MCS ITalks

Technical product description for firmware 4.68

General description

The general profile is designed for the ITalks MCS 1608 Full version. This profile is used for tracking objects, sending measurement data and detecting rotation and movement.

The unit can be configured by sending commands via the downlink channel.

Tracker Sensor

The unit has two main states:

- Not Moving, or Idle; The unit will detect motion via the accelerometer, or magnetometer every second and after 20 (default for accelerometer and immediately for magnetometer) seconds of motion it will go to the moving state
- Moving: The unit will try to get an indoor location when enabled. If not enabled or not successful the unit will try to get a GPS fix. This process is repeated every 3 minutes.

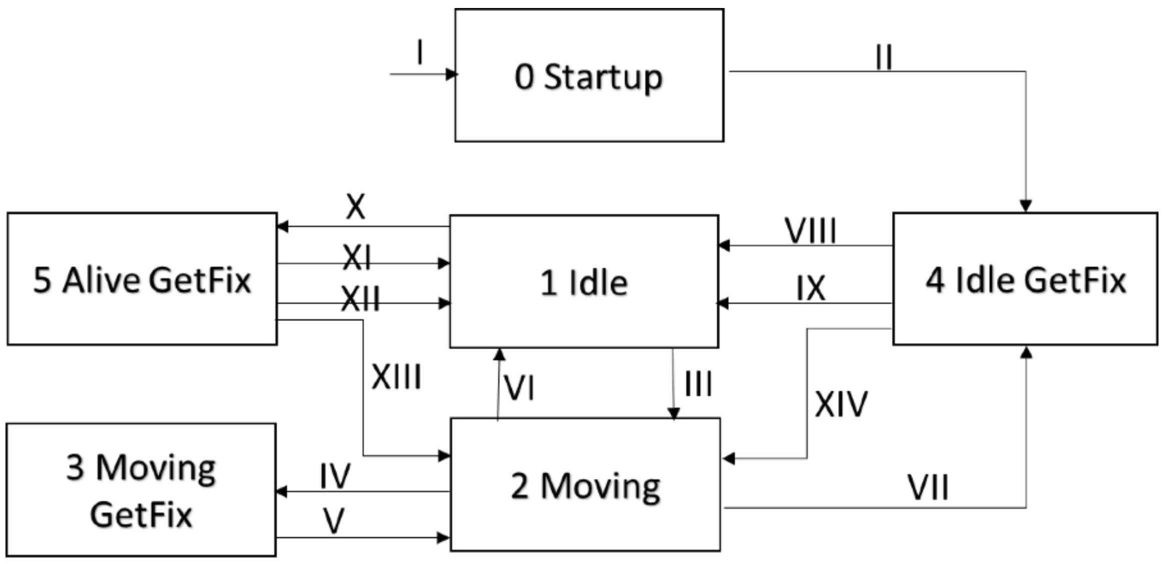
In the idle state all sensors, the CPU and the GPS receiver are switched off, only the accelerometer is active. It tries to detect motion and notifies the CPU immediately when motion is detected. This is extremely energy efficient and uses around 30 uA.

Formal description

The ITalks MCS 1608 Tracker Sensor operates as a “State Machine” and it has the following formal states:

State #	**State**	**Action**	
0	Startup	Initial	Initialize ITalks MCS 1608 and wait 10 seconds
		Continuous	Goto IdleState
1	Idle	Initial	Disable GPS
		Continuous	Check accelerometer for Moving or Check magnetometer for Moving Check Alive Timer
2	Moving	Initial	Enable GPS when configured to do so
		Continuous	Check accelerometer for Idle or Check magnetometer for Idle Goto Moving GetFix every 3 (configure) minute
3	Moving Getfix	Initial	Get Indoor Localization; if not possible Enable GPS
		Continuous	Wait for GPS Fix (Max 4 minutes)
4	Idle GetFix	Initial	Get Indoor Localization; if not possible Enable GPS
		Continuous	check accelerometer for Moving or check magnetometer for Moving Wait for GPS Fix (Max 4 minutes)
5	Alive Getfix	Initial	Enable GPS
		Continuous	check accelerometer for Moving or check magnetometer for Moving Wait for GPS Fix (Max 4 minutes)

The states and their transitions are shown in the following figure:



The state transitions are defined as follows:

Start State		Transition	End State		Description	
0	Startup	II	4	Idle Getfix	Condition	After 10 seconds Sent MsgIDReboot Sent MsgIDAlive

Start State		Transition	End State		Description	
1	Idle	III	2	Moving	Condition	Accelerometer detects movement for #DTMove seconds Or MagnRot event Sent Message MsgIDStart
		X	5	Alive GetFix	Condition	Every 6 hours

Start State		Transition	End State		Description	
2	Moving	VI	1	Idle	Condition	No more movement or MagnRot events for #DTIdle seconds AND Valid GPS Fix
		VII	4	Idle GetFix	Condition	No more movement or MagnRot events for #DTIdle seconds AND No Valid GPS Fix
		IV	6	Moving GetFix	Condition	Every #UTMoving minutes

Start State		Transition	End State		Description	
4	Idle GetFix	XIV	2	Moving	Condition	Movement detected for #DTMove seconds or MagnRot event
		X	1	Idle	Condition	After #TOGPSFix seconds Sent Message MsgIDStopNoFix
		VIII	1	Idle	Condition	GPSFix found then wait 20 seconds Sent Message MsgIDStopFixOk

Start State		Transition	End State		Description	
5	Alive GetFix	XIII	2	Moving	Condition	Accelerometer Detects Movement for #DTMove seconds or MagnRot event Sent Message MsgIDStart
		XI	1	Idle	Condition	After 4 minutes Sent Message MsgIDGPSAbort
		XII	1	Idle	Condition	GPSFix found Sent Message MsgIDAlivePos

Start State		Transition	End State		Description	
3	Moving GetFix	V	2	Moving	Condition	After #TOGPSfix minutes Sent Message MsgIDGPSAbort
					Condition	GPSFix found Sent Message MsgIDMovingFix

Indoor Localization

The ITalks MCS 1608 General Sensor profile when tracker is enabled and indoor localization is enabled will activate indoor localization. This means that the unit will try to ping beacons in the area. If a beacon receives such ping it will respond to it. The response is received by the device and stored. If indoor localization is successful (at least one response to a ping received), the unit will send an indoor location message instead of a tracking message.
The Indoor Localization Message contains the RSSI of three beacons with the strongest signal.

Based on the signal strength of the beacons it is possible to estimate the position of the device relative to the beacons.

For further explanation please view https://www.youtube.com/watch?v=CWvRJdF7oVE

There are many articles available on the internet to calculate position of a device based on RSSI. In many applications it is enough to know that a device is near a beacon and what beacon has the strongest signal.

Traveled Distance

The ITalks MCS 1608 General Sensor profile when tracker is enabled registers the traveled distance. This distance will be used for future trip registration and be reported in a to be developed Trip Message (Future)
Note: The distance meter uses the Geofence functionality to determine the interval of traveled distance calculation. Setting the #Geofence to "0" will disable traveled distance registration.

Vibration Sensor

The vibration sensor detects vibration of the ITalks MCS 1608. It registers the frequency and the amplitude of the three most intense vibrations in the spectrum.

