MEDIATEK

MT7681

500	30
5K	28
10K	25
6	10%

http://www.ai-thinker.com

MT7681
802.11 b/g/n single chip
Preliminary datasheet

Version: 0.00

Release date: 2014-01-08

IoT Control protocol

Version: v0.2

Release date: 2014-3-20

Document Revision History

Revision	Date	Author	Description
0.1	2013-12-06	Arron	Initial version
0.2	2014-03-20	Arron	Add ctrl password command

1 说明

The MEDIATEK Production Tool is used for MP production test..

2 Control protocol design

2.1 Header format

2.1.1 Magic

IoT Control protocol is an application level protocol. So a specific identifier to identify it is much important to us.

We can use a magic number to identify the package is IoT Control protocol package. Magic number is an four octets number, and we use 0x76814350 as IoT Control protocol Magic number. (0x7681: our first IoT chip, 0x43 C: Control, 0x50 P: Protocol)

2.1.2 Receive MAC

This filed is six octets. If this filed values is set to FF:FF:FF:FF:FF:FF means this message will deal with by all client device. Others will deal with by the client that match the MAC.

2.1.3 Send MAC

This filed is six octets. This filed does not set to FF:FF:FF:FF:FF:FF. It will set to the MAC address of client MAC address or Server MAC address.

2.1.4 Session ID

Server will assigned all client an unique ID. After client be assigned an ID, then it will only deal with the ID match message.

2.1.5 Sequence number

This filed is 12 bits. This filed used for one request-response transaction when client communication with server.

2.1.6 Flag

Flag filed is four bits. Flag filed defined in table 5.

Table 5 Flag definitions

bit	description
0	Retransmission flag 1: retry frame
1	Urgency flag 1: urgency frame
2	Reserved

3	Reserved

Reserved

2.1.7 Type

Type filed is four bits and it defines the control type of IoT control protocol. Type filed defined in table 1.

Table 1 Type definitions

Type	description
0000	Management command
0001	Function command
0010	Unidirection notification
0011 – 1111	reserved

2.1.8 Subtype

Subtype filed is one octets. This filed depends on type filed

2.1.8.1 Management command

Management command subtype filed defined in table 3.

Table 3 Management command subtype definitions

subtype	description
1	Query client capability request
2	Query client capability response
3	Server control client offline request
4	Server control client offline response
5	Control password request
6	Control password response
7	Control password confirm
8	Client report leave response
9	Keep alive request
10	Keep alive response
Others	reserved

2.1.8.2 Function command

Function command subtype filed defined in table 4.

Table 4 Function command subtype definitions

subtype	description
1	GPIO Set request
2	GPIO Set response
3	GPIO Get request

4	GPIO Get response
5	UART Set request
6	UART Set response
7	UART Get request
8	UART Get response
9	PWM Set request
10	PWM Set response
11	PWM Get request
12	PWM Get response
Others	reserved

2.1.8.3 Unidirection notification

Unidirection notification subtype filed defined in table 2.

Table 2 Unidirection notification subtype defines

subtype	description
0	client notify message
others	reserved

Reserved

2.2 Data format

All data format can use TLV types. Format as follows:

Type	Length	value

Data format are defined to have a common general format consisting of a 1 octet type filed, a 2 octet length filed and variable-length specific information.

The data types defined in table 6.

Table 6 Data types definitions

Data types	description
0	Status
1	Client device capability
2	UART information
3	GPIO information
4	PWM information
5	Control Password
Others	Reserved

2.2.1 Status

Status is used to signal status information in the response frame of a request-response transaction. Status format is shown in table 7.

