PSD-MTDOT Radio Packet ICD

This document describes the packet structures sent and received over the Multitech MT-DOT LoRaWAN radio in the PNI Parking Sensor Demo.

TX Packet Format

A single, fixed-length packet structure is used to wrap the various data structures to be sent to the radio for transmission. In the future these may be broken out and sent individually to optimize transmission times.

All structures are packed and little-endian unless otherwise noted.

Parking Sensor Data Packet

Format

Type	Size	Field	Description
uint8	1	packet_type	Parking Sensor Data Packet Type
uint8[]	12	payload	See payload section for each packet type
uint32	1	timestamp	32 kHz ticks since boot

Example Code

```
struct parking_sensor_data_packet {
  uint8_t packet_type;
  union {
    struct car_detector car_detector;
    struct car_detector car_detector_data;
    struct configuration_state configuration_state;
    struct version_info version_info;
    struct meta_event meta_event;
    uint8_t bytes[12];
  } payload;
  uint32_t timestamp;
};
```

Parking Sensor Data Packet Types

Packet type that indicates the format of the payload.

Format

Value	Description
0x15	Car Detector Sensor
0x20	Car Detector Data Sensor
0x30	Configuration State

0x31	Version Info
0xFE	Meta Event

Example Code

```
enum parking_sensor_packet_types {
  PACKET_TYPE_CAR_DETECTOR =        0x15,
  PACKET_TYPE_CAR_DETECTOR_DATA =   0x20,
  PACKET_TYPE_CONFIGURATION_STATE = 0x30,
  PACKET_TYPE_VERSION =             0x31,
  PACKET_TYPE_META_EVENT =          0xFE,
};
```

Car Detector Payload

Car Detector sensor output containing the result of the car detection algorithm and additional information related to the result.

Format

Type	Size	Field	Description
uint8	1	result	Result
uint8	1	flags	Flags

Example Code

```
struct car_detector {
  uint8_t result;
  uint8_t flags;
};
```

Car Detector Result

Car detection result which contains the No Car/Car Parked state.

Format

Value	Description
0x00	Uninitialized
0x01	No Car
0x02	Car Entering
0x03	Car Parked
0x04	Car Leaving

Example Code

```
enum car_detector_results {
  CAR_DETECTOR_RESULT_UINITIALIZED =  0x00,
  CAR_DETECTOR_RESULT_NO_CAR =        0x01,
  CAR_DETECTOR_RESULT_CAR_ENTEREING = 0x02,
  CAR_DETECTOR_RESULT_CAR_PARKED =    0x03,
  CAR_DETECTOR_RESULT_CAR_LEAVING =   0x04,
};
```

Car Detector Flags

Additional information describing the Car Detector result.

Format

Bitfield

7	6	5	4	3	2	1	0	Description
0	0	0	-	-	X	X	X	Confidence
0	0	0	-	X	-	-	-	Calibration Complete
0	0	0	X	-	-	-	-	LS Mode

Example Code

```
struct car_detector_flags {
  uint8_t confidence:3;
  uint8_t calibration_complete:1;
  uint8_t ls_mode:1;
  uint8_t rfu:3;
};
```

Car Detector Data Payload

Magnetometer sample data (in microtelsa) from the RM3100RTI Parking Sensor.

Format

Type	Size	Field	Description
float	4	x	Magnetometer X (µT)
float	4	y	Magnetometer Y (µT)
float	4	z	Magnetometer Z (µT)

Example Code

```
struct car_detector_data_payload {
  float x;
  float y;
  float z;
};
```

Configuration State Payload

Structure containing the current state of the remote configurable options.

Format

Type	Size	Field	Description
uint8	1	sensors_enabled_state	Sensors Enabled State
uint8	1	car_detector_output_state	Car Detector Output State
uint16	2	wakeup_interval_s	Wakeup Interval (s)

Example Code

```
struct configuration_state_payload {
  struct sensors_enabled_state;
  struct car_detector_output_state;
  utin16_t wakeup_interval_s;
};
```

Sensors Enabled State

Flags describing which sensors are enabled.