The vibration detection is able to detect frequencies up to 650Hz.

The unit scans for vibrations at a certain interval #ScanIntVibrate, when #ScanIntVibrate is set to '0' no vibration is detected anymore

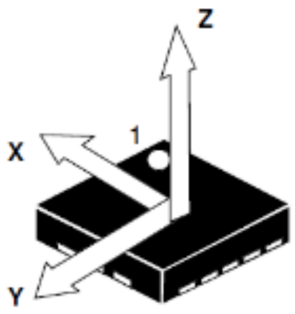
When a Vibration is detected the unit can send an alarm message (if #EnVibrAl is set to 1, default is Off).
The alarm will be sent again if the frequency of the vibration changes.
The unit will periodically send a Vibration Message. The period can be changed via Parameter #UTVibrating. If #UTVibrating is set to 0, the vibration message is sent when the vibration stops.
After sending an alarm message and after sensing the vibration message the vibration is set to '0' so the next message contains the most recent vibration data.

Running hours

During vibration the unit accumulates running hours.

Shock Sensor

The shock sensor detects shocks of the ITalks MCS 1608. It registers the maximum amplitude of the three Axes X, Y and Z and calculates the Shock Impact (experimental)



What is a shock?
An easy answer would be a sudden increase (or decrease) in acceleration.
This also how the ITalks MCS 1608 measures shock.

Tapping the ITalks MCS 1608 might already lead to high accelerations, but are they shocks? In the way the shock sensor in the ITalks MCS 1608 works the answer is YES.
The shock sensors senses for shocks by sampling the Accelerometer at a frequency of 50 Hz. As soon as the acceleration comes above a certain threshold (the sensitivity). The sampling frequency is increased to 200Hz and during a certain period (now 4 seconds). During this period the maximum "deviations" of the acceleration in the X, Y and Z direction are measured and stored. "Deviations" means that the normal acceleration, due to gravity of the earth is filtered out, before any calculation on the accelerometer values are performed.

A more sophisticated way of defining shocks is a sudden in- or decrease in acceleration where the acceleration changes are between certain frequency limits. High frequencies (like a tap) may result in high accelerations, but the energy they contain is normally very low. In other words they do not give a high impact.
For that reason an Impact is calculated, based on an FFT performed on the change in the XYZ value (vector) of the accelerometer during 4 seconds after the first impact. The accelerometer is sampled at 200Hz.

The FFT calculation is performed continuously and at each frequency band the maximum FFT value is stored. This leads to a sample set like this:

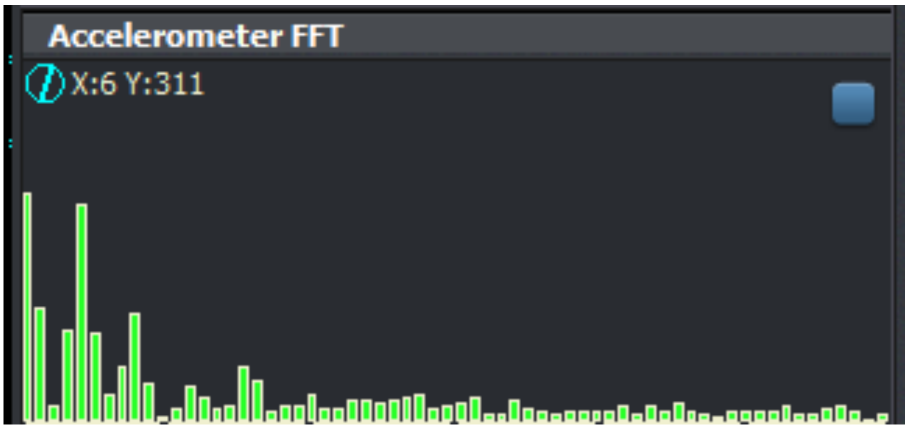


Figure 1 FFT Spectrum 0..100Hz

After 4 seconds the values of the first frequency bands (<30Hz) are averaged. This average is reported as an impact value. This is Experimental!!!

There are 2 ways of reporting the shocks the way is configured by the reporting variable #ShockInterv. If ShockInterv = 0, shocks are reported immediately. After every detected shock, a Shock Message is send. Any value of ShockInterv between 1 and 255 gives the interval in minutes of the shock reporting.

1. The number of shocks are counted and the maximum values of X, Y and Z are reported at a certain interval ShockInterv 1..255 (in minutes)

2. Every shock is reported in a separate shock message ShockInterv = 0

The sensitivity of the shock detector can be set with the variable ShockSens. #Shocksens = 0 disables shock sensor; The sensitivity is given in 0,1G that means that a value of 20 means shock above 2G are reported.

The following configuring commands are used for the shock sensor:

Parameter	**Description**	**Factory Value**	**Command to change**
#CmdShockIntv	Shock report Interval 0..255 in minutes	0 = Immediate	0x2F
#CmdShockGain	Shock Gain 0..3 (see description)	2 = 8G Max	0x30
#CmdShockSens	Shock Sensitivity 0..255 in 0,1G	0 = Off	0x31

The shock message contains the following parameters:

MaxDx, MaxdY and MaxDZ giving the maximum acceleration in X, Y and Z direction.
Combined with the reported gain the value of acceleration in any of the give directions van be calculated with the following formula:

$$\Delta Acceleration_{xyz}(in\,G) = \frac{MaxD_{xyz} * 2^{Gain}}{256}$$

Example: if Gain is 2 and MaxDx is 325 it means that the actual acceleration during the shock in the X-direction is:

$$\Delta Acceleration_{x}(in\,G) = \frac{325 * 2^{2}}{256}$$

So the acceleration is 325*4/256 = 5,07 G
In the same way the acceleration in Y and Z directions can be calculated.

The Range (or Gain) is reported 0 = 2G; 1= 4G; 2 = 8G; 3 = 16G

The number of shocks in the last period is reported.

Rotation Sensor

The ITalks MCS 1608 General Sensor profile has two Rotation Sensors on board.

1. The magnetic rotation sensor, detects changes in the magnetic field and sends a rotation or an alarm message when a change is detected. If the Parameter #RotMagAl is On, the Sensor will sent an Alarm message otherwise the Sensor will sent a Rotation Message.
   When the Parameter #RotMagnSens is set to "0" detection is switched off.
   The sensitivity of the magnetic rotation sensor can differ a lot on different locations. The default Value will work in most circumstances. But can be changed via a command.

2. The Gravity rotation sensor, detects changes in the earth's gravity field and sends a rotation or an alarm message when a change is detected. If the Parameter #RotGravAl is On, the Sensor will sent an Alarm message otherwise the Sensor will sent a Rotation Message.
   When the Parameter #RotGravSens is set to "0" detection is switched off.
   The sensitivity of the gravity rotation sensor can be specified in degrees (1..90) . After 1 minute of absolutely no movement at all the sensor sets its reference orientation and starts detecting rotation.

Movement Alarm Sensor

The movement alarm sensor detects movement. After 3 seconds of movement it sends an Alarm Message. The sensor is then deactivated for 30 seconds to 1 minute. After that it will again sent an alarm message The sensor can be switched off using the #EnMotAlarm command.

