MXT-1XX Protocol Standard

Revision History

Date	Version	Description	Author
02/JUN/2009	2.0.0	Initial Version	Gustavo
04/JUN/2009	2.0.1	Add position sample and some commands.	Fei Xie; Kurt Wang
10/JUN/2009	2.0.2	1. Modify max speed configuration to UINT8 2. Modify protocol segment to UINT8 in keep alive packets	Kurt Wang
16/JUL/2009	2.0.3	1.Add anti-theft configurations 2.Add configuration for ignition off operate 3.Add buzzer control command 4.Add position sending reason to position packet	Kurt Wang
23/JUL/2009	2.0.4	Add more reasons	Kurt Wang

CONFIDENTIALITY DECLARATION

The information contained in this document is confidential and belongs to MXT Technology Co., Limited (MXT). This information cannot be used for other purposes, and cannot be disclosed outside of this organization without previous MXT's authorization, in writing. The generation of photocopies of this document is prohibited, as well as its reproduction or distribution, totally or partially, by any graphic, magnetic, optical, photographic or electronic means.

Contents

1. Introduction

This document defines the communication protocol used for development involving MXT device through GPRS/USB communication. All standard command types are listed along with mandatory or optional parameters.

2. GPRS/USB Communication Protocol

2.1 Format

This is the communication protocol frame format:

SOF	DD	MT	DEVID	DATA	CRC	EOF

NOTE 1: Gray fields are coded using byte stuffing method, described on next topic.

NOTE 2: Unless there are any explicit remarks, all values use "Little Endian" order (lowest byte or bit first).

SOF: Start of Frame, **0x01**, 1 byte. This byte starts every packet sent;

DD: Device descriptor, according to the following table:

    0xA0: MXT-100 (portable, no 2.4Ghz transceiver)

    0xA1: MXT-101 (portable, 2.4Ghz enabled)

    0xA2: MXT-150 (automotive, no 2.4Ghz transceiver)

    0xA3: MXT-151 (automotive, 2.4Ghz enabled)

MESSAGE TYPE: Message type, 1 byte, described more ahead in this document.

DEVID: Device ID, 4 bytes;

DATA: Reserved for data, size varying according with message type and destination device type. This field is coded by the byte stuffing method explained on the next topic;

CRC: The CRC16-CCITT calculation from DD until the last byte of DATA. CRC is calculated before the byte stuffing process (please notice that previous versions of MXT protocol used Big Endian for CRC);

EOF: End of Frame, **0x04**, 1 byte. This byte ends every packet sent.

2.2   Byte stuffing

The used byte stuffing method on this protocol is the insertion of one token byte (**0x10**), followed by the real byte added to 0x20. The bytes coded by this method are only: **0x01**, **0x04, 0x10, 0x11** and **0x13**.

3. Standard Command Types

The following standard frames derive from the basic structures presented on topic 2. The specific field values are presented below:

3.1 Ack

This is the general acknowledgment command. Used for confirmation of position packets and other packets when necessary.

MT: 0x02;
DATA FIELD: 2 bytes;

UINT16 CRC_received;

3.2 Nack

This is the general command reporting error. Used for reporting any error of other commands when necessary.

MT: 0x03;
DATA FIELD: 3 bytes;

UINT8 Error_type;
UINT16 CRC_received;

Error Type descriptions:

> 0x48 – CRC error (NACK_CRC)
> 0x49 – Command number error (NACK_CMD_NUM)
> 0x50 – Command parameter error (NACK_CMD_PAR)
> 0x51 – Offset error (NACK_OFFSET)
> 0x52 – Command execution error (NACK_ERRO_EXEC_CMD)
> 0x53 – Busy error (NACK_BUSY)
> 0x54 – Image not found error (NACK_IMG_NOT_FOUND)
> 0x55 – IO Module communication failure (NACK_ERRO_HCS)

3.3 Position Packets

3.3.1 Position packet format

MT: 0x31;
DATA FIELD: variable,

typedef struct
{
    u8   protocol;   //Actually only 0x08
    u8   info_groups;   //1: enable waypoint ID (u32), wpt_group (u16), accelerometer event (u8) and accelerometer value (u8) info
                                    //1: enable wireless accessory packet transmission (8 bytes long each, quantity defined by input_mask/acc_count)
                                    //1: enable GPS_SVN (u8), GPS_HDOP (u8), GPS_SNR (u8), GSM_CSQ (u8), life after reset (u16 - minutes), input voltage (u8 – volts/5) and internal temperature (s8 – degrees)
                                    //1: enable odometer (u32 - meters)
                                    //1: enable hourmeter (u32 – minutes with ignition on)
                                    //1: enable position sending reason (u32)
                                    //2: reserved

```
u16              position_count;        // auto-increment from 0 to 65535
t32_date_time    date_time;
s32              latitude;              // Ex: -19.923293 (*1000000) > -19923293
s32              longitude;             // Ex: -19.923293 (*1000000) > -19923293
t32_flags        flags;
u8               speed;                 // km/h
u8               input_mask/acc_count
```

