: uint8\_t

{

DTY\_PixelsNormal = 0x10, //Pixels (up to 648 points)

DTY\_ProfileNormal = 0x11, //Profile (up to 648 points)

DTY\_PixelsInterpolated = 0x12, //2x interpolated pixels (up to 1296)

DTY\_ProfileInterpolated = 0x13 //2x interpolated profile (up to 1296)

} data\_type\_t;

Profile type

#pragma pack(push,1)

typedef struct

{

data\_type\_t data\_type; //Profile type (one of data\_type\_t enum)

uint8\_t flags; //Flags (bit 7 indicates that packet requires acknowledgement from host)

uint16\_t device\_type; //627

uint32\_t serial\_number;

uint64\_t system\_time;

uint8\_t proto\_version\_major;

uint8\_t proto\_version\_minor;

uint8\_t hardware\_params\_offset;

uint8\_t data\_offset;

uint32\_t packet\_count; //Sequential number of packet emitted

uint32\_t measure\_count; //Seq. number of measurement

uint16\_t zmr; //Measurement range by Z

uint16\_t xemr; //Range by X at end of Z range

uint16\_t discrete\_value;

uint8\_t reserved\_0[14];

uint32\_t exposure\_time; //Exposure time used to take a measure

uint32\_t laser\_value;

uint32\_t step\_count; //STEP value in STEP/DIR mode

uint8\_t dir; //DIR value

uint8\_t reserved\_1[3];

}

rf627\_stream\_msg\_t;

#pragma pack(pop)

A header of profile packet

#pragma pack(push,1)

typedef struct

{

char name[64]; //Readable name of scanner

uint16\_t device\_id; //627

uint32\_t serial\_number;

uint32\_t firmware\_version;

uint32\_t hardware\_version;

uint32\_t config\_version;

uint32\_t fsbl\_version;

uint32\_t z\_begin; //Begin of Z range

uint32\_t z\_range; // Measurement range by Z

uint32\_t x\_smr; //Meas. Range by X at start of Z range

uint32\_t x\_emr; //Meas. Range by X at end of Z range

uint8\_t reserved\_0[36];

uint16\_t eth\_speed; //100 or 1000 (Mbps)

uint32\_t ip\_address;

uint32\_t net\_mask;

uint32\_t gateway\_ip;

uint32\_t host\_ip; //IP address of host receiving data stream

uint16\_t stream\_port; //Host port number of data stream

uint16\_t http\_port;

uint16\_t service\_port;

uint16\_t eip\_broadcast\_port;

uint16\_t eip\_port;

uint8\_t hardware\_address[6];

uint8\_t reserved\_1[26];

uint32\_t max\_payload\_size;

uint8\_t reserved\_2[32];

uint8\_t stream\_enabled; //Nonzero if data stream is enabled

uint8\_t stream\_format; //Profile type (data\_type\_t & 0x0F)

uint8\_t reserved\_3[32];

uint8\_t reserved\_4[256];

}

rf627\_device\_info\_t;

#pragma pack(pop)

Scanner information

#pragma pack(push,1)

typedef struct

{

char name[64]; //Readable scanner name

uint8\_t reserved[128];

}

rf627\_general\_params\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

int16\_t fpga\_temp; //FPGA temperature (Celsius \* 10)

uint8\_t reserved[80];

}rf627\_sysmon\_params\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

uint8\_t dhs; //Enable double speed mode

uint8\_t gain\_analog;

uint8\_t gain\_digital;

uint32\_t exposure; //Exposure time, ns

uint32\_t max\_exposure;

uint32\_t frame\_rate; //Frame rate limitation

uint32\_t max\_frame\_rate;

uint8\_t reserved\_0;

uint8\_t auto\_exposure; //Enable auto-exposure

uint8\_t frame\_by\_request;

uint8\_t reserved\_1[61];

