MediaTek

AT command customization

Documents Number:

Preliminary (Released) Information

Revision: 0.00

Release Date: Jun, 26, 2004

Revision History

Revision	Date	Author	Comments
0.00	06/25/2004	Erica	Draft version

Table of contents

1 Introduction

This document is to illustrate how our customer can define and implement their own AT commands.

1.1 Overview

In the mcu\ps\custom folder, there's a file called **custom_at_command.c** , which contains the entry function to handle custom-defined AT commands; the function is called **custom-command handler function**. The related codes to handle new customer-defined AT commands must be added to this function when a new command is added.

Customer can define a special symbol charactor such as ^ or * or $ (default is ^), so the commands have prefix AT<special symbol> (ex. AT^) will be recognized as customer-defined AT commands. The details will be describe later.

2 Proposed Implementation

2.1 Using a special symbol character

In customer_at_command.c, customer can defined their preferred symbol.

#define CUSTOM_SYMBOL '^' // '+' and '/' and ' \ 'is NOT allowed

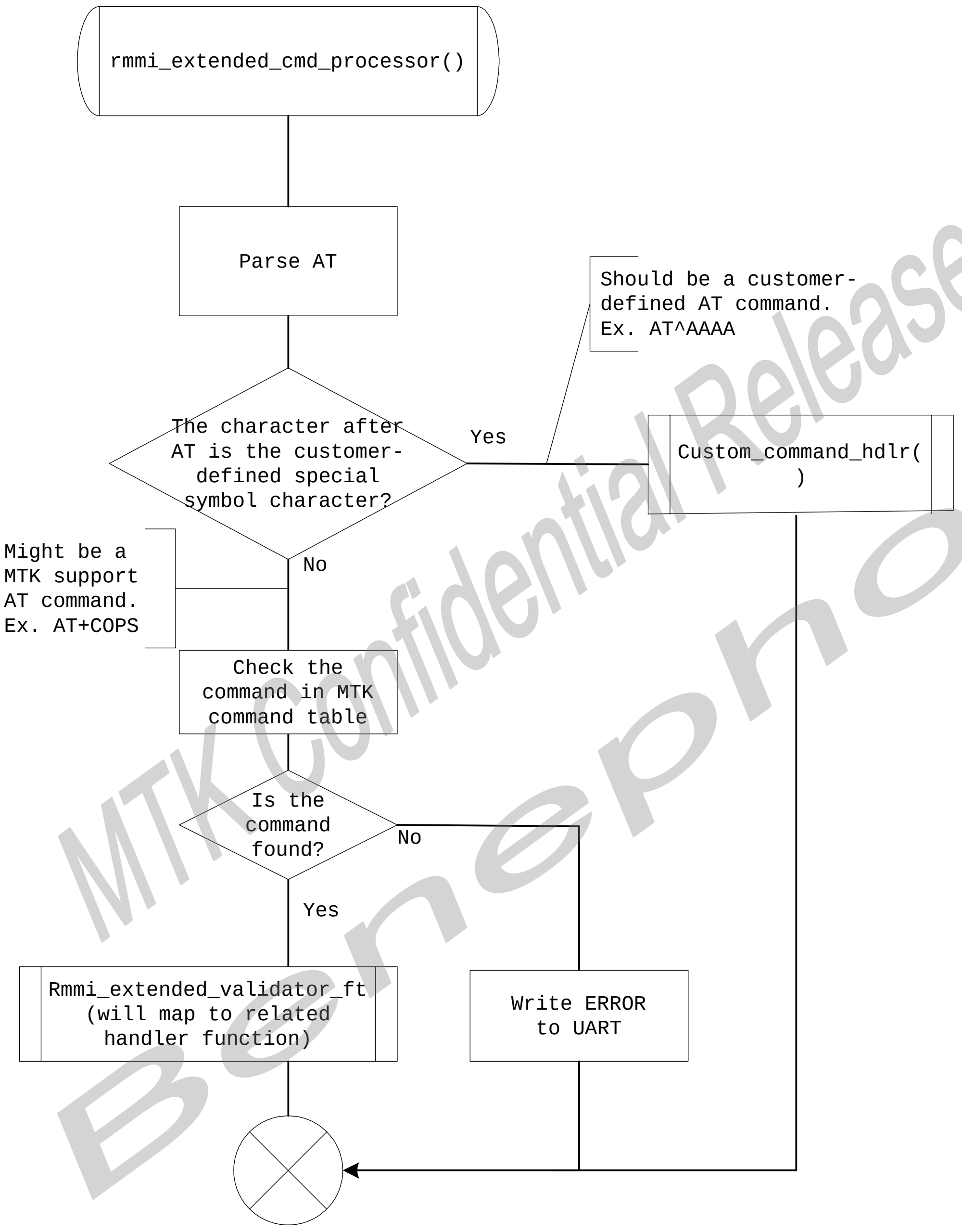
We propose to use a special symbol character to distinguish MTK AT command and Customer-defined AT command.
Currently all MTK AT commands have prefix "AT+"
For example:
AT+COPS -> standard AT command
AT+EIMG -> proprietary AT command

So customer can choose other symbol character which is different from '+' as the special symbol.

For example, if ^ is chosen, then
AT^AAAA
Or
AT^BBBB (Where <AAAA> and <BBBB> can be any string.)
Will be recognized as customer-defined AT commands.

When ATCI parser does the parsing and encounters this special symbol character, it will recognize the command as a customer-defined AT command. And the ATCI parser will pass the command string to the customer's command handler function defined in custom folder.

2.2 The flow to parse an AT command


rmmi_extended_cmd_processor()
Parse AT
Should be a customer-defined AT command. Ex. AT^AAAA
The character after AT is the customer-defined special symbol character?
Yes
Custom_command_hdlr( )
No
Might be a MTK support AT command. Ex. AT+COPS
Check the command in MTK command table
Is the command found?
No
Yes
Rmmi_extended_validator_ft (will map to related handler function)
Write ERROR to UART

2.3 Customer command handler function

Customers are responsible to do the parsing after RMMI parser passes the command string to the **custom command handler function**.

Here is a just basic **example** of parsing flow:

1. Parse the command name and see if it's a recognized command.
   Customer can have their own implementation: such as maintaining a command_table or using if-else to find the commands.

2. If the command is recognized. Parse "**=?**", "**?**", or "**=**" to decide which command mode. (TEST, READ, or EXEUTE)
3. If it's a execute command with parameters, parse the parameters. Each command might have different parameter number and type.
4. Do correspondent action according to the command (ex. Call another function)
5. At last, write final result code "OK" or "ERROR" to UART.

We will provide a basic example source code to customers in the custom folder.

2.4 Response to DTE

We will provide two RMMI extern function to customers, so customers can write data to UART:

extern void rmmi_write_to_uart (kal_uint8 *buffer, kal_uint16 length, kal_bool stuff);
This function writes a string to UART.
buffer is the pointer to the string.
length is the length of byte to be written.
When **stuff** = KAL_TRUE, <CR><LF> will be added to the beginning and end of the string.

2.5 Restriction

The customers can define and implement their own AT commands to access such as UEM, PHB which they have the source code.
If the customer-defined AT command's function is protocol stack related, they still need our support to provide l4c function.