ITR-3800 Radar API
Read Me

Version: 0.3

Date: 2022-10-19

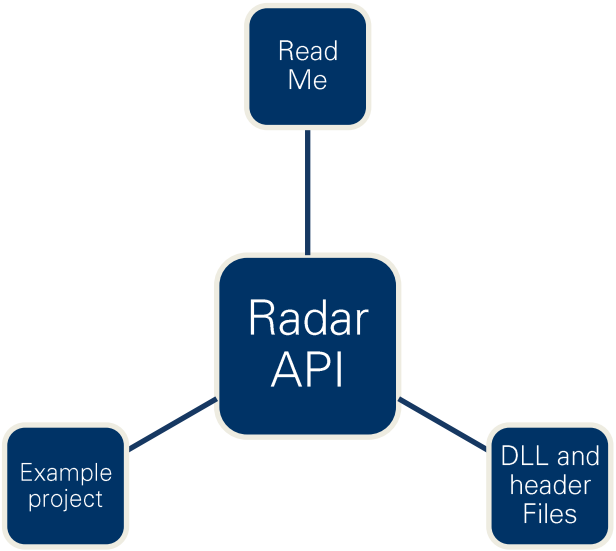
Table of content

History

Version	Date	Change log
0.1	2022-06-10	Initial Release
0.2	2022-06-22	Removed set/get commands for baudrate
0.3	2022-10-19	Removed selfTest

1. What is the Radar API?

The Radar API includes the following components:



- Read me
  Description of the Radar API with explanation of all data types and functions
- DLL and header files
  DLL and header file for easy access to system functionality
- Example project
  Shows how to use the Radar API functionality and gives an easy start

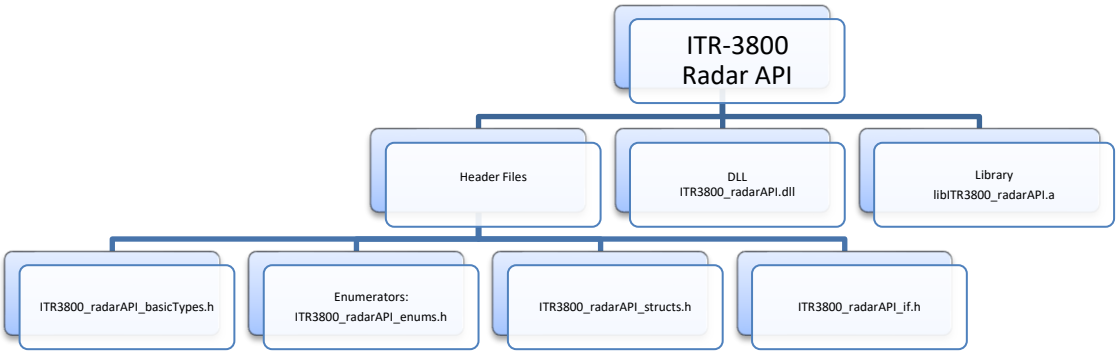
2. General Information

- The Radar API was built in Windows as well as Linux systems
- The Radar API was tested with MS Windows 7/10 32-bit/64-bit
- The Radar API is incompatible with MS Windows XP, Vista and older MS Windows versions
- To guaranty 8 Byte alignment of defined structures, the union type is used
- The Radar API is designed for synchronal use only (blocking application)
- Do not modify delivered Radar API files
- Use the Radar API specific data types defined in ITR3800_radarAPI_basicTypes.h
- There are provided different versions of build versions to cover majority part of possible development environments. If there is any version missing, feel free to contact the InnoSenT Sales Team.

Environment	**32 bit version**	**64 bit version**
Qt Creator	Windows_mingw_x86	Windows_mingw_x64
Visual Studio	Windows_msvc_2017_x86	Windows_msvc_2017_x64
Linux	armv7	aarch64, x64

3. Get the API running in your application

The Radar API consists of the following components:



- ITR3800_radarAPI_basicTypes.h
  Contains all basic type definitions used by the API
- ITR3800_radarAPI_enums.h
  Contains all enumerators used by the API
- ITR3800_radarAPI_structs.h
  Contains all structures used by the API
- ITR3800_radarAPI_if.h
  Contains all necessary definitions, type definitions, structures and commands featured by the API
- ITR3800_radarAPI.dll
  Library file which contains all functions referred in the header file
- libITR3800_radarAPI.a

  *Links against *.dll functions*

4. Programming Example

The delivered software package contains an example project, which shows how to use the ITR-3800 Radar API.

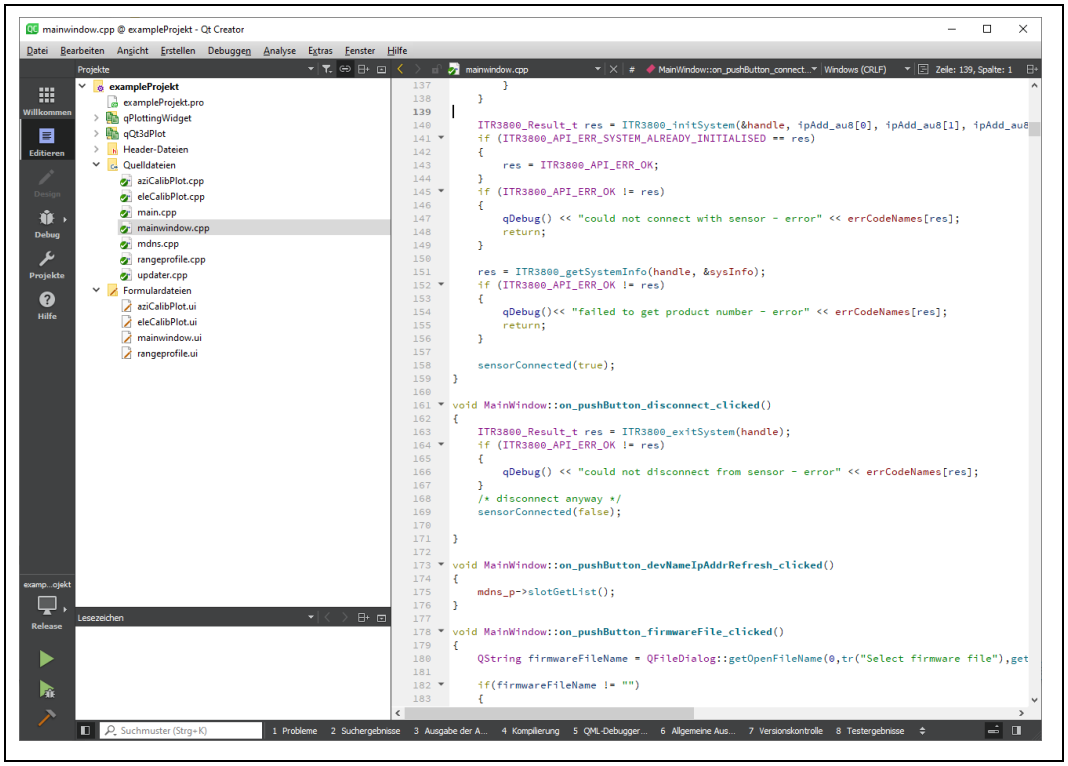


Table 1: Abstract from example project

5. Used types in the API

The API uses specific data types. They are defined in the ITR3800_radarAPI_basicTypes.h, ITR3800_radarAPI_enums.h ITR3800_radarAPI_structs.h file.

Note:	To avoid alignment errors, some enumerators are also defined as union and the union type is used.

5.1. Basic types

The *ITR3800_radarAPI_basicTypes.h* defines the API data types. Please use these types for full compatibility.

Data type	Definition
bool_t	1 bit boolean
float32_t	32 bit floating point
sint8_t / uint8_t	8 bit signed / unsigned integer
sint16_t / uint16_t	16 bit signed / unsigned integer
sint32_t / uint32_t	32 bit signed / unsigned integer

5.2. Structures

There are predefined structures. They are defined in ITR3800_radarAPI_structs.h and make function call easier for the user. Please use these types as well.