//1: input 1 (0x00 = not masked, 0x01 = masked)

//1: input 2 (0x00 = not masked, 0x01 = masked)

//1: input 3 (0x00 = not masked, 0x01 = masked)

//1: input 4 (0x00 = not masked, 0x01 = masked)

//4: qty of 8 byte long wireless packets included at the end of position packet (maximum of 15 packets)

}MXT1XX_POS_ITEM;

NOTE 1: Each info group, when present, will add its corresponding amount of bytes to packet size. They will be found immediately after input_mask/acc_count, in the same order as flags are represented inside info_groups byte (default 0x06).

NOTE 2: Cell ID information, if set in corresponding flag of t32_flags, will add another 8 bytes, immediately after last info_group present, containing Country Code (u16), Network Code (u16), Local Area Code (u16) and Cell ID (u16).

NOTE 3: If quantity of wireless packets indicated by highest 4 bits of input_mask/acc_count is higher than zero AND corresponding flag of info_groups is set, 8 bytes will be added for each wireless packet present, according to protocol described more ahead in this document. If number of wireless packets is higher than zero, but flag for wireless packets is not set inside info_groups byte, data of wireless packets will not be added to position.

NOTE 4: the position packets sending reason define as below:
1 - device power on
2 - GPRS first attached or reattached
3 - transmission interval stopped
4 – transmission interval moving

5 – transmission interval in panic

6 - some configuration changed (change transmission interval or modify position packet content)

7 - server's requirement

8 - get GPS valid after transmission interval (on transmission interval the GPS does not fix)

9 - ignition on

10 – ignition off

11 - panic activated

12 – panic deactivated

13 – input 1 activated

14 – input 1 opened

15 – input 2 activated

16 – input 2 opened

17 – input 3 activated

18 – input 3 opened

19 - input 4 activated

20 – input 4 opened

21 – moving detect

22 – stopped detect

23 - anti-theft alarmed

24 - at least one accessories critical

25 - external power fail

26 – external power ok

27 - GPS antenna fail

28 – GPS antenna OK

29 – 2.4Ghz packet received

30 – entering sleep

31 – output 1 activated

32 – output 1 deactivated

33 – output 2 activated

34 – output 2 deactivated

35 – output 3 activated

36 – output 3 deactivated

37 – maximum speed exceeded

38 – maximum speed OK (after a exceed event)

39 – Entering waypoint

40 – Leaving waypoint

41 – Backup battery fail

42 –Backup battery OK (after fail event)

```
typedef union
{
    u32 value;
    struct
    {
        u32 seconds      :6;
        u32 minutes      :6;
        u32 hours        :5;
        u32 days         :15; // from 01/01/2000
    } info;
} t32_date_time;


typedef union
{
    u32 value;
    struct
    {
        u32 ignition            :1; // (0x00 off, 0x01 on)
        u32 panic               :1; // (0x00 no panic status, 0x01 panic status)
        u32 input1              :1; // (0x00 open, 0x01 low)
        u32 input2              :1; // (0x00 open, 0x01 low)
        u32 input3               :2; // (0x00 open, 0x01 low, 0x02 high)
        u32 input4              :1; // (0x00 open, 0x01 high)
        u32 output1             :1; // (0x00 open, 0x01 grounded)
        u32 output2             :1; // (0x00 open, 0x01 grounded)
        u32 output3             :1; // (0x00 open, 0x01 high)
        u32 direction           :3; // (0x00 north, 0x01 north east, 0x02 east, 0x03
                                    //     south east, 0x04 south, 0x05 south west,
                                    //     0x06 west, 0x07 north west)
        u32 gprs_connection     :1; // (0x00 no server connection, 0x01 connected
                                    //     to server)
        u32 voice_call          :1; // (0x00 no voice call, 0x01 voice call)
        u32 gps_fix             :1; // (0x00 valid position retrieved from memory,
                                    //     0x01 valid position actual)
        u32 gps_antenna_fail    :1; // (0x00 antenna OK, 0x01 antenna fail)
        u32 max_speed_exceeded  :1; // (0x00 speed OK, 0x01 max speed
                                    //     exceeded)
        u32 low_power           :1; // (0x00 backup battery level OK, 0x01 backup
                                    //     battery level too low)
```

```
        u32  sleep                  :1; // (0x00 GPS not sleeping, 0x01 will enter
                                        sleep and turn off GPS)
        u32 battery_charging        :1; // (0x00 not charging, 0x01 charging)
        u32 battery_fault           :1; // (0x00 internal battery OK, 0x01 battery not
                                        found)
        u32 power_fail              :1; // (0x00 power ok, 0x01 no external power)
        u32 waypoint_restricted     :1; // (0x00 off, 0x01 on)
        u32 waypoint_entering       :1; // (entering waypoint)
        u32 waypoint_leaving        :1; // (leaving waypoint)
        u32 GSM_jamming             :1; // (0x00 no jamming, 0x01 jamming)
        u32 anti_theft_status       :2; // (0x00 disarmed, 0x01 armed, 0x02
                                        suspended , 0x03 alarmed)
        u32 accessory_missing       :1; //at least one expected accessory not found
        u32 moving_status           :1; // (0x00 stopped, 0x01 moving)
        u32 cell_id_present         :1; // (CC, NC, LC, ID - 8 bytes – will be present)
    } info;
} t32_flags;
```