Table 6 Data types definitions

Filed	Size(octets)	Value	Description
Type	2	0	
Length	2	1	
Status code	1	0 – 255	Defines in table 8

Table 8 Status code definitions

Status code	description
0	success
1	Command fail
Others	Reserved

2.2.2 Client device capability

Client basic information is defines as follows:

Filed	Size(octets)	Value	Description
Type	2	1	
Length	2	variable	
IoT descriptor	variable	-	Defines in table 10

Table 10 client descriptor code definitions

Filed	Size(octets)	Value	Description
Vendor Name Len	2	-	厂商名称长度
Vendor Name	32	-	厂商名称字符串(1-32)
Product Type Len	2	-	产品型号长度
Product Type	32	-	产品型号字符串(1-32)
Product Name Len	2	-	产品名称长度
Product Name	32	-	产品名称字符串(1-32)
Flag	2	-	Bit0 :1 UART enable :0 UART disable Others Reserved
GPIO Information	6	-	See Table-15
WMT bitmap	4	-	WMT bitmap

2.2.3 UART information

Table 12 UART information format definitions

Filed	Size(octets)	Value	Description
Type	2	3	
Length	2	1-1024	For Layer 2 does not support fragmentation
UART data	variable	-	uart data

2.2.4 GPIO information

Table 14 GPIO information format definitions

Filed	Size(octets)	Value	Description
Type	2	4	
Length	2	8	
GPIO descriptor	8	-	Defines in table 15

Table 15 GPIO descriptor code definitions

Filed	Size(octets)	Value	Description
GPIO Number	4	-	BitList of GPIO. Bit0:0 GPIO0 not support 1 GPIO0 support
GPIO bitmap	4	-	Bitmap is two octets, it can contain GPIO 0 – 31. 0: GPIO is low level 1: GPIO is high level

2.2.5 PWM information

Table 16 PWM information format definitions

Filed	Size(octets)	Value	Description
Type	2	5	
Length	2	8	
RGB descriptor	8	-	Defines in table 17

Table 17 RGB descriptor code definitions

Filed	Size(octets)	Value	Description
length	2	-	
Red	2	0 - 255	Red level
Green	2	0 - 255	Green level
Blue	2	0 - 255	Blue level

2.2.6 Control Password information

Table 18 control password information format definitions

Filed	Size(octets)	Value	Description
Type	2	6	

Length	2	4	
password	4	variable	

3 Protocol detail

3.1 IoT identifier description

Each device in IoT network has an only identifier. Identifier is an 2 octets number, two values is for special meaning:

0x00: server ID
0xFF: client are not initialize
        all client can Receive

Identifier	values	description
Receive Identifier	0xFF	Send to all client, but not contain server
Send identifier	0xFF	Client are not initialize, only uninitialized client can use 0xFF in send identifier filled.
Receive Identifier	0x00	Client send the message to server, then filled receive identifier to 0x00
Send identifier	0x00	Server send the message will filled send identifier filled to 0x00
others	0x01 – 0xFE	Dispatched by server to each client.

All clients connected to the server, server will dispatch an uniqueness ID to the each client.

3.2 Keep alive mechanism

Keep alive message interval KEEP_ALIVE_INTERVAL_DEFAULT default is 500ms. Server will set keep alive timeout to 2 * KEEP_ALIVE_INTERVAL_DEFAULT, default is 1s.

When client send a keep alive request to the server, it expect received keep alive response from server. If client does not received the response in 100 ms, client will change the keep alive interval from default value to 100ms. Once client received keep alive response from server, then it set the interval to default values.

If server does not received client keep alive request message in 2* KEEP_ALIVE_INTERVAL_DEFAULT, then server will consider this client has leave network. Server will clear the record of this client.

If client does not received server keep alive response from server in 2* KEEP_ALIVE_INTERVAL_DEFAULT, client will consider server has out of network. And the client will set it status to un-init status.

3.3 Retransmission mechanism

4 Message format

4.1 Management command

4.1.1 Query client capability request

Query client capability request is use by server send to client to query all connected client capability. Package format all contain header.

Header format as follow:

4 octets	6 octets	6 octets	4 octets	2 octets	4 bits	4 bits	8 bits
Magic	Receive MAC	Send MAC	SessionID	Sequence number	flag	type	subtype
0x76814350	FF:FF:FF:FF:FF:FF	Server MAC	0xFFFFFFFF	0x0001	0000	0000	0000 0001

Magic ：0x76814350 IoT control protocol Magic, all message must use this Magic number.

Receive MAC: Set to FF:FF:FF:FF:FF:FF means all client received this quest must reply this message.