5.2.1. APIHandle_t

The API handle is only used internal in the API.

5.2.2. ITR3800_SystemInfo_t

The system info contains the most important information to identify the device with its software version.

```
typedef struct {
   uint32_t productcode;      /* e.g. 3800 */
   uint32_t serialNumber;     /* e.g. 12345678 */
   uint32_t swVersionMajor;   /* e.g. major 1 and minor 3 means 1.003 */
   uint32_t swVersionMinor;
   uint32_t reserverd1;       /* e.g. major 4 and minor 1 means 4.001 */
   uint32_t reserverd2;
   uint32_t reserverd3;
   uint32_t reserverd4;
} ITR3800_SystemInfo_t;
```

5.2.3. ITR3800_Description_t

To identify a device by a user defined attribute a description can be added. The description can for example tell the position of the device like the name of the street with the direction of view (“Mainstreet 1 view to east”).

```
#define ITR3800_MAX_LENGTH_DESCRIPTION_NAME          (64)

/* description */
typedef struct{
   char description[ITR3800_MAX_LENGTH_DESCRIPTION_NAME];
   uint8_t descriptionLength;
}ITR3800_Description_t;
```

5.2.4. **ITR3800_GpsSatellitesInView_t**

Detailed GPS satellite information struct.

```
/* GPS satellites in view */
typedef struct{
   char talkerId_au8[2];
   uint16_t satelliteId_u16;
   uint16_t elevation_deg_u16;
   uint16_t azimuth_deg_u16;
   uint16_t signalStrength_u16;
}ITR3800_GpsSatelliteProperties_t;

typedef struct{
   uint32_t nrOfGpsSatellitesInView;
   ITR3800_GpsSatelliteProperties_t satellites[32];
}ITR3800_GpsSatellitesInView_t;
```

5.2.5. **ITR3800_EventZones_t**

Event zones are defined by a number of points that are connect by straight lines one by one. There can also be set a zone name to distinguish different zones by a name.

Each zone can be defined as a motion or presence zone. The triggers behavior is different for these types:

- Presence zone: triggers are generated in each measurement cycle if the number of static objects changed within the zone
- Motion zone: triggers are generated once for each object when it moves into the zone.

```
#define ITR3800_MAX_LENGTH_EVENTZONE_NAME          (20)
#define ITR3800_MAX_NR_OF_POINTS_PER_EVENT_ZONE    (20)
#define ITR3800_MAX_NR_OF_EVENTZONES               (64)
#define ITR3800_MAX_NR_OF_CONDITIONS               (10)

typedef struct {
   float32_t x;
   float32_t y;
} ITR3800_PointXY_t;

typedef struct {
   bool enable;
   uint8_t outputNumber;
   ITR3800_ConditionClass_t conditionClass;
   ITR3800_ConditionDirection_t direction;
   float32_t velocity_min_kmh;
   float32_t velocity_max_kmh;
   float32_t queuelength_min_m;
   float32_t queuelength_max_m;
   uint16_t eventmessage_delay;
   uint16_t eventmessage_extend;
   float32_t eta_min_s;
   float32_t eta_max_s;
   uint8_t nrOfPedestBikes_min;
   uint8_t nrOfPedestBikes_max;
   uint8_t nrOfCars_min;
   uint8_t nrOfCars_max;
   uint8_t nrOfSmallTrucks_min;
   uint8_t nrOfSmallTrucks_max;
   uint8_t nrOfBigTrucks_min;
   uint8_t nrOfBigTrucks_max;
} ITR3800_Condition_t;

/* event zone struct */
typedef struct {
   bool enable;
   bool eventFlag;
   char zoneName[ITR3800_MAX_LENGTH_EVENTZONE_NAME];
   uint8_t zoneNameLength;
   uint32_t nrOfZonePoints;
   ITR3800_PointXY_t zoneData[ITR3800_MAX_NR_OF_POINTS_PER_EVENT_ZONE];
   ITR3800_Condition_t conditions[ITR3800_MAX_NR_OF_CONDITIONS];
   ITR3800_EventZoneType_t zoneType;   /* EventZone Type */
   uint8_t phaseNumber;
   uint8_t outputNumber;
   uint16_t eventmessage_delay;
   uint16_t eventmessage_extend;
   bool eta_enable;
   ITR3800_PointXY_t etaPoint;
} ITR3800_EventZone_t;

typedef struct {
   ITR3800_EventZone_t eventZones[ITR3800_MAX_NR_OF_EVENTZONES];
} ITR3800_EventZones_t;
```

5.2.6. **ITR3800_IgnoreZones_t**

Ignore zones are defined by a number of points that are connect by straight lines one by one. There can also be set a zone name to distinguish different zones by a name.

No Tracks will be created within the zone.

```
#define ITR3800_MAX_LENGTH_EVENTZONE_NAME          (20)
#define ITR3800_MAX_NR_OF_POINTS_PER_IGNORE_ZONE    (10)
#define ITR3800_MAX_NR_OF_IGNOREZONES              (10)

/* ignore zone struct */
typedef struct {
   bool enable;
   bool eventFlag;
   char zoneName[ITR3800_MAX_LENGTH_EVENTZONE_NAME];
   uint8_t zoneNameLength;
   uint32_t nrOfZonePoints;
   ITR3800_PointXY_t zoneData[ITR3800_MAX_NR_OF_POINTS_PER_IGNORE_ZONE];
   bool ignoreEverything;
} ITR3800_IgnoreZone_t;

typedef struct {
   ITR3800_IgnoreZone_t ignoreZones[ITR3800_MAX_NR_OF_IGNOREZONES];
} ITR3800_IgnoreZones_t;
```

5.2.7. **ITR3800_ObjectList_t**

```
#define ITR3800_MAX_NR_OF_TRACKS                (256)
#define ITR3800_MAX_EVENT_MESSAGE_LENGTH        (128)
#define ITR3800_MAX_NR_OF_EVENT_MESSAGES        (256)

typedef struct ITR3800_eventMessage{
   char c_eventMessage[ITR3800_MAX_EVENT_MESSAGE_LENGTH];
   uint8_t ui8_eventMessageLength;
} ITR3800_EventMessage_t;

typedef struct ITR3800_EventMessageList{
   ITR3800_EventMessage_t eventMessages[ITR3800_MAX_NR_OF_EVENT_MESSAGES];
   uint8_t nrOfMessages;
} ITR3800_EventMessageList_t;

typedef struct {
   uint32_t ui32_objectID;
   uint16_t ui16_ageCount;
   uint16_t ui16_predictionCount;
   uint16_t ui16_staticCount;
   float32_t f32_trackQuality;         /* track quality */
   union ITR3800_TrackClass_u classID;  /* object class of vehicle [] */
   sint16_t si16_motion_eventZoneIndex;   /* object in motion eventzone [index] */
   sint16_t si16_presence_eventZoneIndex; /* object in presence eventzone [index] */
   float32_t f32_positionX_m;          /* position x in cartesian [meter] */
   float32_t f32_positionY_m;          /* position y in cartesian [meter] */
   float32_t f32_velocityX_mps;        /* velocity x in cartesian [meter per second] */
   float32_t f32_velocityY_mps;        /* velocity y in cartesian [meter per second] */
   float32_t f32_velocityInDir_mps;    /* velocity in direction [meter per second] */
   float32_t f32_directionX;           /* normed direction x */
   float32_t f32_directionY;           /* normed direction y */
   float32_t f32_distanceToFront_m;    /* distance To Front [meter] */
   float32_t f32_distanceToBack_m;     /* distance To Back [meter] */
   float32_t f32_length_m;             /* length of object [meter] */
   float32_t f32_width_m;              /* width of object [meter] */
} ITR3800_TrackedObject_t;

typedef struct {
   uint16_t protocolVersion;
   uint8_t reserved_0;
   uint8_t configurationChanged;
   uint32_t systemState;
   union ITR3800_ObjectListError_u error;
   uint32_t timestamp_ms;
   uint16_t frameID;
   uint32_t nrOfTracks;
   ITR3800_TrackedObject_t trackedObjects[ITR3800_MAX_NR_OF_TRACKS];
   ITR3800_EventMessageList_t eventMessages;
   float32_t f32_rainInterferenceLevel;
   float32_t reserved1;
   float32_t reserved2;
   float32_t reserved3;
} ITR3800_ObjectList_t;
```