3.3.2 Accessories information

If wireless packets from accessories are present inside position packet, their data must be decoded according to the following protocol.

Each packet has same length (8 bytes). The structure is as below:

ID	4 bytes
Device type	1 byte
Protocol	1 byte
Data	2 bytes

Device types actually defined are:

0x20: WT100 (wireless watch)
0x21: WT110 (wireless button/tag)
0x22: WT111 (wireless temperature sensor)
0x23: WT112 (wireless switch sensor)
0x24: WT200 (wireless anti-theft/relay)
0x25: WT300 (wireless LCD)

Actually only protocol 0x20 exists.

Data can have the following formats based on *Device type*:

0x20 - WT100 – wireless watch:

Battery level:	3bits	percentage of current voltage (0x00<12.5%, 0x07>87.5%)
Button status:	2bits	00: no pressed, 01: short pressed, 10: long pressed, 11: short pressed 3 times in 5 seconds
Touch sensor status:	1bit	0: normal, 1: no body
Wrist loop 1 event:	1bit	0: normal, 1: open
Wrist loop 2 event:	1bit	0: normal, 1: open
Reserved:	1bit	
Internal temperature:	7bits	-40~+88 (0x00=-40 or lower, 1 degree per unit, 0x7F=+88 or higher)

0x21 - WT110 – wireless button/tag:

Battery level:	3bits	percentage of current voltage (0x00<12.5%, 0x07>87.5%)
Button status:	2bits	00: no pressed, 01: short pressed, 10: long pressed, 11: short pressed 3 times in 5 seconds
Reserved:	4bits	
Internal temperature:	7bits	-40~+88 (0x00=-40 or lower, 1 degree per unit, 0x7F=+88 or higher)

0x22 - WT111 – wireless temperature sensor:

Battery level:	3bits	percentage of current voltage (0x00<12.5%, 0x07>87.5%)
Button status:	2bits	00: no pressed, 01: short pressed, 10: long pressed, 11: short pressed 3 times in 5 seconds
Under-temperature:	1bit	0: normal, 1: bellow -30
Over-temperature:	1bit	0: normal, 1: above +70
Reserved:	1bit	
External temperature:	8bits	-128~127

0x23 - WT112 – wireless switch sensor:

Battery level:	3bits	percentage of current voltage (0x00<12.5%, 0x07>87.5%)
Button status:	2bits	00: no pressed, 01: short pressed, 10: long

pressed,

11: short pressed 3 times in 5 seconds

Loop 1:	1bit	0: closed,　1: opened
Loop 2:	1bit	0: closed,　1: opened
Reserved:	2bits	
Internal temperature:	7bits	-40~+88 (0x00=-40 or lower, 1 degree per unit,

0x7F=+88 or higher)

0x24 - WT200 – wireless anti-theft/relay:

Ignition status:	1bit	0: off,　1: on
Button status:	2bits	00: no pressed, 01: short pressed, 10: long pressed,
Doors status:	1bit	0: closed, 1: opened
Mode:	3bits	000: normal, 001: parking, 010: armed,

011: violated antitheft, 100: sleep

High-side output:	2bits	00: not activated, 01: activated
		10: open load, 11: over-temperature/current
Low-side output 1:	2bits	00: not activated, 01: activated
		10: reserved, 11: over-temperature/current
Low-side output 2:	2bits	00: not activated, 01: activated
		10: reserved, 11: over-temperature/current
Under-temperature:	1bit	0: normal,　1: bellow -30
Over-temperature:	1bit	0: normal,　1: above +70
Reserved:	1bit	

0x25 – WT300 – wireless lcd:

OK Button status:	2bits	00: no pressed, 01: short pressed, 10: long pressed,

11: short pressed 3 times in 5 seconds

Under-temperature:	1bit	0: normal,　1: bellow -30
Over-temperature:	1bit	0: normal,　1: above +70
Event code:	2bits	0x00=no event, 0x01=accept received message,

0x02=reject received message,

0x03=send pre-configured message)

Message group:	2bits	(group of message sent)
Message code:	8bits	(code of message accepted, rejected or sent)

3.3.3 Reply from server

When MXT-1xx sends a position packet to server, server must send an ACK to device in order to confirm that packet was received successfully.