Barometer/Temperature/Relative Humidity and Beam Level Sensor

The ITalks MCS 1608 can be configured to send its sensor values on a regular time interval. The time interval can be configured using the #UTSensors command. The time is given in minutes. If the parameter #UTSensors is set to "0" no updates will be sent.

Geofence

The ITalks MCS 1608 can be configured to send GeoFence violations as alarm messages.
Default the messages are Off.
The alarm can be switched on with the #GeoFenceAl Command.
The default value for the GeoFence Radius can be changed also. For this the #GeoFence command is used.
After a GeoFence violation, the center of the GeoFence is set to the new position and the sensor starts monitoring GeoFence violation around this new center point.
In order to detect GeoFence violations the GPS sensor must be on.

NOTE: Setting the #GeoFence to "0", will disable the GeoFence Alarms also, but it will also disable KM/Mileage registration, since the GeoFence functionality is also used to set a distance for new KM/Mileage calculation.

1Wire Temperature Sensor

1Wire Temperature sensors, type DS1820 can be connected to the ITalks MCS 1608. A special 2 wire connector is available.
However we recommend to connect the 1 Wire sensors with three wires. The sensors with the correct connectors are available through your supplier.
The maximum 1 wire temperature sensors that can be connected to the interface is 5.
The 1Wire measurements are sent at the interval specified by #UT1WireT.

Setting +5V, pin 5 of 10pin connector

With the downlink command it is possible to set the +5V of the ITalks MCS 1608.
By sending the command #CmdSet5V with parameter 0xFF the +5V is permanently switched on. By setting the Parameter to 0x00 the +5V is switched off. Any parameter in between gives a pulse of <parameter>*10ms on the output.

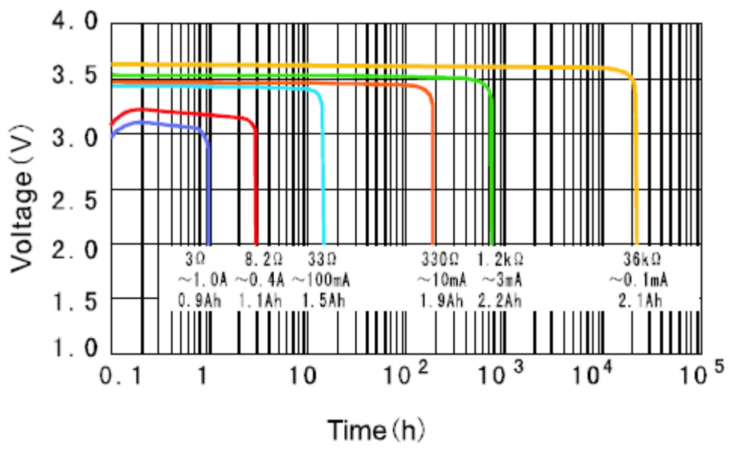
Please be aware that a load on the +5V can easily drain the battery very quickly.

Digital Inputs/Counters

See separate document
https://drive.google.com/uc?id=1qVLrJ3DAHWoCv3jgaFU9OuXj-7Qo3Ns2&export=download
www.1m2m.eu/downloads/Description ITalks MCS 1608 Digital Inputs V1.01.pdf

Battery

The used primary Lithium Metal batteries have very flat discharge characteristic.



In previous versions the battery voltage was used to measure the remaining capacity of the battery.
As of V4.6 the battery usage is calculated by the ITalks MCS 1608 itself and stored in Non Volatile Memory.
At boot, the default Battery Capacity is set to 6.16 Ah. A standard 4-cell battery is 8.8 Ah but in worst case (very low or very high temperature) this capacity can go down to 6.16 Ah, so we use the low value to be safe.
From that moment on the used capacity is calculated for the following power consumers by multiplying the time a consumer is active with the measured current per consumer on a reference device.

- MCU (Processor)
- LoRa Radio
- SigFox Radio
- GPS
- LED's
- Other Sensors (like DigIn)

Furthermore it keeps track of Idle Usage of the unit.
Every 6 hours a usage report is sent to the Terminal:

```
Battery capacity : 6160 mAh
Cumulative Power use : 2528 mAs
Base: 2  GPS : 1628
MCU : 800  Sfx :  0
LoRa: 98  LED : 106
Misc: 0
```

All values in this report are in mAs.
To convert it to mAh values must be divided by 3600.

The functionality is tested but we have no long-term experience with it. Please report any issues you have at support@1m2m.eu
The battery capacity and remaining capacity can be set via the terminal and downlink messages. See Chapter "System Commands" for details.

Capacity is an estimation and based on a temperature 10 to 30 ^{o}C during lifetime. When used in very cold environments battery lifetime will be less. Please contact us for details.

LED status

The ITalks MCS 1608 has three LED's give the following information:

RED-Led

- Fast flashes: LoRa Transmission
- 1 blink per 10 seconds: Tracker Moving
- 2 blinks per 10 seconds: Vibration detected
- 3 blinks per 10 seconds: Getting GPS Fix
- 3 long blinks: SigFox Transmission

GREEN-Led (Only on Full Version)

- Flashes during power up

YELLOW-Led (Only on Full Version)

- Fast flashes: LoRa Downlink Message
- 1 flash per second: Duty Cycle Limitation active or joining

Parameters

The following parameters can be changed via downlink messages

Parameter	**Description**	**Default Full**	**Default Basic**	**Default R.Only**	**Change command**
#DTMove	Delay to Move	20 sec	20 sec	20 sec	0x01
#DTIdle	Delay to Idle	30 sec	30 sec	30 sec	0x02
#ScanIntVibrate	Scan Interval for vibration	0 sec (off)	0 sec (off)	0 sec (off)	0x03
#TOGPSFix	Time out for GPS Fix	240 sec	240 sec	240 sec	0x04
#UTMoving	Update Time Moving 0..255 Min 0 Means no updates of position while moving	15 min	15 min	15 min	0x05
#UTVibrating	Update Time while Vibrating	10 min	10 min	10 min	0x06
#UTIdle	Update Time while Idle (alive message)	6 hours	6 hours	6 hours	0x07
#AliveHR	Hour in UTC when an alive message must be send (0..24)	0 (off)	0 (off)	0 (off)	0x08
#GPSOnShock	Set to 1 will activate a GPS fix when a shock is detected	0 (Off)	0 (off)	0 (off)	0x09
#UTSensors	Update time sensor values	15 min	0 min	0 min	0x0A
#EnMotAlarm	Motion Detection Alarm 1 is enabled 0 is disabled	0 (off)	0 (off)	0 (off)	0x0B
#RotMagAl	Rotation Magnetic Detection //1 is enabled 0 is disabled	0 (off)	0 (off)	0 (off)	0x0C
#RotGravAl	Rotation Gravity Detection //1 is enable 0 is disable	0 (off)	0 (off)	0 (off)	0x0D
#TrackerOn	Enable tracker 1 is enable for accelerometer 2 is enabled for magnetometer 0 is disable tracker	1 (on)	1 (on)	0 (off)	0x0E
#TripOn	Enable Trip Reporting //1 is enable 0 is disable trip reporting	0 (off)	0 (off)	0 (off)	0x0F