Format

Bitfield

7	6	5	4	3	2	1	0	Description
0	0	0	0	0	0	-	X	Car Detector Enabled
0	0	0	0	0	0	X	-	Car Detector Data Enabled

Example Code

```
struct sensors_enabled_state {
  uint8_t car_detector_enabled:1;
  uint8_t car_detector_data_enabled:1;
  uint8_t rfu:6;
};
```

Car Detector Output State

Flags describing which optional Car Detector sensor modes are enabled.

- Continuous Mode – When disabled (default) only state changes are reported. When enabled all algorithm results are reported.
- Intermediate State – When disabled (default) only “No Car/Car Parked” states are reported. When enabled the intermediate states “Car Entering/Car Leaving” states are reported.

Format

Bitfield

7	6	5	4	3	2	1	0	Description
0	0	0	0	0	0	-	X	Continuous Mode Enabled
0	0	0	0	0	0	X	-	Intermediate State Report Enabled

Example Code

```
struct sensors_enabled_state {
  uint8_t continuous_mode_enabled:1;
  uint8_t intermediate_state_enabled:1;
  uint8_t rfu:6;
};
```

Wakeup Interval

Interval in seconds, between 10 and 43200, during which the device will sleep before waking up to check for incoming data.

Meta Event

Sensor state metadata. For PNI use only.

Format

Type	Size	Field	Description
uint8	1	type	Event type
uint8[]	2	data	Event Data

Example Code

```
struct meta_event {
  uint8_t type;
  uint8_t data[2];
}
```

Version Information

Structure containing the version information for the software components in the stack.

Format

Type	Size	Field	Description
struct	3	host	Host Application Version
struct	8	sensor	Sensor Firmware Version

Example Code

```
struct version_info {
  struct host_application_version host;
  struct sensor_firmware_version sensor;
};
```

Host Application Version

Version of the host (this) application.

Format

Type	Size	Field	Description
uint8	1	major	Major version number
uint8	1	minor	Minor version number
uint8	1	patch	Patch version number

Example Code

```
struct host_application_version {
  uint8_t major;
  uint8_t minor;
  uint8_t patch;
};
```

Sensor Firmware version

Version of the RM3100RTI Parking Sensor firmware.

Format

Type	Size	Field	Description
uint8	1	major	Major version number
uint8	1	minor	Minor version number
uint8	1	patch	Patch version number
uint8	1	other	Other version number
uint32	4	build	Build number

Example Code

```
struct sensor_firmware_version {
  uint8_t major;
  uint8_t minor;
  uint8_t patch;
  uint8_t other;
  uint32_t build;
};
```

RX Packet Format

Packets received from the radio consist of at least one byte for the packet type (command) and optionally a payload. Currently only the “Set Wakeup Interval” command has a payload.

All structures are packed and little-endian unless otherwise noted.

Parking Command Packet

Format

Value	Description
0x01	Recalibrate
0x02	Force Occupied
0x03	Force Vacant
0x04	Self-Test
0x05	Intermediate Report Enable
0x06	Intermediate Report Disable
0x07	Set Wakeup Interval
0x08	Car Detector Data Enable
0x09	Car Detector Data Disable

Example Code

```
enum parking_command_types {
  PARKING_CMD_TYPE_RECALIBRATE =                0x01,
  PARKING_CMD_TYPE_FORCE_OCCUPIED =             0x02,
  PARKING_CMD_TYPE_FORCE_VACANT =               0x03,
  PARKING_CMD_TYPE_SELF_TEST =                  0x04,
  PARKING_CMD_TYPE_INTERMEDIATE_REPORT_ENABLE =  0x05,
  PARKING_CMD_TYPE_INTERMEDIATE_REPORT_DISABLE = 0x06,
  PARKING_CMD_TYPE_SET_WAKEUP_INTERVAL =        0x07,
  PARKING_CMD_TYPE_CAR_DETECTOR_DATA_ENABLE =   0x08,
  PARKING_CMD_TYPE_CAR_DETECTOR_DATA_DISABLE =  0x09,
};
```

Set Wakeup Interval Payload

Contains the wakeup interval, in seconds, between 10 and 43200.

Format

Type	Size	Description
uint16	2	Wakeup Interval (s)

Example Code

```
typedef uint16_t wakeup_interval_s;
```