5.2.8. **ITR3800_ObjectListImperial_t**

```
#define ITR3800_MAX_NR_OF_TRACKS                (256)
#define ITR3800_MAX_EVENT_MESSAGE_LENGTH        (128)
#define ITR3800_MAX_NR_OF_EVENT_MESSAGES        (256)

typedef struct ITR3800_eventMessage{
   char c_eventMessage[ITR3800_MAX_EVENT_MESSAGE_LENGTH];
   uint8_t ui8_eventMessageLength;
} ITR3800_EventMessage_t;

typedef struct ITR3800_EventMessageList{
   ITR3800_EventMessage_t eventMessages[ITR3800_MAX_NR_OF_EVENT_MESSAGES];
   uint8_t nrOfMessages;
} ITR3800_EventMessageList_t;

typedef struct {
   uint32_t ui32_objectID;
   uint16_t ui16_ageCount;
   uint16_t ui16_predictionCount;
   uint16_t ui16_staticCount;
   float32_t f32_trackQuality;        /* track quality */
   union ITR3800_TrackClass_u classID; /* object class of vehicle [] */
   sint16_t si16_motion_eventZoneIndex;   /* object in motion eventzone [index] */
   sint16_t si16_presence_eventZoneIndex; /* object in presence eventzone [index] */
   float32_t f32_positionX_feet;      /* position x in cartesian [feet] */
   float32_t f32_positionY_feet;      /* position y in cartesian [feet] */
   float32_t f32_velocityX_mph;       /* velocity x in cartesian [miles per hour] */
   float32_t f32_velocityY_mph;       /* velocity y in cartesian [miles per hour] */
   float32_t f32_velocityInDir_mph;    /* velocity in direction [miles per hour] */
   float32_t f32_directionX;          /* normed direction x */
   float32_t f32_directionY;          /* normed direction y */
   float32_t f32_distanceToFront_feet;  /* distance To Front [feet] */
   float32_t f32_distanceToBack_feet;   /* distance To Back [feet] */
   float32_t f32_length_feet;         /* length of object [feet] */
   float32_t f32_width_feet;          /* width of object [feet] */
} ITR3800_TrackedObjectImperial_t;

typedef struct {
   uint16_t protocolVersion;
   uint8_t reserved_0;
   uint8_t configurationChanged;
   uint32_t systemState;
   union ITR3800_ObjectListError_u error;
   uint32_t timestamp_ms;
   uint16_t frameID;
   uint32_t nrOfTracks;
   ITR3800_TrackedObject_t trackedObjects[ITR3800_MAX_NR_OF_TRACKS];
   ITR3800_EventMessageList_t eventMessages;
   float32_t f32_rainInterferenceLevel;
   float32_t reserved1;
   float32_t reserved2;
   float32_t reserved3;
} ITR3800_ObjectListImperial_t;
```

5.3. Enumerators

There are predefined enumerators. They are defined in ITR3800_radarAPI_structs.h and make function call easier for the user. Please use these types as well.

5.3.1. **ITR3800_Result_t**

This enumerator helps to decode the error number.

Number	ITR3800_Result_t	Description
0x0000	ITR3800_API_ERR_OK	No error – function executed successful
0x0001	ITR3800_API_ERR_HANDLE_NOT_INITIALISED	System is not connected. Unknown Handle ID. Please use InitSystem Function
0x0002	ITR3800_API_ERR_SYSTEM_ALREADY_INITIALISED	System is already connected
0x0003	ITR3800_API_ERR_SYSTEM_NOT_INITIALISED	System is not initialized
0x0004	ITR3800_API_ERR_CMD_SOCKET_CONNECTION_LOST	Connection to System lost
0x0005	ITR3800_API_ERR_CMD_NOT_ACCEPTED	Command rejected
0x0006	ITR3800_API_ERR_CMD_NOT_WRITEABLE	
0x0007	ITR3800_API_ERR_CMD_DATA_CORRUPTED	
0x0008	ITR3800_API_ERR_CMD_PARAMETER	Parameter not accepted
0x0009	ITR3800_API_ERR_CREATE_HANDLE	
0x000A	ITR3800_API_ERR_OBJECT_SOCKET_NO_CONNECTION_AVAILABLE	
0x000B	ITR3800_API_ERR_OBJECT_SOCKET_CONNECTION_LOST	
0x000C	ITR3800_API_ERR_SYSTEM_FILENAME_NOT_VALID	
0x000D	ITR3800_API_ERR_CMD_TOO_MUCH_DATA_TO_READ	
0x000E	ITR3800_API_ERR_CMD_STILL_DATA_AVAILABLE	
0x000F	ITR3800_API_ERR_NULL_POINTER	
0x0010	ITR3800_API_ERR_OBJECT_NO_VALID_SOCKET	
0x0011	ITR3800_API_ERR_CMD_UNKNOWN	
0x0012	ITR3800_API_ERR_FUNCTION_DEPRECATED	
0x0013	ITR3800_API_ERR_CONNECTION_CLOSED	
0x0014	ITR3800_API_ERR_CONNECTION_RESET	
0x0015	ITR3800_API_ERR_COMMUNICATION_TIMEOUT	

0x0016	ITR3800_API_ERR_COMMUNICATION_ERROR	
0x0017	ITR3800_API_ERR_CONNECTION_LOST	
0x0018	ITR3800_API_ERR_ISYS_NOT_FOUND	
0x0019	ITR3800_API_ERR_IP_MASK	
0x001A	ITR3800_API_ERR_CMD_TOO_LESS_DATA_TO_READ	
0x001B	ITR3800_API_ERR_MUTEX_ERROR	
0x001C	ITR3800_API_ERR_OBJECT_LIST_MUTEX	
0x001D	ITR3800_API_ERR_OBJECT_THREAD_INIT	
0x001E	ITR3800_API_ERR_OBJECT_LIST_TIMEOUT	
0x001F	ITR3800_API_ERR_OBJECT_LIST_UNKNOWN	
0x0020	ITR3800_API_ERR_OBJECT_LIST_DISCONNECTED_BY_SYSTEM	
0x0021	ITR3800_API_ERR_OBJECT_LIST_DISCONNECTED_BY_DLL	
0x0022	ITR3800_API_ERR_OBJECT_LIST_RESET	
0x0023	ITR3800_API_ERR_OBJECT_LIST_SOCKET_NULL	
0x0024	ITR3800_API_ERR_OBJECT_LIST_SET_MUTEX	
0x0025	ITR3800_API_ERR_OBJECT_LIST_NO_VALID_MUTEX	
0x0026	ITR3800_API_ERR_OBJECT_LIST_MUTEX_TIMED_OUT	
0x0100	ITR3800_API_ERR_COMMBUFFER_RAWDATA_SOCKET_CONNECTION_LOST	
0x0101	ITR3800_API_ERR_COMMBUFFER_RAWDATA_TIMEOUT	
0x0102	ITR3800_API_ERR_COMMBUFFER_RAWDATA_MUTEX	
0x0103	ITR3800_API_ERR_COMMBUFFER_RAWDATA_MUTEX_TIMEOUT	
0x0104	ITR3800_API_ERR_COMMBUFFER_RAWDATA_INIT	
0x0200	ITR3800_API_ERR_NO_CONNECTION_AVAILABLE	
0x0201	ITR3800_API_ERR_NO_VALID_SOCKET	
0x0202	ITR3800_API_ERR_ISYS_SW_VERSION_NOT_SUPPORTED	
0x0203	ITR3800_API_ERR_RAWDATA_NO_FIRST_VALID_FRAME	
0x0204	ITR3800_API_ERR_OBJECT_LIST_SIZE_METRIC_IMPERIAL	
0x0205	ITR3800_API_ERR_FIRMWARE_VERSION_INCOMPATIBLE	API doesn´t match. Please perform firmware update
0x0206	ITR3800_API_ERR_FUNCTION_NOT_IMPLEMENTED	

5.3.2. **ITR3800_TimeZone_t**

This enumerator helps to set the systems time zone.