Parameter	Description	Default Full	Defaul t Basic	Defaul t R.Only	Change command
#RotMagnSens	Magnetic Rotation Sensitivity (0 = Off) (try 100 to actually use it for Rotation Detection) **This value also sets the sensitivity for the magneto enabled tracker**	0 (off)	0 (off)	0 (off)	0x10
#RotGravSens	Gravity Rotation Sensitivity in Degrees < 90 (0 = Off)	0 (off)	0 (off)	0 (off)	0x11
#GeoFence	Geofence Radius in meters / 10 (0 = Off)	50m	50m	50m	0x12
#GeoFenceAl	GeoFence Alarm //1 is enable 0 is disable	0 (Off)	0 (off)	0 (off)	0x13
#SendStartMessage	Reports when a device starts moving	1 (On)	1 (on)	0 (off)	0x14
#WAGPSFix	Wait after GPS fix (only when going to Idle)	20 sec	20 sec	20 sec	0x15
#AccSens	Sensitivity of the accelerometer	5	5	5	0x16
#GPSOnMov	GPS On while Moving	0 (Off)	0 (off)	0 (off)	0x17
#UT1WireT	Update Time 1 Wire sensors in minutes	0 (Off)	0 (off)	15 min	0x18
#MaxTempLimit	Sends an Alarm Message when temperature is above this limit (for ext and internal Temp sensor)	2000 (off) (200.0 C) = No Alarm	2000 (off)	2000 (off)	0x19
#DRMoving	Datarate while moving (7..12)	0x0C (SF12)	0x0C	0x0C	0x1A
#UseGravXYZ	(1) Send GenSensMsg (2) Send GenSensGravMsg	0	0	0	0x1C
#EnVibrAl	Enable Vibration Alarm 0 or 1	0 (No Alarm sent)	0	0	0x1D
#UTAnalog	Update Time Analog Inputs (Including Battery Voltage) in minutes	0 (Off)	0	30 min	0x20

Parameter	Description	Default Full	Defaul t Basic	Defaul t R.Only	Change command
#DigIn1Mode	0 = Off 1 = Switch 2 = Counter 3 = SlowSwitch (see description)	0 (Not Used)	0	0	0x21
#DigIn1ReportMode	1 = Alarm 2 = Time based 3 = Count based (see description)	3 (Count Based)	3	3	0x22
#DigIn1Hold On/off Time	1..255 (see description)	1(10 Sec)	1	1	0x23
#DigIn1ReportValue	1..255 (see description)	10 (10 minutes) 10 (10 Counts)	10	10	0x24
#DigIn2Mode	0 = Off 1 = Switch 2 = Counter 3 = SlowSwitch (see description)	0 (Not Used)	0	0	0x25
#DigIn2ReportMode	1 = Alarm 2 = Time based 3 = Count based (see description)	3 (Count Based)	3	3	0x26
#DigIn2Hold On/off Time	1..255 (see description)	1(10 Sec)	1	1	0x27
#DigIn2ReportValue	1..255 (see description)	10 (10 minutes) 10 (10 Counts)	10	10	0x28
#CmdSet5V	0 = Off 1..254 = set on for 10..2540 ms FF = On	0	0	0	0x2D
#CmdIndoorLoc	0 = Disabled 1 = Enabled	0 = Enabled	0	0	0x2E
#CmdShockIntv	Shock report Interval 0..255 in minutes	0 = Immediate	0	0	0x2F
#CmdShockGain	Shock Gain 0..3 (see description)	2 = 8G Max	2	2	0x30
#CmdShockSens	Shock Sensitivity 0..255 in 0,1G	0 = Off	0	0	0x31
#CmdMAXGPSHDOP	1..10	10 (HDOP 100)	10	10	0x32
#CmdShowConfig	N.A.	N.A.	N.A.	N.A.	0x33
#Cmd5VOnBeforeAnalog	See *footnote#1*	0	0	0	0x34

Footnote#1
Cmd5VOnBeforeAnalog is used to enable the 5V output some time before the values on AnIn1 and AnIn2 are measured. This can be used to allow some types of 5Volt powered analog sensors to reach a stable

output before reading the values. To deal with the limitations of a single byte parameter two ranges are used. A value of 0 means no 5V output before measurement.
Values between 1 and 127 (0x01 to 0x7F) set the pre-power time in seconds.
Sending config string 340A selects a 10 second power-on before measurements are made.
Values between -1 and -128 (0xFF to 0x80) set the pre-power time in 10 milli second slots.
Sending config string 34FB (-5) selects a 50 ms power-on before measurements are made.
In the range of 0.010 to 1.280 seconds the time resolution is 10 ms.
In the range of 1 second to 127 seconds the time resolution is 1000 ms.

Commands

On 28-05-2019 the format of downlinks and comport commands was seriously changed.
This was required because some devices were reprogrammed with non-sense config, probably via accidental receptions by the radio.
For reference the previous description is still in this document, but it is coloured grey. The new description is just below the grey text.

Commands are defined by their number as mentioned in the table above.
Commands are sent as downlink messages.
The downlink message format is as follows:

```
typedef struct {
 byte CmdSeq;     // Should be incremented every command given
 byte Cmd;        // Command number as mentioned in table above
 byte NewValue;   // The new value of the parameter to be changed
} TGenSensCmd;
```

A command is answered by the ITalks MCS 1608 via a TaliveMsg. The parameter CmdAck in this message contains the last received command.

When entering commands using the terminal it is now no longer necessary from V4.0 and higher to enter the sequence number and can be left out. Via the terminal the commands are 2 bytes. 3 byte commands are still allowed but if used then valid sequence numbers must be used.

System Commands (V4.0 and higher)
System commands are not application specific and should work in any application (V4.0 and above)
The sequence number of system commands is always 0xF0

Commands for setting configuration parameters and System Commands are no longer accepted unless they follow a specific format which has a valid 2 byte security checksum.
In firmware version V4.67 and all versions before, just 2 bytes were sufficient to set configuration parameters. This could happen by accident.
In firmware version V4.68 and higher it takes 6 bytes to change a parameter.
It uses this struct:

```
typedef struct {
        byte ParamID;          // Param/Cmd identifier
        word ParamValue;       // 2 byte value
        byte CfgSequence;      // 1 byte sequence for confirmation in Alive Message
        word CS;               // 2 byte security check
} TCmdStruct;
```

ParamID is one of the app or system command values.
ParamValue is the new value, 0x0000 if not used.
CfgSequence is a byte that is echoed in the next Alive or ParamAck message.

Its value is not used, just for confirmation to the server. CS is what is really important. It is calculated from the first 4 bytes and if it is not valid, the command is not accepted.