}rf627\_sensor\_params\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

uint8\_t enable; //Switch scanner to RF625 emulation mode

uint16\_t tcp\_port;

uint8\_t reserved[32];

}rf627\_rf625compat\_params\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

uint8\_t enable; //Manually enable region of interest

uint8\_t active;

uint16\_t window\_height;

uint8\_t position\_mode; //0 – manual, nonzero - auto

uint16\_t window\_top;

uint16\_t current\_window\_top; //Current ROI top in auto mode

uint16\_t profile\_size; //Required profile size

uint8\_t reserved[80];

}rf627\_roi\_params\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

uint16\_t speed; //100 or 1000 Mbps

uint8\_t autonegotiation; //Auto-detect speed

uint32\_t ip\_address;

uint32\_t net\_mask;

uint32\_t gateway\_ip;

uint32\_t host\_ip;

uint16\_t stream\_port;

uint16\_t http\_port;

uint16\_t service\_port;

uint16\_t eip\_broadcast\_port;

uint16\_t eip\_port;

uint8\_t reserved[64];

}rf627\_network\_params\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

uint8\_t enable; //Enable data stream

uint8\_t format; //Profile type (data\_type\_t & 0x0F)

uint8\_t ack; //Each data packet requires acknowledgement (0 is off, default)

uint8\_t reserved[32];

}rf627\_stream\_params\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

uint32\_t brightness\_threshold;

uint8\_t stg1\_filter\_width;

uint8\_t stg1\_processing\_mode;

uint8\_t stg2\_reduce\_noise;

uint32\_t frame\_rate;

uint8\_t reserved[60];

}rf627\_image\_processing\_params\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

uint8\_t enable; //0 to turn laser off

uint8\_t auto\_level;

uint16\_t level;

uint8\_t reserved[32];

}rf627\_laser\_params\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

uint16\_t params\_mask; //Each bit indicates if following parameters are customizable in selected mode (LSB is in1\_enable, etc.)

uint8\_t in1\_enable;

uint8\_t in1\_mode;

uint32\_t in1\_delay;

uint8\_t in1\_decimation;

uint8\_t in2\_enable;

uint8\_t in2\_mode;

uint8\_t in2\_invert;

uint8\_t in3\_enable;

uint8\_t in3\_mode;

uint8\_t reserved[12];

}rf627\_inputs\_preset\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

uint8\_t preset\_index; //Selected mode

rf627\_inputs\_preset\_t params[12];

uint8\_t reserved[32];

}rf627\_inputs\_params\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

uint8\_t out1\_enable;

uint8\_t out1\_mode;

uint32\_t out1\_delay;

uint32\_t out1\_pulse\_width;

uint8\_t out1\_invert;

uint8\_t out2\_enable;

uint8\_t out2\_mode;

uint32\_t out2\_delay;

uint32\_t out2\_pulse\_width;

uint8\_t out2\_invert;

uint8\_t reserved[32];

}rf627\_outputs\_params\_t;

#pragma pack(pop)

#pragma pack(push,1)

typedef struct

{

rf627\_general\_params\_t general;

rf627\_sysmon\_params\_t sysmon;

rf627\_rf625compat\_params\_t rf625\_compat;

rf627\_sensor\_params\_t sensor;

rf627\_roi\_params\_t roi;

rf627\_network\_params\_t network;

rf627\_stream\_params\_t stream;

rf627\_image\_processing\_params\_t image\_processing;

rf627\_laser\_params\_t laser;

rf627\_inputs\_params\_t inputs;

rf627\_outputs\_params\_t outputs;

uint8\_t reserved[283];

}rf627\_user\_params\_t;

#pragma pack(pop)

Input modes (presets):

0 - Internal Clock

1 - External Trigger

2 - 1-phase Encoder

3 - 1-phase Encoder w/Zero

4 - 2-phase Encoder

5 - 2-phase Encoder w/Zero

6 - Step/Dir

7 - Ext. Trigger/Int. Clock

8 - Software Request