Number	ITR3800_TimeZone_t	Description
/* UTC-10 */		
0	ITR3800_API_TIMEZONE_ALEUTIAN	US/Aleutian
1	ITR3800_API_TIMEZONE_HAWAII	US/Hawaii
/* UTC-9:30 */		
2	ITR3800_API_TIMEZONE_MARQUESAS	Pacific/Marquesas
/* UTC-9:00 */		
3	ITR3800_API_TIMEZONE_ALASKA	US/Alaska
/* UTC-8:00 */		
4	ITR3800_API_TIMEZONE_PACIFIC_TIME	US/Pacific
/* UTC-7:00 */		
5	ITR3800_API_TIMEZONE_ARIZONA	US/Arizona
6	ITR3800_API_TIMEZONE_MOUNTAIN_TIME	US/Mountain
7	ITR3800_API_TIMEZONE_LA_PAZ	America/La Paz
/* UTC-6:00 */		
8	ITR3800_API_TIMEZONE_CENTRAL_TIME	US/Central
9	ITR3800_API_TIMEZONE_MEXICO_CITY	America/Mexico City
10	ITR3800_API_TIMEZONE_EASTER	Pacific/Easter
/* UTC-5:00 */		
11	ITR3800_API_TIMEZONE_BOGOTA_LIMA_QUITO_RIO_BRANCO	America/Bogota
	ITR3800_API_TIMEZONE_EASTERN_TIME	US/Eastern
/* UTC-4:00 */		
12	ITR3800_API_TIMEZONE_ASUNCION	America/Asuncion
13	ITR3800_API_TIMEZONE_CARACAS	America/Caracas
14	ITR3800_API_TIMEZONE_CUIABA	America/Cuiaba
15	ITR3800_API_TIMEZONE_SANTAGIO	America/Santiago
/* UTC-3:00 */		
16	ITR3800_API_TIMEZONE_BUENOS_AIRES	America/Argentina/Buenos Aires
17	ITR3800_API_TIMEZONE_CAYENNE	America/Cayenne
18	ITR3800_API_TIMEZONE_MONTEVIDEO	America/Montevideo

19	ITR3800_API_TIMEZONE_EL_SALVADOR	America/El Salvador
20	ITR3800_API_TIMEZONE_MIQUELON	America/Miquelon
/* UTC-0:00 GREENWICH-TIME */		
21	ITR3800_API_TIMEZONE_CASABLANCA	Africa/Casablanca
22	ITR3800_API_TIMEZONE_DUBLIN	Europe/Dublin
23	ITR3800_API_TIMEZONE_MONROVIA	Africa/Monrovia
/* UTC+1:00 */		
24	ITR3800_API_TIMEZONE_AMSTERDAM_BERLIN_ROME_STOCKHOLM_VIENNA	Europe/Amsterdam
25	ITR3800_API_TIMEZONE_BELGRADE	Europe/Belgrade
26	ITR3800_API_TIMEZONE_BRUSSELS	Europe/Brussels
27	ITR3800_API_TIMEZONE_SARAJEVO	Europe/Sarajevo
28	ITR3800_API_TIMEZONE_WINDHOEK	Africa/Windhoek
/* UTC+2:00 */		
29	ITR3800_API_TIMEZONE_AMMAM	Asia/Amman
30	ITR3800_API_TIMEZONE_ATHENS	Europe/Athens
31	ITR3800_API_TIMEZONE_BEIRUT	Asia/Beirut
32	ITR3800_API_TIMEZONE_CHISINAU	Europe/Chisinau
33	ITR3800_API_TIMEZONE_DAMASCUS	Asia/Damascus
34	ITR3800_API_TIMEZONE_GAZA	Asia/Gaza
35	ITR3800_API_TIMEZONE_HARARE	Africa/Harare
36	ITR3800_API_TIMEZONE_HELSINKI	Europe/Helsinki
37	ITR3800_API_TIMEZONE_JERUSALEM	Asia/Jerusalem
38	ITR3800_API_TIMEZONE_KALININGRAD	Europe/Kaliningrad
39	ITR3800_API_TIMEZONE_CAIRO	Africa/Cairo
40	ITR3800_API_TIMEZONE_TRIPOLIS	Africa/Tripoli
/* UTC+3:00 */		
41	ITR3800_API_TIMEZONE_ISTANBUL	Europe/Istanbul
42	ITR3800_API_TIMEZONE_KUWAIT	Asia/Kuwait
43	ITR3800_API_TIMEZONE_MINSK	Europe/Minsk
44	ITR3800_API_TIMEZONE_MOSCOW	Europe/Moscow
45	ITR3800_API_TIMEZONE_NAIROBI	Africa/Nairobi
/* UTC+4:00 */		

46	ITR3800_API_TIMEZONE_BAKU	Asia/Baku
47	ITR3800_API_TIMEZONE_SAMARA	Europe/Samara
/* UTC+4:30 */		
48	ITR3800_API_TIMEZONE_KABUL	Asia/Kabul
/* UTC+5:00 */		
49	ITR3800_API_TIMEZONE_ASHGABAT	Asia/Ashgabat
50	ITR3800_API_TIMEZONE_KARACHI	Asia/Karachi
/* UTC+5:30 */		
51	ITR3800_API_TIMEZONE_KOLKATA	Asia/Kolkata
/* UTC+5:45 */		
52	ITR3800_API_TIMEZONE_KATMANDU	Asia/Katmandu
/* UTC+6:00 */		
53	ITR3800_API_TIMEZONE_OMSK	Asia/Omsk
/* UTC+7:00 */		
54	ITR3800_API_TIMEZONE_BANGKOK_HANOI_JAKARTA	Asia/Bangkok
55	ITR3800_API_TIMEZONE_HOVD	Asia/Hovd
56	ITR3800_API_TIMEZONE_KRASNOYARSK	Asia/Krasnoyarsk
/* UTC+8:00 */		
57	ITR3800_API_TIMEZONE_IRKUTSK	Asia/Irkutsk
58	ITR3800_API_TIMEZONE_KUALA_LUMPUR	Asia/Kuala Lumpur
59	ITR3800_API_TIMEZONE_HONG_KONG	Asia/Hong Kong
60	ITR3800_API_TIMEZONE_PERTH	Australia/Perth
61	ITR3800_API_TIMEZONE_ULAN_BATOR	Asia/Ulan Bator
62	ITR3800_API_TIMEZONE_TAIPEI	Asia/Taipei
/* UTC+8:30 */		
63	ITR3800_API_TIMEZONE_PYONGYANG	Asia/Pyongyang
/* UTC+8:45 */		
64	ITR3800_API_TIMEZONE_EUCLA	Australia/Eucla
/* UTC+9:00 */		
65	ITR3800_API_TIMEZONE_SEOUL	Asia/Seoul
/* UTC+9:30 */		
66	ITR3800_API_TIMEZONE_ADELAIDE	Australia/Adelaide