3.3.4 Examples

Position packet with no present group and no cell information:

Hex data:		ASCII
31	position indication MT	
08	protocol	8
00	info_group	No group info added
CD 24	position_count	9421
DD 37 87 19	date_time	2008-12-11 19:31:29
10 19 62 02	latitude	39.983376
A0 4B EF 06	longitude	116.345760
02 AC 18 00	flags	panic set gprs connected, Direction is south east Gps fixed, sleep set, Charging,
00	speed	0
02	input_mask/acc_count	input 2 is masked and no accessoires

Position packet with default group and cell information:

Hex data:		ASCII
31	position indication MT	
08	protocol	8
06	info_group	6
9C 24	position_count	9372
BA F1 84 19	date_time	2008-12-10 15:06:58
10 19 62 02	latitude	39.983376
A0 4B EF 06	longitude	116.345760
02 AC 18 80	flags	panic set gprs connected, Direction is south east Gps fixed, sleep set, Charging, Cell id present
00	speed	0

20	input_mask/acc_count	no inputs masked and have 2 accessoires
01 00 00 00	accessory 1 ID	1
23	device type	WT111
20	protocol	
07 1A	Data	full battery and no button Pressed, and temperature is 26
02 00 00 00	accessory 2 ID	2
25	device type	WT200
20	protocol	
01 08	Data	ignition on and in normal Mode, button not pressed Door is closed, low side 2 is activated.

3.3.5 Command from server

Server can send this command to MXT device to require current position.

MT: 0x32
DATA FIELD: NONE

3.4 Read configuration

MT: 0x32;
DATA FIELD: 1 byte;

UINT8 mode;
The mode descriptions:
    0x1 - CRM_DEV_INFO, read device information
    0x2 - CRM_NET_ATTRIB, read data connection configurations
    0x3 - CRM_IP_ADDRESS, read IP configurations
    0x4 - CRM_REPORT_INTERVAL, read position packet reporting configurations
    0x5 - CRM_GSR, read accelerometer configurations
    0x6 - CRM_GPS, read GPS configurations
    0x7 - CRM_SMS, read SMS configurations
    0x8 - CRM_ZIG_INFO, read 2.4Ghz configurations

0x9 - CRM_OTHERS, read other configurations

0xC - CRM_PANIC_MODE, read panic mode

0xD - CRM_ZIG_INFO_EXT, read extend 2.4Ghz information

Reply format is:

MT:0x32

DATA FIELD: variable, please see the following:

3.4.1 Reply of CRM_DEV_INFO

UINT8 mode;   //0x1

UINT32 deviceID;

UINT8   sw_version[4];   //if the version is 1.23 then the byte in sw_version is "0123"

3.4.2 Reply of CRM_NET_ATTRIB

UINT8 mode;   //0x2

UINT8 connectionType; //0 is UDP 1 is TCP

UINT16 keepAliveTimer;   //keep alive timer for UDP

UINT8 apnSize;

UINT8* Apn;   apnSIze bytes

UINT8 userSize;

UINT8* User;   userSIze bytes

UINT8 pswSize;

UINT8* Password;   pswSIze bytes

3.4.3 Reply of CRM_IP_ADDRESS

UINT8 mode;   //0x3

UINT8 primaryIPaddrSize;

UINT8* primaryIPaddr;   primaryIPaddrSize bytes

UINT8 secondaryIPaddrSize;

UINT8* secondaryIPaddr;   secondaryIPaddrSize bytes

UINT16 primaryIPport;

UINT16 secondaryIPport

3.4.4 Reply of CRM_REPORT_INTERVAL

UINT8 mode;   //0x4

UINT16 timerOfIgnitionOff;   //timer for reporting position packets when device is stopped.

UINT16 timerOfMovement;   //timer for reporting position packets when device is moving

UINT16 timerOfPanic   //timer for reporting position packets when device is in panic status

UINT8 reSendAttempts;   //attempts to send again when sending packet failed or ACK from server was not received

UINT16 reSendTimeout;   //timer of waiting for ACK from server after a packet is sent

UINT8 infoGroup;

3.4.5 Reply of CRM_GSR

UINT8 mode;   //0x5

UINT8 sendImmediately;   //1 indicates that when device detects a change in stopped or moving status, a position packet is sent immediately.