CS calculation in C

```
#define byteword union {word w; struct {byte lo; byte hi;};}
void PacketRx(byte *p, byte Length)
{
 if (Length == 6) {
		byteword Val; byteword CS1; byte Cmd,Seq;
		word CS2 = 0xA55A;
		Cmd = *p++; CS2 <<= 2; CS2 += Cmd;
		Val.hi = *p++; CS2 <<= 2; CS2 += Val.hi;
		Val.lo = *p++; CS2 <<= 2; CS2 += Val.lo;
		Seq = *p++; CS2 <<= 2; CS2 += Seq;
		CS1.hi = *p++;
		CS1.lo = *p;
		if (CS1.w == CS2) ExecCmdData(Cmd,Val.w);  // Accept command
    }
}
```

System Commands

System commands are not application specific and should work in any application (V4.0 and above)

```
#define CmdSetDeepSleepMde      0xF5    // New since V4.65 (19-Apr-2019)
#define CmdSendBatteryRep       0xF6    //
#define CmdSetUsedBatCap        0xF7    //
#define CmdResetBatt            0xF8    //
#define CmdLPWANUse             0xF9    //
#define CmdMonitorOn            0xFA    //
#define CmdStartUpgrade         0xFB    //
#define CmdSetABPKeys           0xFC    //
#define CmdSetAPPEUI            0xFD    //
#define CmdReset                0xFE    //
#define CmdFactReset            0xFF    //
```

Set DeepSleepMode command

This command sets the power state and magnetic swipe wakeup behaviour of the device.

```
		byte CmdSeq = 0xF0
		byte Cmd = 0xF5
		byte NewValue = 0x00
```

Value = 0x00: Disabled.
 The hall effect switch is not powered, the device starts normal operation after reboot. The device will not detect magnetic swipes and will not reboot when a magnet is swiped. The application code can activate the hall effect for other purposes.
Value = 0x01: DeepSleep.
 The hall effect switch is powered and ready to detect magnetic swipes.
All other hardware is put in power saving mode. All radios remain disabled in this mode.
A valid swipe detect puts the device in DeepSleep mode 0x02 and triggers a reboot.
Value = 0x02: NormalOperation
The hall-effect switch is powered and ready to detect magnetic swipes.
The device is in normal operation. A valid swipe detect puts triggers a reboot.

Send Battery Report command

The command can be used to get a battery report message.
Has the following content:

byte CmdSeq = 0xF0
byte Cmd = 0xF6
byte NewValue = 0x00

Value = 0x00 send once

SetUsedBatCap command

Is used after a firmware upgrade (from a Version < V4.6) to tell the ITalks MCS 1608 it has a used battery (with value <> 0) or a new battery (with value = 0) The value given is stored as the used capacity of the battery in 0,1 Ah per unit. The ITalks MCS 1608 will use this value to calculate the remaining battery capacity.
Has the following content:

byte CmdSeq = 0xF0
byte Cmd = 0xF7
byte NewValue = 0x<value>

Value = 0 means a fresh battery, note that giving this command with value 0 does exactly the same as the CmdResetBatt command, except that it leaves the Battery Capacity unchanged.

ResetBatt command

Is used when the battery is replaced, to tell the ITalks MCS 1608 it has a new battery. The value given is stored as the capacity of the battery in 0,1 Ah per unit. The ITalks MCS 1608 will use this value to calculate the remaining battery capacity.
Has the following content:

byte CmdSeq = 0xF0
byte Cmd = 0xF8
byte NewValue = 0x<value>

The normal settings for <value> are:
0x3E = Decimal 62 = 6,2Ah Battery (4 pack) ==This is the default value==
0x0F = Decimal 15 = 1,5Ah Battery (Single Cell)
If <value> is set to 0xFF (255) the ITalks MCS 1608 will report its battery value the traditional way
If <value> is set to 0x00 (0) the ITalks MCS 1608 will always report 0 as a remaining capacity.
If the battery is replaced with a battery of the same capacity, the SetUsedBatCap command is preferred.

LPWAN Use Command

has the following content:

byte CmdSeq = 0xF0
byte Cmd = 0xF9
byte NewValue = Set LPWAN use:

1 (SigFox)
2 **(LoRa)** (def)
3 (SigFox&LoRa)
4 (Switch to ABP)
5 (Switch to OTAA)
(if enabled and netw. Params available)

Monitor Command

has the following content:

byte CmdSeq = 0xF0
byte Cmd = 0xFA
byte NewValue = Enable the Terminal:

0 = Off (also disables input of commands)
1 = On
2 = No DebugVar

Reboot command

has the following content:

byte CmdSeq = 0xF0
byte Cmd = 0xFE
byte NewValue = 0xEF

The ITalks MCS 1608 will perform a reboot

Factory Reset command

has the following content:

byte CmdSeq = 0xF0
byte Cmd = 0xFF
byte NewValue = 0xFE

The ITalks MCS 1608 will restore parameters to factory default and perform a reboot

Upgrade command

has the following content:

byte CmdSeq = 0xF0
byte Cmd = 0xFB
byte NewValue = 0xEF

The ITalks MCS 1608 will reboot every now for maximal 24 times or until it receives new firmware.

Set APPEUI command

This command also enables the use of OTAA and disables APB!

Has the following content:

byte CmdSeq = F0<specified by user>
byte Cmd = 0xFD
byte APPEUI[8] = <specified by user>
byte APPLEY[16] = <specified by user> !!OPTIONAL!!

The ITalks MCS 1608 will store the APPEUI and if given the APPKEY perform a reboot.
If APPKEY is left empty, the length of the message is 10 bytes, the APPKEY is not changed.
If APPKEY is specified, the length of the message is 26 bytes and the specified APPKEY is used.
Example : 01FD0102030405060708 in a downlink sequence will set the

APPEUI to 0x0102030405060708.
APPKEY is not changed

Example : 01FD0102030405060708090A0B0C0D0E0F101112131415161718 in a downlink sequence will set the

APPEUI to 0x0102030405060708.
APPKEY to 0x090A0B0C0D0E0F101112131415161718

The DEVEUI cannot be changed.

In firmware V2.6 and lower a factory reset will restore the original APPSEUI and APPKEY.

Changing APPEUI via the terminal (serial cable or ED1000 required)

Step 1: Connect the ITalks MCS 1608 to the terminal via the serial interface
Step 2: In the terminal go with the mouse to the Command entry area, and make sure the button "Direct" is not selected.
Step 3: type "APPEUI=1122334455667788APPKEY=00112233445566778899AABBCCDDEEFF" *) and press Send

The ITalks MCS 1608 will respond with the message "APPEUI Set:8877665544332211" *) and APPKEY Set: 00112233445566778899AABBCCDDEEFF and when the unit goes to idle it will reboot.
This command also enable the use of OTAA and disables APB!
*) the APPKEY part is optional if this part is not entered the APPKEY will remain unchanged
*) the APPEUI is displayed in reverse order, however the APPEUI is correctly set in the unit
*) In firmware V2.6 and lower a factory reset command will erase the APPEUI and go back to the original setting

Set ABP Parameters via downlink

This command also enables the use of ABP and disables OTAA!