67	ITR3800_API_TIMEZONE_DARWIN	Australia/Darwin
/* UTC+10:00 */		
68	ITR3800_API_TIMEZONE_BRISBANE	Australia/Brisbane
69	ITR3800_API_TIMEZONE_CANBERRA	Australia/Canberra
70	ITR3800_API_TIMEZONE_GUAM	Pacific/Guam
71	ITR3800_API_TIMEZONE_HOBART	Australia/Hobart
72	ITR3800_API_TIMEZONE_VLADIVOSTOK	Asia/Vladivostok
/* UTC+10:30 */		
73	ITR3800_API_TIMEZONE_LORD_HOWE	Australia/Lord Howe
/* UTC+11:00 */		
74	ITR3800_API_TIMEZONE_MAGADAN	Asia/Magadan
75	ITR3800_API_TIMEZONE_NORFOLK	Pacific/Norfolk
/* UTC+12:00 */		
76	ITR3800_API_TIMEZONE_AUCKLAND	Pacific/Auckland
77	ITR3800_API_TIMEZONE_FIJI	Pacific/Fiji
/* UTC+12:45 */		
78	ITR3800_API_TIMEZONE_CHATHAM	Pacific/Chatham
/* UTC+13:00 */		
79	ITR3800_API_TIMEZONE_SAMOA	Pacific/Samoa
/* UTC+14:00 */		
80	ITR3800_API_TIMEZONE_KIRIMATI	Pacific/Kiritimati
/* UTC MAX */		
81	ITR3800_API_TIMEZONE_MAX	

5.3.3. ITR3800_FrequencyChannel_t

Number	ITR3800_FrequencyChannel_t	Description
1	ITR3800_FREQUENCY_CHANNEL_1	Frequency Channel 1
2	ITR3800_FREQUENCY_CHANNEL_2	Frequency Channel 2
3	ITR3800_FREQUENCY_CHANNEL_3	Frequency Channel 3
4	ITR3800_FREQUENCY_CHANNEL_4	Frequency Channel 4

5.3.4. ITR3800_EventZoneType_t

Number	ITR3800_EventZoneType_t	Description
0x00	ITR3800_EVENTZONE_MOTION	Type for Event-Zone
0x01	ITR3800_EVENTZONE_PRESENCE	Type for Presence-Zone
0x02	ITR3800_EVENTZONE_SIDEWALK	Type for Sidewalk-Zone

5.3.5. ITR3800_ConditionClass_t

Number	ITR3800_ConditionClass_t	Description
0x00	ITR3800_CONDITIONCLASS_ALL	All classes
0x01	ITR3800_CONDITIONCLASS_CAR	Car only
0x02	ITR3800_CONDITIONCLASS_BIKE_PEDEST	Bike and Pedestrian only
0x03	ITR3800_CONDITIONCLASS_SMALL_TRUCK	small Track only
0x04	ITR3800_CONDITIONCLASS_BIG_TRUCK	big Truck only
0x05	ITR3800_CONDITIONCLASS_CAR_BIG_TRUCK	Car and big Truck only
0x06	ITR3800_CONDITIONCLASS_CAR_SMALL_TRUCK	Car and small Truck only
0x07	ITR3800_CONDITIONCLASS_CAR_SMALL_BIG_TRUCK	Car and small Truck and big Truck Only

5.3.6. ITR3800_ConditionDirection_t

Number	ITR3800_DetectorDirection_t	Description
0x00	ITR3800_CONDITIONDIRECTION_BOTH	Both directions
0x01	ITR3800_CONDITIONDIRECTION_APPROACHING	Approaching only
0x02	ITR3800_CONDITIONDIRECTION_RECEDING	Receding only

5.3.7. **ITR3800_UnitType_t**

Number	ITR3800_UnitType_t	Description
0x00	ITR3800_UNIT_METER	Meter
0x01	ITR3800_UNIT_KMH	Kilometer per hour
0x02	ITR3800_UNIT_FEET	Feet
0x03	ITR3800_UNIT_MPH	Miles per hour

5.3.8. **ITR3800_Udp_Mode_t**

Number	ITR3800_Udp_Mode_t	Description
0x00	ITR3800_UDP_MODE_BROADCAST	Send UDP Message to network broadcast. Every device in the network will receive the messages.
0x01	ITR3800_UDP_MODE_MULTICAST	Send UDP Message to network broadcast. Every device in the network will receive the messages after joining the multicast group.
0x02	ITR3800_UDP_MODE_IP	Send UDP Message to a specific ip address. Only the specific ip will receive udp messages

5.3.9. **ITR3800_TrackClass_u**

```
union ITR3800_TrackClass_u {
   ITR3800_TrackClass_t ITR3800_TrackClass;
   uint32_t dummy;
};
```

5.3.10. **ITR3800_TrackClass_t**

Number	ITR3800_TrackClass_t	Description
2u	ITR3800_API_TRACK_CLASS_OTHERS	Class for ‘others’
10u	ITR3800_API_TRACK_CLASS_PEDESTRIAN	Class for Pedestrians
12u	ITR3800_API_TRACK_CLASS_BICYCLE	Class for Bicycles
30u	ITR3800_API_TRACK_CLASS_CAR	Class for Cars
60u	ITR3800_API_TRACK_CLASS_SMALL_TRUCK	Class for small Trucks
70u	ITR3800_API_TRACK_CLASS_BIG_TRUCK	Class for big Trucks

5.3.11. ITR3800_ObjectListError_u

```
union ITR3800_ObjectListError_u {
  ITR3800_ObjectListError_t ITR3800_ObjectListError;
  uint32_t dummy;
};
```

5.3.12. ITR3800_ObjectListError_t

Number	ITR3800_ObjectListError_t	Description
0x00	ITR3800_API_OBJECT_LIST_OK	No Error
0x01	ITR3800_API_OBJECT_LIST_FULL	Object list full
0x02	ITR3800_API_OBJECT_LIST_REFRESHED	
0x03	ITR3800_API_OBJECT_LIST_ALREADY_REQUESTED	No new Object list available
0x04	ITR3800_API_OBJECT_LIST_NOT_RUNNING	Object list Output disabled
0x05	ITR3800_API_OBJECT_LIST_FIRMWARE_UPDATE_RUNNING	Firmware is being updated
0x06	ITR3800_API_OBJECT_LIST_LISTENING	
0x07	ITR3800_API_OBJECT_LIST_COMMUNICATION_TIMEOUT	

5.3.13. ITR3800_NetworkSetting_t;

Number	ITR3800_NetworkSetting_t	Description
0	ITR3800_API_NETWORK_DHCP	
1	ITR3800_API_NETWORK_STATICIP	

5.3.14. ITR3800_SaveLocation_t;

Number	ITR3800_SaveLocation_t	Description
0	ITR3800_API_LOCATION_RAM	
1	ITR3800_API_LOCATION_EEPROM	

5.3.15. ITR3800_TrafficDirection_t

Number	ITR3800_TrafficDirection_t	Description
0x00	ITR3800_API_DIRECTION_APPROACHING	
0x01	ITR3800_API_DIRECTION_RECEDING	
0x02	ITR3800_API_DIRECTION_BOTH	

5.3.16. **ITR3800_TargetListError_u**

```
union ITR3800_TargetListError_u {
   ITR3800_TargetListError_t ITR3800_TargetListError;
   uint32_t dummy;
};
```

5.3.17. **ITR3800_TargetListError_t**

Number	ITR3800_TargetListError_t	Description
0x00	ITR3800_API_TARGET_LIST_OK	No Error
0x01	ITR3800_API_ TARGET _LIST_FULL	Object list full
0x02	ITR3800_API_ TARGET _LIST_REFRESHED	
0x03	ITR3800_API_ TARGET _LIST_ALREADY_REQUESTED	No new Object list available
0x04	ITR3800_API_ TARGET _LIST_NOT_RUNNING	Object list Output disabled
0x05	ITR3800_API_ TARGET _LIST_FIRMWARE_UPDATE_RUNNING	Firmware is being updated
0x06	ITR3800_API_ TARGET _LIST_LISTENING	
0x07	ITR3800_API_ TARGET _LIST_COMMUNICATION_TIMEOUT	

6. API function description

This chapter describes all available functions including calling examples. Please have also a look to the example project.

6.1. System functions

6.1.1. **getApiVersion**

Read the current RadarAPI Software Version.

```
ITR3800_Result_t res;
float32_t version;

/************** CONNECT DEVICE **************/
res = ITR3800_getApiVersion(&version);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_getApiVersion failed\n");
   return -1;
}
```

6.1.2. **initSystem**

This function establishes a connection to the device with the IP address.

Please create one APIHandle_t for each device.

