

# AT Command Server



MARVELL®

# Overview

- One Task
- Supports 9 logical channels:
  - Channel 0: USB Modem
  - Channel 1: Dialer
  - Channel 2: Not Used
  - Channel 3: Web UI
  - Channel 4: SMS receiving
  - Channel 5: SMS sending
  - Channel 6: Customer can use it to monitor indication
  - Channel 7: Serial port for Daseul
  - Channel 8: MIFI statistic task

# Overview

- Split into 9 groups:
  - CC: call control
  - DATA: high Speed circuit switched data
  - DEV: device
  - MSG: message
  - MM: Mobility management
  - PS: packet switched data
  - PB: phoneBook
  - SIM: SIM card
  - PROD: production related
  - Except PROD group, the other groups have a 2 files to add process callback, API function to communicate with protocol stack, like telcc.c includes all AT commands callback handler for CC group, cc_api.c includes the API which send/receive CI to protocol stack
  - PROD only has one file (telprod.c) to include callback function

# Process Flow

- ProcessAtChanThread in telcontroller.c receives the AT command from different Channel
- Parser the AT command and match it to a command in the table of telcontroller.c
- Invoke the callback function registered in the table for each command
- The AT command task will return to wait for and process next command

# AT command type

- Basic AT command
  - Defines the handler for AT commands that are defined in V.250, like ATA
- Exaction AT command
  - Defines the AT command handler that supports action and test operations, like +CLCC
  - Action: no prarameters;usually queries the current status or general Comm processor or network telephony parameters
  - Test: '=?' return syntax string
- Exaction VSYNTAX AT command
  - Defines the AT command handler that supports action and test operations, not used in current implementation
  - Action: same as Exaction AT command
  - Test: not just return a syntax string. Further implementation is needed, for example when Comm is queried what parameter is supported

# AT command type

- Extended AT command
  - Defines the AT commands handler that supports set, get and test operations, like +CFUN
  - Set: ‘=‘ and parameter list; set the relevant Comm processor or network telephony parameter
  - Get: ‘?’;query the relevant Comm provcessor or network telephony parameter values
  - Test: ‘=?’ return syntax string
- Exteneded VSYNTAX AT command
  - Defines the AT commands handler that supports set, get and test operations, like +CBST
  - Set and Get are same as extended AT command
  - Test: not just return a syntax string. Further implementation is needed

# How to Add one AT command

- Define the AT command type, define the callback function and add a definition in the table of telcontroller.c

```
utlDEFINE_EXTENDED_AT_COMMAND("*MRD_CDF", starMRD_CDF_params, "*MRD_CDF=<a>,<f>", AtMrdCdf, AtMrdCdf),   // s
```

- Add the parameters definition in telcontroller.c

```
741:
742: static utlAtParameter_T starMRD_CDF_params[] = { utlDEFINE_STRING_AT_PARAMETER(    utlAT_PARAMETER_ACCESS_READ_WRITE, utlAT_PARAMETI
743:                         utlDEFINE_STRING_AT_PARAMETER(    utlAT_PARAMETER_ACCESS_READ_WRITE, utlAT_PARAMETER_PRESENCE_REQUIRED),
744:                         utlDEFINE_STRING_AT_PARAMETER(    utlAT_PARAMETER_ACCESS_READ_WRITE, utlAT_PARAMETER_PRESENCE_OPTIONAL),
745:                         utlDEFINE_STRING_AT_PARAMETER(    utlAT_PARAMETER_ACCESS_READ_WRITE, utlAT_PARAMETER_PRESENCE_OPTIONAL),};
746:
```

- Define the group and add callback in relevant callback file, like adding AtMrdCdr in telProd.c
- For not prod group, add API to send CI to protocol Stack,like SIM_GetPinState for AT+CPIN in sim_api.c
- For not prod group, add API to process the confirmation and indication. For each group, there is a 2 API functions to process different Indications and confirmations. The specific indication and confirmation can be added into these 2 API, please refer to simInd and simCnf

# Thank You!