UINT16 debMoving;   //debounce timer for checking if device is moving

UINT16 debStopped;   //debounce timer for checking if device is stopped

UINT16 detectMoving;   //timer to detect the device new state when it is moving

3.4.6 Reply of CRM_GPS

UINT8 mode;   0x6

UINT16 keepWorkingTimer   //timer for keeping GPS working after position packet is sent

UINT16 unFixTimeout   //max timer for waiting for GPS fix

UINT16 unfixColdTimeout   //max timer for waiting for GPS fix when cold started

UINT16 openBfTransMove   //timer to turn on GPS before transmission when moving

UINT16 openBfTransStop   //timer to turn on GPS before transmission when

stopped

3.4.7 Reply of CRM_SMS

UINT8 mode;   //0x7
UINT8 sendMode;   //mode of sending position packets through SMS
UINT8 aliasName[32];   //alias of the device, this alias will be included in SMS
                               header and sent to server.
UINT8 destination[16];   //destination phone number for device to send short
                               message to
UINT8 allowNumberMode;   //set allow number to 0(any number), or 1(only
                               destination number)

3.4.8 Reply of CRM_ZIG_INFO

UINT8 mode;   //0x8
UINT8 masterSleepEnable;   //1 is enabled, 0 is disabled
UINT8 keepAliveInterval;   //timer for communication between master and
                               accessory, in minutes
UINT8 keepAliveDuration;   //timer for master to wait for reports from accessory
UINT8 accRetry;   //attempts for accessory to send report.
UINT16 accRxTimeout   //max timer of accessory for waiting master's ACK
UINT8 accEncryptKey[16]   //encrypt key for every packet transmitted between
                               2.4Ghz devices

3.4.9 Reply of CRM_OTHERS

UINT8 mode;   //0x9
UINT8 ledEnable;   //1 is enabled, 0 is disabled
UINT8 charingOnly   //1 is enabled, 0 is disabled
UINT8 input1Enable   //1 is enabled, 0 is disabled
UINT8 input2Enable   //1 is enabled, 0 is disabled
UINT8 input3Enable   //1 is enabled, 0 is disabled
UINT8 lowPowerAlert   //1 is enabled, 0 is disabled
UINT8 timezone   //0~25, 0 is -12, 12 is 0, 25 is +13
UINT8 maxSpeedLimit   //max speed limit
UINT32 odometer   //initial odometer in meters

UINT8 cellInfo   //1 is "add cell ID to all position packets", 0 is "add only when there is no actual GPS position"
UINT8 dtmfPassword[8]   //DTMF password
UINT8 input4Enable   //1 is enabled, 0 is disabled
UINT32 hourmeter   // minutes with ignition on
UINT8 bSmartOutput2   // 1 activate smart output2, 0 deactivate
UINT8 antiTheftEnable   // 1 activate, 0 deactivate
UINT8 doorDetect   // 1 enabled, 0 disabled
UINT8 localParkingMode   // 1 enabled, 0 disabled
UINT8 bDisableZigbee   // 1 enabled, 0 disabled zigbee when ignition off
UINT8 bOdometerCalc   // 1 enabled, 0 disabled calculate odometer when ignition off

3.4.10 Reply of CRM_PANIC_MODE

UINT8 mode;   //0xC
UINT8 panicMode;   // 0 no trigger, 1 key release trigger, 2 long press key trigger

3.4.11 Reply of CRM_ZIG_INFO_EXT

UINT8 mode;   0xD

UINT8 apPowerLevel;   1-16，the zigbee power level

UINT8 apRadioChan;   1-4,   Zigbee radio channel
UINT8 edRelinkInterval;   the sync device's link retry interval in minutes
UINT8 edCmdInterval; …………the sync device's command sending interval in hundred million seconds
UINT8 apRevAllBc;   1 is receiving all broadcast message; 0 is only receiving the message in the list
UINT16   edTimeRetryMax;   the max retry duration time of the device in minutes
UINT32   linkToken;   link token(RFU)

3.5 Set configuration

Frame containing a list of specific binary parameters, as listed on this topic. The reply must be an ACK or NACK.
All configurations description, please see chapter 3.4
MT: 0x33
DATA FIELD:

ID (1 BYTE)	PARAMETER (VARIABLE)

Note: this DATA FIELD can be composed by more than one parameter, as long as the whole command is limited to 1K.

ID	PARAMETER	ID	PARAMETER	ID	PARAMETER	…

Therefore the whole parameter must be sent even when it contains null fields.

Set ID descriptions:

0x3 - (SPC_DTMF_PWD): set DTMF password
UINT8 pwdSize :   the size of the password
UINT8* password: pwdSize bytes, the DTMF password

0x4 - (SPC_CONNECT_MODE): set type of connection
UINT8 type;

0x5 - (SPC_APN): set APN user and password
UINT8 apnSize   : apn size
UINT8 userSize : user name size
UINT8 pwdSize   : password size
UINT8* apn   : apnSize bytes, the apn
UINT8* user   : userSize bytes, the user name
UINT8* password : pwdSize bytes, the password

0x6 - (SPC_IP_ADDR): set IP and port
UINT8 ipIndex   : 0 is primary ip, 1 is secondary ip
UINT8 addrSize :   address size
UINT16 port   : the port of the IP
UINT8* addr   : addrSize bytes, the IP address

0x7 - (SPC_KEEP_ALIVE_TIMER): set keep alive timer

UINT16 keepAliveTimer

0x8 - (SPC_RI_STOPPED): set report time with ignition off
UINT16 timerOfIgnitionOff;