Create an empty APIHandle_t for each connection. This is a unique identifier to communicate with this system. This function initiates the connection and initializes the handle.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** CONNECT DEVICE **************/
res = ITR3800_initSystem(&handle,192,168,178,253);
if((res != ITR3800_API_ERR_OK) && (res != ITR3800_API_ERR_SYSTEM_ALREADY_INITIALISED))
{
   printf("error – ITR3800_initSystem failed\n");
   return -1;
}
```

6.1.3. **exitSystem**

To close (disconnect) an existing connection.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** EXIT DEVICE **************/

res = ITR3800_exitSystem(handle);
```

6.1.4. **restartSystem**

Performs a system restart. Take in mind that this takes up to 90 second.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** RESTART DEVICE **************/
res = ITR3800_restartSystem(handle);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_restartSystem failed\n");
   return -1;
}
```

6.1.5. **getSystemInfo**

Reads the system information to identify the device and its versions.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** GET SYSTEM INFORMATION **************/
ITR3800_SystemInfo_t info;
res = ITR3800_getSystemInfo(handle,& info);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_getSystemInfo failed\n");
}
else
{
   printf("device:         ITR-%d\n", info.productcode);
   printf("serial:         %d\n", info.serialNumber);
   printf("software version: %d.%.3d\n", info.swVersionMajor, info.swVersionMinor);
}
```

6.1.6. **set/getDescription**

Use the description to give the devices unique identifiers which refers for example to installation place in field like street names of the crossings and the location with the cardinal points.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** SET DESCRIPTION **************/
ITR3800_Description_t name;
snprintf(name.description,sizeof(name.description),"this is my device");
name.descriptionLength = strlen(name.description);

res = ITR3800_setDescription(handle,name);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setDescription failed\n");
}

/************** GET DESCRIPTION **************/
ITR3800_Description_t description;
res = ITR3800_getDescription(handle,&description);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getDescription failed\n");
}
else
{
```

```
    printf("description: %s\n", description.description);
}
```

6.2. Configuration functions

6.2.1. **set/getUdpSettings**

Use this function to define where the udp event messages will be sent. Default is the network broadcast.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** SET UDP SETTINGS **************/
res = ITR3800_setUdpSettings(handle,ITR3800_UDP_MODE_BROADCAST,62200,0,0,0,0);
if(res != ITR3800_API_ERR_OK)
{
    printf("error – ITR3800_setUdpSettings failed\n");
}
#endif

/************** GET UDP SETTINGS **************/
ITR3800_Udp_Mode_t mode;
uint16_t port;
uint8_t ipPart1;
uint8_t ipPart2;
uint8_t ipPart3;
uint8_t ipPart4;

res = ITR3800_getUdpSettings(handle,&mode,&port,&ipPart1,&ipPart2,&ipPart3,&ipPart4);
if(res != ITR3800_API_ERR_OK)
{
    printf("error - ITR3800_getUdpSettings failed\n");
}
else
{
    printf("mode: <ITR3800_Udp_Mode_t>(%d)\n ", mode);
    printf("port: %d\n ", port);
}
```

6.2.2. **setStaticIP**

Set device to a static IP address and disable requesting IP address from DHCP server.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** SET STATIC IP ADDRESS **************/
res = ITR3800_setStaticIP(handle,192,168,178,253,255,255,255,0,192,168,178,1);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setStaticIP failed\n");
}
```

6.2.3. **setDhcpIP**

Set device to request IP address by DHCP server. If no DHCP server available, the device sets the default IP address 192.168.178.253.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** SET DHCP **************/
res = ITR3800_setDhcpIP(handle);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setDhcpIP failed\n");
}
```

6.2.4. **set/getNetworkHostname**

Set/Get custom network hostname. Please note that the system must be restarted and the new hostname can be reached in the network after a while.

```
ITR3800_Result_t res;
APIHandle_t handle;
char tmpChar[255];

#ifndef READ_ONLY_FUNCTIONS
/************** SET HOSTNAME **************/
memcpy(&tmpChar[0],"Hostname123",11);

res = ITR3800_setNetworkHostname(handle, &tmpChar[0], 11);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setNetworkHostname failed\n");
}
#endif

/************** GET HOSTNAME **************/
uint8_t length;
res = ITR3800_getNetworkHostname(handle,&tmpChar[0], &length);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getNetworkHostname failed\n");
}
```

6.2.5. **set/getInstallationHeight**

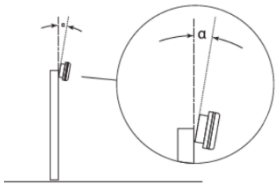
Set the installation height of the device in meters above ground. There is also a function for imperial available if you want to set the height in feet. Use the imperial function in the same way than the metric.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** SET INSTALLATION HEIGHT **************/
res = ITR3800_setInstallationHeight(handle,6.0f);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setInstallationHeight failed\n");
}
#endif

/************** GET INSTALLATION HEIGHT **************/
float32_t height;
res = ITR3800_getInstallationHeight(handle,&height);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getInstallationHeight failed\n");
}
else
{
   printf("installation height:          %.2f m\n", height);
}
```

6.2.6. **set/getElevationAngle**



Set the elevation angle of the device in degree. The angle is related between the device and the street. If the street has an inclination, please regard this for the elevation angle.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
   /************** SET ELEVATION ANGLE **************/
   res = ITR3800_setElevationAngle(handle,-7.0f);
   if(res != ITR3800_API_ERR_OK)
   {
      printf("error – ITR3800_setElevationAngle failed\n");
   }
#endif

   /************** GET ELEVATION ANGLE **************/
   float32_t angle;
   res = ITR3800_getElevationAngle(handle,&angle);
   if(res != ITR3800_API_ERR_OK)
   {
      printf("error - ITR3800_getElevationAngle failed\n");
   }
   else
   {
      printf("elevation angle: %.2f deg\n", angle);
   }
```

6.2.7. **getInclinationAngles**

The device has an internal inclination angle measurement sensor. This sensor measures the pitch and roll angle of the device. The pitch angle corresponds to the elevation angle.

In ideal mounting the roll angle is 0° ± 1°.

The pitch angle can be used to figure out the elevation angle which has to be set. Just read the pitch angle and add the inclination of the road.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** GET INCLINATION SENSOR **************/
float32_t pitch, roll;
res = ITR3800_getInclinationAngles(handle,&pitch, &roll);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_getInclinationAngles failed\n");
}
else
{
   printf("inclination - pitch: %.1f deg\n", pitch);
   printf("inclination - roll:  %.1f deg\n", roll);
}
```

6.2.8. **set/getDateTime**

The date can be set manual, if the network does not provide a time server.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
   /************** SET DATE AND TIME **************/
res = ITR3800_setDateTime(handle, 2019, 9, 21, 7, 8, 9,
                  ITR3800_API_TIMEZONE_AMSTERDAM_BERLIN_ROME_STOCKHOLM_VIENNA);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setDateTime failed\n");
}
#endif
/************** GET DATE AND TIME **************/
uint16_t year, month, day, hour, minute, second;
ITR3800_TimeZone_t timeZone;
res = ITR3800_getDateTime(handle, &year, &month, &day, &hour, &minute, &second,
                  &timeZone);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getDateTime failed\n");
}
else
{
   printf("date:    %d-%d-%d\n", year, month, day);
   printf("time:    %d-%d-%d\n", hour, minute, second);
   printf("timezone: < ITR3800_TimeZone_t>(%d)\n", timeZone);
}
```

6.2.9. **set/getGpsDateTimeSyncEnable**

Usually the device gets the time information by a time server with the network. If there is no time server available the time can also be synchronized by the internal GPS sensor. Just enable the synchronization and take in mind that you have to place the device outdoor for proper GPS reception.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** SET GPS SYNCHRONIZATION ENABLE **************/
res = ITR3800_setGpsDateTimeSyncEnable(handle,false);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setGpsDateTimeSyncEnable failed\n");
}
#endif