Has the following content:

```
byte CmdSeq = F0<specified by user>
byte Cmd = 0xFC
byte DevAddr[4] = <specified by user>
byte AppSKey[16] = <specified by user>
byte NwkSKey[16] = <specified by user>
```

The ITalks MCS 1608 will Store the ABP (Activation by personalization parameters) and perform a reboot
Example : 01FC0102030405060708090A0B0C0D0E0F101112131415161718191O1A1B1C1D1E1F20212223

in a downlink sequence will set the
DevAddr = 01020304
AppsKey = 05060708090A0B0C0D0E0F1011121314
NwkSKey = 1516171819101A1B1C1D1E1F20212223

In firmware V2.6 and lower a factory reset will restore the original ABP Parameters (usually empty)

Setting ABP Parameters via the terminal (serial cable or ED1000 required):

Step 1: Connect the ITalks MCS 1608 to the terminal via the serial interface
Step 2: In the terminal go with the mouse to the Command entry area, and make sure the button "Direct" is not selected.
Step 3: type "DEVADDR=00112233NWSKEY=00112233445566778899AABBCCDDEEFFAPPSKEY=00112233445566778899AABBCCDDEEFF" and press Send

The ITalks MCS 1608 will respond with the message
DevAddr Set:33221100 *)
NwkSKey Set:00112233445566778899AABBCCDDEEFF
AppSKey Set:00112233445566778899AABBCCDDEEFF

and when the unit goes to idle it will reboot.
This command also enables the use of ABP and disables OTAA!

*) the DevAddr is displayed in reverse order, however the DevAddr is correctly set in the unit
*) In firmware V2.6 and lower a factory reset command will erase the ABP Parameters and go back to the original setting

Formal PayLoad description

General Sensor Message Format

This is the latest message format, to fully use the General Sensor capabilities.
There are currently 18 message types

• MsgIDAlive	MsgID 0x00
• MsgIDTracking	MsgID 0x01
• MsgIDGenSens	MsgID 0x02
• MsgIDRot	MsgID 0x03
• MsgIDAlarm	MsgID 0x04
• MsgID1WireT	MsgID 0x06
• MsgIDRunning	MsgID 0x07
• MsgIDVibrate	MsgID 0x08 (obsolete, use 0x95 instead)
• MsgIDAnalog	MsgID 0x09
• MsgIdGenSensGravMsg	MsgID 0x0A (Same as MsgIDGenSens, but with raw gravity values)
• MsgIdDailyReport	MsgID 0x0B
• MsgIdDigIn1Msg	MsgID 0x0C
• MsgIdDigIn2Msg	MsgID 0x0D
• MsgIDIndoor	MsgId 0x81
• MsgIdShock	MsgID 0x82
• MsgIDReboot	MsgID 0x0E
• MsgIDBatteryReport	MsgID 0xF0
• MsgIDVibrate_6_25	MsgID 0x95 (replaces MsgID 0x08, gain corrected.
• MsgIDRFU1	MsgID 0xFC reserved for another 256 messagetypes
• MsgIDRFU2	MsgID 0xFD reserved for another 256 messagetypes
• MsgIDRFU3	MsgID 0xFE reserved for another 256 messagetypes

For some values a "lossy compression" is used to get more range in size limited payloads.
For some values individual bits are put in "container bytes" that hold several types of data.

Fixage is not linearly encoded.
If it was less than an hour ago the resolution is 1 minute, but after an hour the resolution becomes less important than the range. To decode the GPS Fix age you need a function like this:

```
uint16_t GetFixAge(uint8_t FixAge)            // in minutes [0..4140]
{
  if (FixAge<60) return FixAge;
  if (FixAge<120) return 60+(FixAge-60)*5;
  return 120+(FixAge-120)*30;
}
```

Everything below 60 is in minutes
60 to 119 is in 5 minutes intervals
120 to 254 is in 30 minute intervals
255 results in 4148 minutes = 73 hours

Barombar has an offset of -100000
To get the real value typecase the 16 bit value to an int16, and add it to a uint_32 with a value of 100000.
The range is 672.33 mBar to 1327.67 mBar (normal around sea level is 1020.00 mBar)

Latitude, **Longitude** and **SatCnt** are compressed like this:
The lower 24 bits of Latitude and longitude are in 2 3 byte arrays, MSB first.
Latitude has a range of -90.00000 degrees to +90.00000 degrees (25 bit)
Longitude has a range of -180.00000 degrees to +180.00000 degrees (26 bit)
The number of satellites used in the fix is (SatInFix) is encoded in the lower 5 bits of SatCntHiLL.
Bit 24 of the latitude is encoded in bit 5 of SatCntHiLL.
Bit l25 and 24 if the longitude are encoded in bit 7 and 6 of SatCntHiLL.