0x9 - (SPC_RI_MOVING): set report time with movement
UINT16 timerOfMovement;

0xA - (SPC_RI_PANIC): set report time with panic
UINT16 timerOfPanic;

0xB - (SPC_RI_RESEND) set resend attempts and timeout
UINT8 attempts
UINT16 timeout

0xC - (SPC_GSR_DEB_MOVING): set debounce timer of moving
UINT16 timer;

0xD - (SPC_GSR_DEB_STOPPED): set debounce timer of stopped
UINT16 timer;

0xE - (SPC_GSR_DETECT): set the timer for detecting when moving
UINT16 timer;

0xF - (SPC_GSR_REPORT): set send immediately or not
UINT8 sendImmediately;

0x10 - (SPC_GPS_KEEP_WORKING): set GPS keep working timer
UINT16 timer;

0x11 - (SPC_GPS_UNFIX_TIMEOUT): set GPS timeout for fix
UINT16 timer;

0x12 - (SPC_GPS_COLDSTART_UNFIX_TIMEOUT): set GPS timeout for fix
when cold started

UINT16 timer;

0x13 - (SPC_GPS_OPEN_BEF_TRANS): set GPS on before transmission
UINT16 MovingTimer;
UINT16 StoppedTimer;

0x14 - (SPC_SMS_ALIAS): set SMS alias
UINT8 aliasSize :   the size of the alias
UINT8* alias   : aliasSize bytes, the alias

0x15 - (SPC_SMS_DESTINATION) set SMS destination number
UINT8 destSize :   the size of the destination number
UINT8* destination   : destSize bytes, the destination number

0x16 - (SPC_SMS_SEND_MODE): set send SMS mode
UINT8 smsMode;
UINT8 allowNumberMode;

0x17 - (SPC_ZIG_AP_SLEEP): set master accessory can sleep or not
UINT8 masterSleepEnable;

0x18 - (SPC_ZIG_KA_INTERVAL): set keep alive interval
UINT8 timer;

0x19 - (SPC_ZIG_KA_DURATION): set keep alive duration
UINT8 timer;

0x1A - (SPC_ZIG_ACC_RETRY): set accessory attempts for sending reports
UINT8 accRetry;

0x1B - (SPC_ZIG_ACC_RX_TIMEOUT): set accessory RX timeout for waiting for
                                            master response

UINT16 timeout;

0x1C - (SPC_ZIG_ACC_ENCRYPT_KEY): set the encrypt key of data.
UINT8* accEncryptKey : 16bytes, the accessory encrypt data key

0x1D - (SPC_LED_ENABLE): set enable led working or not
UINT8 enable;

0x1E - (SPC_CHARGING_ONLY): set start or stop charging only mode
UINT8 chargingOnly;   1 is started, 0 is stopped

0x1F - (SPC_INPUT_ENABLE)

UINT8 inputIndex : the index of the input port (1~4)
UINT8 enable   : enable or disable

0x20 - (SPC_OUPUT_ACTIVATE)
UINT8 outputIndex : the index of the output port (1~3)
UINT8 activate   : activate or deactivate

0x21 - (SPC_TIME_ZONE): set the time zone
UINT8 timezone;

0x22 - (SPC_MAX_SPEED_LIMIT): set max speed limit
UINT8 maxSpeed;

0x23 - (SPC_DEACTIVATE_PANIC): deactivate panic status
No parameters

0x24 - (SPC_CLEAR_OLD_REPORT): clear unsent position packets log.
NO parameters

0x25 - (SPC_SET_ODOMETER): set initial odometer value.
UINT32 odometer

0x26 - (SPC_CELL_INFO_PRESET): set if cell info is added to position packet or
                                    not.
UINT8 cellInfo

0x27 - (SPC_LOW_POWER_ALERT): enable low power alert or not
UINT8 enable;

0x28 - (SPC_PANIC_MODE): set trigger panic's mode
UINT8 mode;

0x29 – (SPC_ZIG_AP_POWER_LEVEL): set the zigbee's power level, value is
                                    1-16
UINT8 apPowerLevel

0x2A – (SPC_ZIG_AP_RADIO_CHN): set the Zigbee's radio channel, value is
                                    1-4
UINT8 apRadioChan

0x2B – (SPC_ZIG_ACC_RELINK_INT): the accessories' link retry interval in minutes

UINT8 edRelinkInterval

0x2C – (SPC_ZIG_ACC_CMD_INT): the sync device's command sending interval in hundred million seconds

UINT8 edCmdInterval

0x2D – (SPC_ZIG_ACC_RETRY_MAX_INT): the max retry duration time of the device in minutes

UINT16   edTimeRetryMax

0x2E – (SPC_ZIG_AP_REV_ALL_BC): 1 is receiving all broadcast message; 0 is only receiving the message in the list

UINT8 apRevAllBc

0x2F – (SPC_SET_HOURMETER): set initial hourmeter

UINT32 hourmeter

0x30 – (SPC_SET_POSITION_GROUP): set information group of the position packets.