/************** GET GPS SYNCHRONIZATION ENABLE **************/
bool_t gpsEnable;
res = ITR3800_getGpsDateTimeSyncEnable(handle,&gpsEnable);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getGpsDateTimeSyncEnable failed\n");
}
else
{
   printf("gps synchronization enable: %d\n", gpsEnable);
}
```

6.2.10. **set/getNtpDateTimeSyncEnable**

Option to enable/disable date/time synchronization over NTP (network time protocol). After disable, system will not search for pool.ntp.org or other timeserver.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** SET NTP SYNCHRONIZATION ENABLE **************/
res = ITR3800_setNtpDateTimeSyncEnable(handle,false);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setNtpDateTimeSyncEnable failed\n");
}
#endif

/************** GET NTP SYNCHRONIZATION ENABLE **************/
bool_t ntpEnable;
res = ITR3800_getNtpDateTimeSyncEnable(handle,&ntpEnable);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getNtpDateTimeSyncEnable failed\n");
}
else
{
   printf("ntp synchronization enable: %d\n", ntpEnable);
}
```

6.2.11. **getGpsCoordinates**

This function reads the GPS location of the device.
Take in mind that you have to place the device outdoor for proper GPS reception.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** GET GPS LOCATION **************/
float32_t lat, lng;
res = ITR3800_getGpsCoordinates(handle,&lat, &lng);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_getGpsCoordinates failed\n");
}
else if((lat == 0.0f) && (lng == 0.0f))
{
   printf("gpslocation: no signal\n");
}
else
{
   printf("gpslocation lateral:   %.5f deg\n", lat);
   printf("gpslocation longitude: %.5f deg\n", lng);
}
```

6.2.12. **set/getFrequencyChannel**

There are four frequency channels provided. This features up to four devices at the same location with no interference. Just set different channels for the devices.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** SET FREQUENCY CHANNEL **************/
res = ITR3800_setFrequencyChannel(handle, ITR3800_FREQUENCY_CHANNEL_1);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setFrequencyChannel failed\n");
}
#endif

/************** GET FREQUENCY CHANNEL **************/
ITR3800_FrequencyChannel_t channel;
res = ITR3800_getFrequencyChannel(handle, &channel);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getFrequencyChannel failed\n");
}
else
{
   printf("frequency channel:              %d\n", channel);
}
```

6.2.13. **set/getRainInterferenceLevelThreshold**

The device measures the rain interference level and sends an information if this value is above the threshold. This function adapts this threshold to a user defined value.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** SET RAIN INTERFERENCE LEVEL THRESHOLD **************/
res = ITR3800_setRainInterferenceLevelThreshold(handle,30.0f);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setRainInterferenceLevelThreshold failed\n");
}
#endif

/************** GET RAIN INTERFERENCE LEVEL THRESHOLD **************/
float32_t threshold_dB;
res = ITR3800_getRainInterferenceLevelThreshold(handle, &threshold_dB);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getRainInterferenceLevelThreshold failed\n");
}
else
{
   printf("rain interference level threshold: %.2f dB\n", threshold_dB);
}
```

6.2.14. **set/getSimulation**

Enable this simulation mode to get a simulated objectlist.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** ENABLE SIMULATION MODE **************/
res = ITR3800_setSimulation(handle,1);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setSimulation failed\n");
}
#endif

/************** GET SIMULATION MODE STATUS **************/
float32_t enable;
res = ITR3800_getSimulation(handle, &enable);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getSimulation failed\n");
}
else
{
   printf("simulation mode: %d\n", enable);
}
```

6.2.15. **uploadPlaybackFile**

Upload a playback file (*.bin) recorded with the traffic manager. Max filesize: 3GB.
This function creates a new thread uploading the file. To monitor the upload, use the "getUploadPlaybackFileStatus" function.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** UPLOAD PLAYBACK FILE TO ITR-3800 SYSTEM (max. 3GB) **************/
res = ITR3800_uploadPlaybackFile(handle, "C:/temp/record.bin");
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_uploadPlaybackFile failed\n");
}
else
{
   printf("upload started\n");
}
```

6.2.16. **getUploadPlaybackFileStatus**

Get the current file upload progress in percent.

```
ITR3800_Result_t res;
ITR3800_Result_t res_upload;
APIHandle_t handle;
uint8_t status_percent;

/************** GET UPLOAD PLAYBACK FILE STATUS **************/
res_upload = ITR3800_API_ERR_OK;
res = ITR3800_API_ERR_OK;

while (res_upload == ITR3800_API_ERR_OK && status_percent != 100){
   res = ITR3800_getUploadPlaybackFileStatus(handle,&status_percent,&res_upload);
   if (res == ITR3800_API_ERR_OK && res_upload == ITR3800_API_ERR_OK){
      printf("upload progess in percent: %d\n", status_percent);
   }
   else{
      printf("error – ITR3800_uploadPlaybackFile failed\n");
      return;
   }

   msleep(250);
}
```

6.2.17. **set/getPlayback**

Enable plackback from file. Must be uploaded first with "uploadPlaybackFile" function.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** ENABLE PLAYBACK MODE **************/
res = ITR3800_setPlayback(handle,1);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setPlayback failed\n");
}
#endif

/************** GET PLAYBACK MODE STATUS **************/
float32_t enable;
res = ITR3800_getPlayback(handle, &enable);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getPlayback failed\n");
}
```

```
else
{
   printf("Playback mode: %d\n", enable);
}
```

6.2.18. **set/getUnitTypes**

Set/Get Unit type for event messages.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************* SET UNIT TYPE **************/
res = ITR3800_setUnitTypes(handle, ITR3800_UNIT_METER, ITR3800_UNIT_KMH);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setUnitTypes failed\n");
}
#endif

/************** GET SIMULATION MODE STATUS **************/
ITR3800_UnitType_t distanceUnit;
ITR3800_UnitType_t velocityUnit;

res = ITR3800_ getUnitTypes(handle, &distanceUnit, &velocityUnit);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getUnitTypes failed\n");
}
```

6.2.19. **set/getStaticTargetsEnable**

Enable/Disable true static target detection (default: enabled). Disable this option, if you have a lot of false detections / objects in presence zones.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** SET STATIC TARGETS ENABLE **************/
res = ITR3800_setStaticTargetsEnable(handle, 1);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_ setStaticTargetsEnable failed\n");
}
#endif

/************** GET STATIC TARGETS ENABLE **************/
uint8_t enable;

res = ITR3800_getStaticTargetsEnable (handle, &enable);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getStaticTargetsEnable failed\n");
}
```

6.2.20. **getGpsSatellitesInView**

Read detailed GPS satellite information

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** GET GPS SATELLITE INFORMATION **************/
ITR3800_GpsSatellitesInView_t gpsSatellitesInView;
```

```
res = ITR3800_getGpsSatellitesInView (handle, &gpsSatellitesInView);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getGpsSatellitesInView failed\n");
}
```

6.2.21. **set/getHighwayMode**

Enable this to get an optimized performance in highway applications.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** ENABLE HIGHWAY MODE **************/
res = ITR3800_setHighwayMode(handle,1);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setHighwayMode failed\n");
}
#endif

/************** GET HIGHWAY MODE STATUS **************/
float32_t enable;
res = ITR3800_getHighwayMode(handle, &enable);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getHighwayMode failed\n");
}
else
{
   printf("highway mode: %d\n", enable);
}
```

6.2.22. **set/getMaxStopTimeObject**

Static objects will be held within presence zones for 300 seconds. The time starts when they come to a halt. The time to hold static objects is adjustable with this function.