The messages always contain the most recent data. If no updated data is available the old data is sent. The payloads can be decrypted via the 1M2M Payload decoding JSON service
https://1m2m.eu/services/GETPAYLOAD?Human=0&PL=0102096100064f7a3c07a50300000000

```
typedef struct {
        byte MsgId;                     // Message Identification Value = 0x00
        byte Battery;                   // 0..100 == 0%..100%
        unsigned int DigIn1 :1;         // DigIn1 State
        unsigned int DigIn2 :1;         // DigIn2 State
        unsigned int Spare1 :1;         // Spare
        unsigned int Spare2 :1;         // Spare
        unsigned int Spare3 :1;         // Spare
        unsigned int Spare4 :1;         // Spare
        unsigned int Spare5 :1;         // Spare
        unsigned int PowerOut:1;        // +5V Output State
        uint8 CmdAck;                   // Sequence number of last received Command
        byte GPSFixAge;                 // bit 0..7 = Age of last GPS Fix in Minutes MsgID see above),
        byte SatCnt_HiLL;               // bit 0..4 = SatInFix, bit5 Latitude 24 bit 6,7 = Longitude 24,25
        byte Lat[3];                    // bit 0..23 = latitude bit 0..23
        byte Lon[3];                    // bit 0..23 = longitude bit 0..23
 }TaliveMsg;

typedef struct {
        byte MsgId;                     // Message Identification Value = 0x01
        unsigned int Start :1;          // Start Message
        unsigned int Move :1;           // Object Moving
        unsigned int Stop :1;           // Object Stopped
        unsigned int Vibr :1;           // Vibration Detected
        int16 Temp;                     // Temperature in 0,01 degC
        byte GPSFixAge;                 // bit 0..7 = Age of last GPS Fix in Minutes,
        byte SatCnt_HiLL;               // bit 0..4 = SatInFix, bit5 Latitude 25 bit 6,7 = Longitude 25,26
        byte Lat[3];                    // bit 0..23 = latitude bit 0..23
        byte Lon[3];                    // bit 0..23 = longitude bit 0..23
}TtrackMsg;


typedef struct {
        byte MsgId;                     // Message Identification Value = 0x02
        byte Status;                    // Content Depends on Message ID ==for future use
        int16 BaromBar;                 // Air Pressure in mBar = BaromBar +100.000)/100
        int16 Temp;                     // in 0,01 degC
        byte Humidity;                  // Relative Humidity in %
        int8 LevelX;                    // Inverse Sinus of Beam Level in Deg X-Direction -128 =
                                        // -90 Degr .. +127 = +90 Degr
        int8 LevelY;                    // Inverse Sinus of Beam Level in Deg Y-Direction -128 =
                                        // -90 Degr .. +127 = +90 Degr
        int8 LevelZ;                 // Inverse Sinus of Beam Level in Deg Z-Direction -128 =
                                        // -90 Degr .. +127 = +90 Degr
        uint8 VibAmp;                // Amplitude of Vibration Detected == Future
        uint8 VibFreq;               // Approx. Frequency of Vibration Detected in Hz
                                        // Future
}TgenSensMsg;

typedef struct {
        byte MsgId;                  // Message Identification Value = 0x03
        unsigned int GravRotAl :1;   // Gravity Rotation Detected
        unsigned int MagRot :1;      // Mag Rotation Detected
        int8 GravX;                  // Gravity in X-Direction 64 ~~ 1G
        int8 GravY;                  // Gravity in Y-Direction 64 ~~ 1G
        int8 GravZ;                  // Gravity in Z-Direction 64 ~~ 1G
        int8 MagX;                   // Magnetic Field in X-direction 10 uTesla
        int8 MagY;                   // Magnetic Field in Y-direction 10 uTesla
        int8 MagZ;                   // Magnetic Field in Z-direction 10 uTesla
}TrotMsg;

typedef struct {
        byte MsgID;                  // Message Identification Value = 0x04
        unsigned int GravRotAl :1;   // Gravity Rotation Detected
        unsigned int MagRot :1;      // Magnetic Rotation Detected
        unsigned int MotAlarm :1;    // Motion Alarm detected
        unsigned int GeoFenceAl:1;   // GeoFence Violation Detected
        unsigned int VibrAl :1;      // Vibration Alarm Detected
        unsigned int TempAlarm :1;   // Temperature alarm
        unsigned int DigIn1Al :1;    // Digin1 Alarm Detected
        unsigned int DigIn2Al :1;    // Digin2 Alarm Detected

        int16 Temp;                  // Temperature in 0,01 Celcius
        byte Hum;                    // Relative Humidity in %
        word BaromBar;               // Air Pressure in Mbar=MsgIDMsgIDBaromBar +100.000)/100)
}TalarmMsg;



Typedef struct {
        byte MsgID;                  // Message Identification Value = 0x06
        byte NumOfSensors;           // Number of 1Wire sensors currently connected
        word Temp[5];                // Store for temperatures
                                        // bit 0..11 Temperature in 0,1 Celcius + 550
                                        // Temperature range 0 = -55.0C, 1800 = 125.0C
                                        // bit 12..15 ShortID (0..15)

}T1WireTMsg;
```

```
typedef struct {
`       byte MsgID;                     // Message Identification Value = 0x08
        byte MaxdX;                     // Maximum deviation in AccelerometerX
        byte MaxdY;                     // Maximum deviation in AccelerometerY
        byte MaxdZ;                     // Maximum deviation in AccelerometerZ
        byte Max1Freq;                  // Frequency with highest amplitude
                                        // Frequency = Max1Freq * 630/53
        byte Max1Ampl;                  // Amplitude of Frequency with highest Amplitude
        byte Max2Freq;                  // Frequency with second highest amplitude
                                        // Frequency = Max2Freq * 630/53
        byte Max2Ampl;                  // Amplitude of Frequency with second highest Amplitude
        byte Max3Freq;                  // Frequency with third highest amplitude
                                        // Frequency = Max3Freq * 630/53
        byte Max3Ampl;                  // Amplitude of Frequency with third highest Amplitude
        byte vAgcVibr;                  // Gain Value Vibration Detection 0x00=2G, 0x01=4G, 0x02=8G, 0x03=16G
}TvibrMsg;

typedef struct {
        byte MsgID;                     // Message Identification Value = 0x09
        int16 Vbat;                     // Battery voltage in mV
        int16 AnalogIn1;                // AnalogIn 1 in mV
        int16 AnalogIn2;                // AnalogIn 2 in mV
        int16 Distance;                 // Distance measurement for Sonar
        int16 Analogin4;                // future use
}TanalogMsg;

typedef struct {
        byte MsgId;                     // Message Identification Value = 0x0A
        byte Status;                    // Content Depends on Message ID ==for future use
        word BaromBar;                  // Air Pressure in mBar
        int16 Temp;                     // in 0,01 degC
        byte Humidity;                  // Relative Humidity in %
        int8 GravX;                     // Accelerometer X
        int8 GravY;                     // Accelerometer Y
        int8 GravZ;                     // Accelerometer Z
        uint8 VibAmp;                   // Amplitude of Vibration Detected == Future
        uint8 VibFreq;                  // Approx. Frequency of Vibration Detected in Hz
}TgenSensGravMsg;

typedef struct {
        byte MsgId;                     // Message Identification Value = 0x0B
        byte Status;                    // Content Depends on Message ID ==for future use
        int8 MinTemp;                   // Minimum Temperature DgrC -27..+100 DgrC since last DailyRep Message
        int8 MaxTemp;                   // Maximum Temperature DgrC -27..+100 DgrC since last DailyRep Message
        byte MinHum;                    // Minimum Humidity since last DailyRep Message
        byte MaxHum;                    // Maximum Humidity since last DailyRep Message
        byte MaxBaro;                   // Maximum Baro since last DailyRep Message
        byte MinBaro;                   // Minimum Baro since last DailyRep Message
        word RunHrs;                    // Running Hours (in hours)
        word KM;                        // Distance traveled in KM
}TdailyRepMsg;

typedef struct {
        byte MsgId;                     // Message Identification Value = 0x0C
        byte Mode;                      // Current Digin1 Mode
        byte RepMode;           // Current Digin Report Mode
        dword Counter;          // Digin1 Counter
        dword RunTimer;         // Digin1 RunTimer (seconds)
        byte State;             // State of Digin1
}TdigIn1Msg;

typedef struct {
        byte MsgId;             // Message Identification Value = 0x0D
        byte Mode;              // Current Digin2 Mode
        byte RepMode;           // Current Digin Report Mode
        dword Counter;          // Digin2 Counter
        dword RunTimer;         // Digin2 RunTimer (seconds)
        byte State;             // State of Digin2
}TdigIn2Msg;

typedef struct {
        byte MsgId;                     // Message Identification Value = 0x81
        unsigned int Start :1;          // Start Message
        unsigned int Move :1;           // Object Moving
        unsigned int Stop :1;           // Object Stopped
        unsigned int Vibr :1;           // Vibration Detected
        word BeaconID1;                 // ID of beacon 1 (16 bits of DEVEUI)
        int8 RSSI1;                     // RSSI of Beacon1
        word BeaconID2;                 // ID of beacon 2 (16 bits of DEVEUI)
        int8 RSSI2;                     // RSSI of Beacon2
        word BeaconID3;                 // ID of beacon 3 (16 bits of DEVEUI)
        int8 RSSI3;                     // RSSI of Beacon3
        byte RFU1;                      // reserved for future use
} TindoorMsg;

typedef struct {
        byte MsgId;             // Message Identification Value = 0x82
        byte Impact;            // The average of the FFT values over X+Y+Z < 30Hz (experimental)
        word Shock;             // Maximum Measured Acceleration in combined direction XYZ
        word MaxDx;             // Max Acceleration in X-direction
        word MaxDy;             // Max Acceleration in Y-direction
        word MaxDz;             // Max Acceleration in Z-direction
        byte Range;             // Accelerometer Range 0 = 2G; 1= 4G; 2 = 8G; 3 = 16G
        byte NumOfShocks;       // Number of Shocks
} TshockMsg;

typedef struct {
        byte MsgId;             // Message Identification Value = 0x0E
        byte RebootReason;      // For internal use
        uint8 Profile;          // For internal use
        uint8 CmdAck;           // Last received Command
        dword 1M2MID;           // 1M2M Serial number
        byte SrcID;             // Reboot reason source file ID incl. reboot reason
        word LineNR;            // Reboot reason line number
        byte Version;           // Firmware Version bit 0..3 Low 4..7 High
}Treboot;

typedef struct {
        byte MsgId;             // Message ID = 0xF0
        byte Capacity;          // in mAh
        word MCUUsage;          // in mAh
        word GPSUsage;          // in mAh
        word LoRaUsage;         // in mAh
        word SigFoxUsage;       // in mAh
        word MiscUsage;         // in mAh
        word LEDUsage;          // in mAh
}Tbattery;
```