UINT8 infoGroup

0x31 – (SPC_SET_SMART_OUTPUT2): enable or disable smart output port2.

UINT8 enabled

0x32 – (SPC_OUTPUT_TEST): start all output test.

No parameters

0x33 – (SPC_OUTPUT_MARCO): start output marco sequence.

No parameters

0x34 – (SPC_SET_ANTI_THEFT): enable or disable anti-theft function.

UINT8 enabled

0x35 – (SPC_SET_BTN_PARKING): enable or disable local parking mode

UINT8 enabled

0x36 – (SPC_SET_DOOR_DETECT): enabled or disable door detect

UINT8 enabled

0x37 – (SPC_DISARMED_MODE): let device go to normal mode.
No parameters

0x38 – (SPC_PARKING_MODE): let device go to parking mode or exit.
UINT8 enter

0x39 – (SPC_ENABLE_ZIG): enable or disable zigbee communication when ignition off.
UINT8 enable

0x3A – (SPC_CALC_ODOMETER): enable or disable calculate odometer when ignition off.
UINT8 enable

0x3B – (SPC_BUZZER_CTRL): control the buzzer.
UINT8 circle
UINT8 onTime
UINT8 offTime

0xFF – (SPC_RESET_TO_DEFAULT): reset all configurations to default value.
NO parameters

3.6 Power off

Inform power off event;
MT: 0x34;
DATA FIELD: 1 bytes;
UINT8 delay; (1 ~255) indicates device delay in multiples of 100 milliseconds to power off.

The reply must be an ACK or NACK

3.7 Reset

Inform reset event.
MT: 0x35;
DATA FIELD: 1 bytes;
UINT8 delay; (1 ~255) indicates device delay in multiples of 100 milliseconds to reset.
The reply must be an ACK or NACK.

3.8 GPRS pause

Inform GPRS connection pause, and waits for voice call
MT: 0x36;
DATA FIELD: variable

Waiting time (1 BYTE)	Number Size (1 BYTE)	Phone number

The waiting time is in minutes (1 ~ 255)

The reply must be an ACK or NACK

3.9 GPRS resume

Device sends this command to server when the voice call has dropped or the waiting timer is expired.
MT: 0x37;
DATA FIELD: none

Server does not need to reply this command.

3.10 Waypoint

Add, edit, delete, modify priority and set route filter.

MT: 0x38
DATA FIELD:

ID (1 BYTE)	PARAMETER (VARIABLE)

Set ID descriptions:

0x0 - (WPO_GET_INFO): get information about waypoints
No parameter.

REPLY:
UINT8 operate   //0x0
UINT16 count   //number of waypoints stored in device
UINT16 group   //group of current searching

0x1 - (WPO_NEW): add a new waypoint to device
WAY_POINT_ITEM* pItem,   //waypoint item structure,

REPLY:
ACK or NACK

0x2 - (WPO_READ): read waypoint item
UINT16 index.   //Index of the waypoint
REPLY:
UINT8 operate   //0x2
WAY_POINT_ITEM* pItem

0x3 - (WPO_EDIT): edit an existent waypoint
UINT16 index.
WAY_POINT_ITEM* pItem,

REPLY:
ACK or NACK

0x4 - (WPO_DEL): delete an existent waypoint
UINT16 index.

REPLY:
ACK or NACK

0x5 - (WPO_DEL_ALL): delete all waypoints
No parameter.

REPLY:

ACK or NACK

0x6 - (WPO_MOVETO): modify current waypoint's priority
UINT16 index_current
UINT16 index_moveto.

REPLY:
ACK or NACK

0x7 - (WPO_FILTER): set group want to search with.
UINT16 group.

REPLY:
ACK or NACK

The structure's definition:
typedef union
{
    u32 value;
    struct
    {
        u32 panic   : 1;   0 do nothing, 1 set panic status
        u32 output1   : 2;   0 do nothing, 1 activate, 2 deactivate
        u32 output2   : 2;   0 do nothing, 1 activate, 2 deactivate
        u32 set_forbidden_point   : 1;   0 clear forbidden point,
                                            1 set forbidden point
        u32 buzzer_led_mode   : 2;   0 do nothing,
                                            1 trigger mode 1
                                          2 trigger mode 2
                                          3 trigger mode 3
        u32 transmission_mode   : 2;   0 do nothing,
                                          1 set GPRS transmission with ignition off
                                          2 set GPRS transmission with ignition on
        u32 transmission_nogprs   : 1;   0 do nothing,
                                          1 send one position through SMS if GPRS is not available
        u32 zig_position   : 1;   0 do nothing,
                                          1 transmit position through 2.4Ghz once
        u32 dummy1   : 4;

```
            u32 direction                    : 3;   same as position protocol
            u32 init_hour                    : 5;   minimum hour of day to validate point, 0
                                                    for don't care
            u32 end_hour                     : 5;    maximum hour of day to validate point,
                                                    0 for don't care
            u32 dummy2                       : 3;
        } info;
} t32_actions_validation;


typedef struct
{
    u32                     id_waypoint;
    u16                     wpt_group;
    u8                      min_speed;      minimum speed to validate the point -
                                                default 0x00
    u8                       max_speed;      speed above which speed excess
                                                flag will be set - default 0xFF
    t32_actions_validation      actions_conditions;
    s32                     latitude_1;       latitude of top left corner
    s32                     longitude_1;     longitude of top left corner
    s32                     latitude_2;       latitude of bottom right corner
    s32                     longitude_2;      longitude of bottom right corner
}WAY_POINT_ITEM;
```