Send MAC: will set to sever MAC address

Session ID: reserved

Sequence number: one request-response transaction

Flag: reserved

Type: 0000 is management command

Subtype：0000 0001 is Query client capability request

4.1.2 Query client capability response

Query client capability response as follow:

	data type	length	value
header	0000 0001	*	*

Query client capability response header as follow:

4 octets	6 octets	6 octets	4 octets	2 octets	4 bits	4 bits	8 bits
Magic	Receive MAC	Send MAC	Session ID	Sequence number	flag	type	subtype
0x76814350	Server MAC	Client MAC	0xFFFFFFFF	0x0001	0000	0000	0000 0002

Magic ：0x76814350 IoT control protocol Magic, all message must use this Magic number.

Receive MAC: will set to client MAC address

Send MAC: will set to client MAC address

Session ID: reserved

Sequence number: one request-response transaction, must match with Query client capability request set.

Flag: reserved

Type: 0000 is management command

Subtype：0000 0002 is Query client capability response

Query client capability response data format as follow:

Data type	length	value
0000 0001	*	*

Data type : 0000 0001 is Client device capability

4.1.3   Server control client offline request

Server control client offline request is use by server send to the client that server want it offline. Package only contain header.

4 octets	6 octets	6 octets	4 octets	2 octets	4 bits	4 bits	8 bits
Magic	Receive MAC	Send MAC	SessionID	Sequence number	flag	type	subtype
0x76814350	Client MAC	Server MAC	0xFFFFFFFF	0x0002	0000	0000	0000 0003

4.1.4   Server control client offline response

Server control client offline response is send by client to received request from server. Package format as follows:

	data type	length	value
header	0000 0000	0x0001	0000 0000

Header format :

4 octets	6 octets	6 octets	4 octets	2 octets	4 bits	4 bits	8 bits
Magic	Receive MAC	Send MAC	SessionID	Sequence number	flag	type	subtype
0x76814350	Server MAC	Client MAC	0xffffffff	0x0002	0000	0000	0000 0004

Data format:

data type	length	value
0000 0000	0x0001	0000 0000

Data type: 0000 0000 is status type.

Value: 0000 0000 is success.

4.2 Function command

4.2.1 GPIO Set request

GPIO Set request package format：

	data type	length	value
header	0000 0004	0x0005	*

Header format：

4 octets	6 octets	6 octets	4 octets	2 octets	4 bits	4 bits	8 bits
Magic	Receive MAC	Send MAC	Session ID	Sequence number	flag	type	subtype
0x76814350	Client MAC	Server MAC	0xFFFFFFFF	0x0003	0000	0001	0000 0001

Data format:

data type	length	value
0000 0004	0x0005	*

4.2.2 GPIO Set response

GPIO Set response package format：

	data type	length	value
header	0000 0000	0x0001	0000 0000

Header format：

4 octets	6 octets	6 octets	4 octets	2 octets	4 bits	4 bits	8 bits
Magic	Receive MAC	Send MAC	Session ID	Sequence number	flag	type	subtype
0x76814350	Server MAC	Client MAC	0xFFFFFFFF	0x0003	0000	0001	0000 0002

Data format:

data type	length	value
0000 0000	0x0001	0000 0000

4.2.3 GPIO Get request

GPIO get request package only contain header.

4 octets	6 octets	6 octets	4 octets	2 octets	4 bits	4 bits	8 bits
Magic	Receive MAC	Send MAC	Session ID	Sequence number	flag	type	subtype
0x76814350	Client MAC	Server MAC	0xFFFFFFFF	0x0004	0000	0001	0000 0003

4.2.4 GPIO Get response

Package format：

	data type	length	value
header	0000 0004	0x0005	*

Head format：

4 octets	6 octets	6 octets	4 octets	2 octets	4 bits	4 bits	8 bits
Magic	Receive MAC	Send MAC	Session ID	Sequence number	flag	type	subtype
0x76814350	Server MAC	Client MAC	0xFFFFFFFF	0x0004	0000	0001	0000 0004

Data format

data type	length	value
0000 0004	0x0005	*

Data type：0000 0004 is GPIO information

4.2.5 others

Appendix