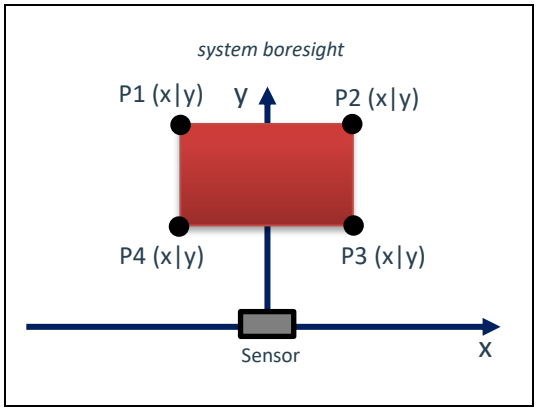
```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** SET MAX STOP TIME FOR STATIC OBJECTS **************/
res = ITR3800_setMaxStopTimeObject(handle,300);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setMaxStopTimeObject failed\n");
}
#endif

/************** GET MAX STOP TIME FOR STATIC OBJECTS **************/
uint16_t time_s;
res = ITR3800_getMaxStopTimeObject(handle, &time_s);
if(res != ITR3800_API_ERR_OK)
{
   printf("error - ITR3800_getMaxStopTimeObject failed\n");
}
else
{
   printf("max stop time for static objects:  %d ms\n", time_s);
}
```

6.2.23. **set/getEventZones**

This function sets the event zones. Each zone has 4 … 20 corner points. The sketch shows four points.



The event zones are necessary to generate triggers in case of events like. For example are generated triggers for each object which moves into a motion zone. This can be used for traffic counting.

After setting a zone the device performs an internal calculation which takes a short time. Please wait 1 second after a set function before the next function call.

There are also an imperial value function available. Just put in the zone data values in feet instead of meters.

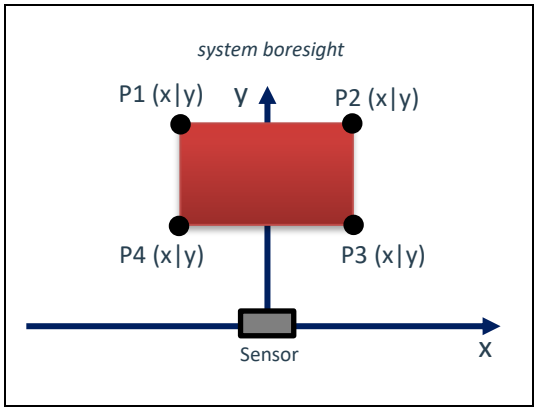
```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** SET EVENT ZONE **************/
ITR3800_EventZones_t event;
memset(&event, 0, sizeof(ITR3800_EventZones_t));
event.eventZones[0].enable = true;
event.eventZones[0].zoneType = ITR3800_EVENTZONE_PRESENSE;
event.eventZones[0].nrOfZonePoints = 4;
event.eventZones[0].zoneData[0] = {-20.0f,40.0f};
event.eventZones[0].zoneData[1] = { 20.0f,40.0f};
event.eventZones[0].zoneData[2] = { 20.0f,20.0f};
event.eventZones[0].zoneData[3] = {-20.0f,20.0f};
res = ITR3800_setEventZones(handle,event);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setEventZones failed\n");
}
QThread::sleep(1);
#endif

/************** GET NUMBER OF SET EVENT ZONES **************/
ITR3800_EventZones_t eventzones;
res = ITR3800_getEventZones(handle, &eventzones);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_getEventZones failed\n");
}
else
{
   uint16_t nrOfEventZones=0, i_event;
   for(i_event = 0; i_event < ITR3800_MAX_NR_OF_EVENTZONES; i_event++)
   {
      if(eventzones.eventZones[i_event].enable == true)
      {
         nrOfEventZones++;
      }
   }
   printf("number of set event zones:       %d\n", nrOfEventZones);
}

#ifndef READ_ONLY_FUNCTIONS
/************** CLEAR EVENT ZONE **************/
memset(&event, 0, sizeof(ITR3800_EventZones_t));
res = ITR3800_setEventZones(handle,event);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setEventZones failed\n");
}
QThread::sleep(1);
#endif
```

6.2.24. **set/getIgnoreZones**



This function sets the ignore zones. Each zone has 4 … 10 corner points. The sketch shows four points.

Set ignore zones to areas in which not tracks should be initialized. If a track is generated outside of an ignore zone it can move into an ignore zone.

After setting a zone the device performs an internal calculation which takes a short time. Please wait 1 second after a set function before the next function call.

There are also an imperial value function available. Just put in the zone data values in feet instead of meters.

It is highly recommended to clear the memory before setting the zones. All zones must have valid values to avoid malfunction.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** SET IGNORE ZONE **************/
ITR3800_IgnoreZones_t ignore;
memset(&ignore,0,sizeof(ITR3800_IgnoreZones_t));
ignore.ignoreZones[0].enable = true;
ignore.ignoreZones[0].nrOfZonePoints = 4;
ignore.ignoreZones[0].zoneData[0] = {-20.0f,40.0f};
ignore.ignoreZones[0].zoneData[1] = { 20.0f,40.0f};
ignore.ignoreZones[0].zoneData[2] = { 20.0f,20.0f};
ignore.ignoreZones[0].zoneData[3] = {-20.0f,20.0f};
res = ITR3800_setIgnoreZones(handle,ignore);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setIgnoreZones failed\n");
}
QThread::sleep(1);
#endif
/************** GET NUMBER OF SET IGNORE ZONES **************/
ITR3800_IgnoreZones_t ignoreZones;
res = ITR3800_getIgnoreZones(handle, &ignoreZones);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_getIgnoreZones failed\n");
}
else
{
   uint16_t nrOfIgnoreZones=0, i_ignore;
   for(i_ignore = 0; i_ignore < ITR3800_MAX_NR_OF_IGNOREZONES; i_ignore++)
   {
      if(ignoreZones.ignoreZones[i_ignore].enable == true)
      {
         nrOfIgnoreZones++;
      }
   }
   printf("number of set ignore zones:      %d\n", nrOfIgnoreZones);
}

#ifndef READ_ONLY_FUNCTIONS
/************** CLEAR IGNORE ZONE **************/
memset(&ignore, 0, sizeof(ITR3800_IgnoreZones_t));
res = ITR3800_setIgnoreZones(handle,ignore);
if(res != ITR3800_API_ERR_OK)
```

```
{
   printf("error – ITR3800_setIgnoreZones failed\n");
}
QThread::sleep(1);
#endif
```

6.3. Read object list functions

6.3.1. getObjectList

This function read the object list. The list will be updated for the ITR-3800 once every 70-85ms. If the list is read before an update is available, the corresponding function result will be set to inform the user.

There is also a function with imperial available. If you want to use imperial parameters just the other command in the same way.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** GET OBJECTS **************/
ITR3800_ObjectList_t pObjectList;
res = ITR3800_getObjectList(handle, &pObjectList);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_getObjectList failed\n");
}
else
{
   printf("number of objects: %d\n", pObjectList.nrOfTracks);
}
```

6.3.2. removeObject

This function removes objects within the object list. In case there is an uninteresting object in the object list it can be removed.

```
ITR3800_Result_t res;
APIHandle_t handle;

#ifndef READ_ONLY_FUNCTIONS
/************** REMOVE OBJECT **************/
res = ITR3800_removeObject(handle, 21);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_removeObject failed\n");
}
#endif
```

6.4. Camera functions

6.4.1. **getRtspUrl**

Get RTSP Video stream url.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** GET RTSP STREAM URL **************/
char *url;
res = ITR3800_getRtspUrl(handle, url);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_getRtspUrl failed\n");
}
```

6.4.2. **capture still camera image**

Receive a high quality camera image.

```
ITR3800_Result_t res;
APIHandle_t handle;

/************** RECEIVE STILL CAMERA IMAGE **************/
char *url;
res = ITR3800_setStillImageCallbackFn(handle, (void(*)())&imageReadyCallBack);
if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_setStillImageCallBackFn failed\n");
}

res = ITR3800_stillImageTrigger(handle);

if(res != ITR3800_API_ERR_OK)
{
   printf("error – ITR3800_stillImageTrigger failed\n");
}


void imageReadyCallBack(){
  ITR3800_PNGBuf *pngbuf;

  res = ITR3800_getStillImage (handle, pngbuf);

  if(res != ITR3800_API_ERR_OK)
  {
     printf("error – ITR3800_getStillImage failed\n");
  }

  //TO DO: Write PNG buffer to PNG file

}
```