3.11 Firmware download start

Start to download new firmware to device, after this command server must send download file frame to device. The reply after each package must be an ACK or NACK.

MT: 0x39
DATA FIELD: 5 bytes

UINT32 fileSize;   // the firmware file size.
UINT8 firmwareMode;   // the firmware mode, defined as below.

1 - UPDATE_MODE_G24HZ
2 - UPDATE_MODE_APP
4 - UPDATE_MODE_MODEM

3.12 Firmware download file

Frame used to send a firmware file. Each package should be limited to 1Kb.

MT: 0x3A
DATA FIELD:

OFFSET (4 BYTES)	SIZE (2 BYTES)	DATA STREAM

The reply after each package must be an ACK or NACK.

3.13 Firmware download end

Upgrades a firmware file loaded on the memory of the processor and must be sent after the Firmware File parameter. The reply must be an ACK or NACK.

MT: 0x3B
DATA FIELD: 3 bytes

UINT16 CRC_firmware;
UINT8 modemAndApp;   // if the application and modem firmware updated at same time, it must be set to 1, otherwise set to 0

3.14 2.4Ghz Accessory Operation

This command is used to let MXT start searching 2.4Ghz accessories or allows accessory to be joined.

MT: 0x3D
DATA FIELD:

ID (1 BYTE)	PARAMETER (VARIABLE)

Operate ID descriptions:

0x0 (AOP_SEARCH) start to search accessories

No parameter.

REPLY:

ACK or NACK

After searching timer has expired, will send following info to server:

UINT8 mode = 0x0

UINT8 addCount;   //searched accessories count.

zig_addr_t*   addList;   //searched accessories address and type.

0x1 (AOP_SEARCH_CANCEL) stop to search accessories

No parameter.

REPLY:

ACK or NACK

0x2 (AOP_SET) set accessories

UINT8 addCount;   //add accessories to list, if it is 0 means delete all accessories.

zig_addr_t*   addList;   //accessories address and type.

REPLY:

ACK or NACK

0x3 (AOP_GET) get accessories of MXT

No parameter.

REPLY:

UINT8 mode = 0x03

UINT8 addCount;   accessories count.

zig_addr_t*   addList;   accessories address and type.

0x4 (AOP_CFG_START) send accessory configuration data

ZIG_CFG_START startInfo;

typedef struct

```
{
    u8      addr[4];
    u8      aesKey[16];
}ZIG_CFG_START;
```

REPLY:
    ACK or NACK

0x5 (AOP_CFG_RESULT) send configuration result
ZIG_CFG_RESULT startInfo;
typedef struct
```
{
    u8      cfgOpt[2]; //RFU
    u8      cfgResult; //1:cfg ok 0:cfg cancel
    u8      newDataLen; //the length of the new configuration
    u8      targetAddr[4]; //used for the case to change a device's addr
    u8      newData[200]; //new configuration data
}ZIG_CFG_RESULT;
```
REPLY:
    ACK or NACK

0x6 (AOP_CFG_ORG_DATA) Send the original configuration data of the accessories
    U8 len;
    U8* data;
    REPLY:
        ACK or NACK

The definition of zig_ed_info_t:
typedef struct
```
{
    u8   type;
    u8   linkId;
    u8   listType; //0 link;1 bc
    u8   dummy;
    u8   addr[NET_ADDR_SIZE];
    u8   edPw[4];
} zig_ed_info_t;
```

Definition of accessory's type:
0x20: WT100 (wireless watch)
0x21: WT110 (wireless button/tag)
0x22: WT111 (wireless temperature sensor)
0x23: WT112 (wireless switch sensor)
0x24: WT200 (wireless anti-theft/relay)
0x25: WT300 (wireless LCD)
0xFF: unknown accessories

3.15 Keep alive packet

In UDP connection mode, device will send this packet to server when keep alive timer is expired in order to update device's IP and port. And server has no need to reply it.

MT: 0x28;
DATA FIELD: 1 byte;

UINT8 protocol;

4. Reference

None.