New since V4.64, replaces message type 0x08:

```
typedef struct {
        byte MsgID;             // Message Identification Value = 0x95
        byte MaxdX;             // Maximum deviation in AccelerometerX
        byte MaxdY;             // Maximum deviation in AccelerometerY
        byte MaxdZ;             // Maximum deviation in AccelerometerZ
        byte Max1Freq;          // Frequency with highest amplitude
        byte Max1Ampl;          // Amplitude of Frequency with highest Amplitude
        byte Max2Freq;          // Frequency with second highest amplitude
        byte Max2Ampl;          // Amplitude of Frequency with second highest Ampl.
        byte Max3Freq;          // Frequency with third highest amplitude
        byte Max3Ampl;          // Amplitude of Frequency with third highest Ampl.
        byte vMaxAgcVibr;       // Gain Value Vibration Detection
} TvibrMsg;
```

When the unit is idle the battery consumption is approximately 15 to 60 uA depending on hardware type. When moving with GPS On, battery consumption is on average 30mA.

Battery life heavily depends on the amount of time the unit is moving.

LoRa Connection behaviour of the ITalks MCS 1608 is as follows:

Send and receive LoRa PORT
The ITalks MCS 1608 sends and receives on port 1 by default. This cannot be changed.

ABP
When ABP Parameters are set the unit will use these. OTAA is never activated when ABP parameters are set.

OTAA
It tries to join a network. Since joining starts at SF7 (short range) after two retries the units increases it spreading factor. This goes on until SF12 is reached, obeying the duty cycle limits this process of joining can take a long time, after that the units stops connecting and goes to sleep for 1 hour before trying to re-connect.

Connection Lost
If the unit is connected it actively guards its connection.
After 64 transmissions without any downlink answer the units starts requesting ack's. If after 32 transmissions without an answer the unit assumes the connection is gone and increases its datarate. When the device has reached SF12 it assumes the connection is lost. The unit will go idle for 1 hour and then start to reconnect.

Version History

V1.08	Baseline		
V1.8	Final version	04-05-2016	
V1.9	BugFix	20-06-2016	• Added UTMoving == 0 -> now means no updates while moving • BugFix Command to change message formats is now ignored, choosing classic message format lead to device not sending data anymore • Battery in Alive Message now 0..254 is 0..100%
V1. A	BugFix/ Feature	13-07-2016	• Removed bug when #DTMove and #DTIdle are set at value > 127 • Removed issue with brown-out when radio is switched on in latest HW Batch (Full Only) • Removed issue with acquiring keys failing now and then • Added the enable/disable monitor command (0x1F)
V1.B	BugFix/ Feature	30-07-2016	• Improved TX Performance with LoRa • Removed bug (introduced in v1.A) where after join failure device stops sending messages
V2.0	Feature	t.b.d.	• LoRa Duty cycle is now obeyed in a more efficient manner, this allows the device to send bursts • LoRa driver is now compatible to LoRa Alliance Spec V1.01 • Digital Inputs are now supported • ABP/OTAA connection behavior changed and simplified • Yellow LED (on Full) added as visual feedback of Duty Cycle limit reached • When #Utvibration is set to 0 the device send the vibration message after the vibration has stopped.

V2.1	BugFix/ Feature	10-10-2016	• Optimize power usage by disable vibration detection by setting scaninterval to '0' in tracker configurations • Corrected a problem where the comport was disabled on trackers
V2.3	BugFix	08-11-2016	• Corrected a bug in setting the RotGrav Sensitivity (this was not correctly translated) since V2.0 • Improved the motion detection alarm. It now works as expected to send an Alarm Message after XX seconds of movement • Removed the "false" MagRot bit in the alarm message when MagnRot Sensitivity was set to "0"
V2.4	Feature	11-12-2016	• Added the SET +5V command • For LoRa SF7BW250 and FSK ADR Commands are now refused
V2.5	Feature	03-01-2017	• Content Alive Message Changed (digital states added) • Content Digin1 and Digin2 Message changed (State of input added)
V2.6	Bug	10-01-2017	• Corrected an issue whet in the Daily report message the Endian of the KM and Running hours was wrong
V2.7	Feature	29-01-2017	• APPKEY can now be changed via a downlink command and via the terminal • Unit can be "forced" to use either ABP or OTAA for LoRa Activation
2.71	Text	18-04-2017	• Corrected the barometer value from word to int16 • Corrected bits description in Alarm Message

4.0	Release	03-07-2017	• Now works with Library, for third party software development • Downlink commands via Terminal no longer require a sequence number • Changed the “system” commands • Indoor Localization Added
4.1	Release	31-07-2017	• Shock Sensor Added
4.5	Beta Release Rc1	09-10-2017	• Indoor localization Added • Added GPS HDOP check for GPS fix, thus removing fixes that are not accurate • Displays current configuration to terminal at startup (ED1000 or serial cable needed) • Now fully based on Library

4.6	BugFix/ Feature	08-01-2018	• Fixed an issue where Gravity Rotation Sensor was not working • Added a feature where an alive message can be send in a specific hour (UTC) when GPS is enabled • Fixed an issue that caused the device to stop sending messages • Corrected an error in shock calculation • Corrected an issue that caused Vibration detection to lead to instability • FFT Interval is now also obeyed when no vibration is detected. This reduces battery drain if the device is moving and no vibration is present. • Created a new method to predict remaining battery capacity • Fixed some internal issues • Added the option for the BasicTracker to use 1Wire temperature. When a 1Wire is connected this temperature is used instead of the one from the internal sensor
4.62	Feature	06-02-2018	• Added magneto tracker (activated by changes in magnetic Field) • Added battery report command and message
4.63	Feature	28-11-2018	• Added Max Temperature Alarm • Added GPSFix on Shock detect • Minor changes

4.64	Bugfix	14-03-2019	• Error fixed in FFT Datagram (0x08), the max amplitudes and their frequencies were not were not reported correctly Fixed by repairing the detector code and defining a new datagram (0x95) for FFT vibration results.
4.65		02-04-2019	• Some minor corrections in this document.
4.65	Feature	19-04-2019	• Support for deepsleep mode with swipe wakeup added
4.66	Improvement	19-05-2019	• Some default values adusted to optimize current consumption and radio duty cycle
4.67	Feature	27-05-2019	• Support for 5V output before analog measurement added (Cmd 0x34) • Defaults adjusted for Sigfox duty-cycle limits
4.68	Security improvement	28-05-2019	• Format change for downlink messages and comport parameter changing. Makes accidental parameter changes highly unlikely.