



▶ **Marvell.** Moving Forward Faster

Marvell PXA1826 Telephony Module

2015



Agenda

- ▶ **Msocket**
- ▶ **NVM**
- ▶ **Diag**
- ▶ **Error Handler**
- ▶ **Atcmd Server**

Msocket

- ▶ MSocket is based on Share memory and ACIPC
- ▶ Each service has its unique port. Different port has different priority and flow control
- ▶ ACIPC is used to notify new data, flow control and peer notify

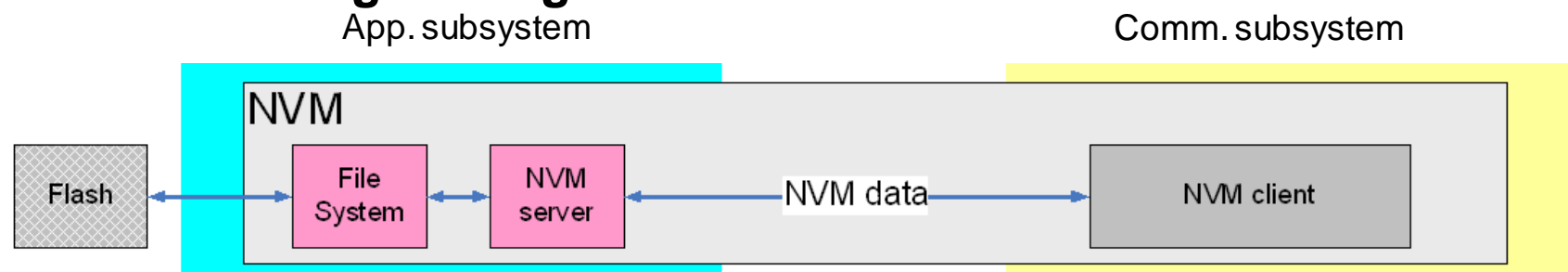
Msocket Priority

PortId	Service Name	Priority
1	cistub	0
2	NVM	0
3	cidatastub	3
5	audiostub	1
6	cicsdstub	2
9	ciimsstub	2

NVM

- ▶ NVM is data storage area in flash for CP usage.
- ▶ As CP cannot access directly to flash, AP is responsible to sync the actual code to flash (CP writes to DDR)
- ▶ Mainly used by the comm
 - Comm. configuration
 - RF calibration
 - Protocol stack related files
 - Audio configuration files

▶ Possible design change



Debug & Logging - DIAG

- ▶ Part of the Telephony. It is a diagnostic module that collect traces from the SW (MSA, APPS) and according to configuration:
 - Sends them over USB to a PC running the CATStudio tool
 - Saves them on the flash/SD card (offline logging).
 - Saves them in the DDR (cyclic buffers)
- ▶ Once the SD log files get to the PC (CATStudio, SD Card Reader, FTP) they can be opened using the CATStudio
- ▶ The “in target” Log files can be read via the CATStudio and they are also available during RAMDump
- ▶ Do not support diag over IP
- ▶ Set default_media in /usr/mrvl_tel_diag.cfg. Now usb is default media.
- ▶ Can filter AP log with APP-DIAG->TELTRACE

Error Handling

Error Handling will be executed in 2 phases:

- ▶ Report

Log of error in NVM, printout to LCD, Diag trace to ACAT, etc'.

- ▶ Recovery

Recovery of the system by means of system reset, silent reset. At the end of this stage system should be in state of full functionality.

AP EEH

AP EEH is divided into two parts:

▶ **EEH Thread running in Linux user space:**

- Read EE Handler configuration from NVM
- Listen on SEH Driver for Comm WDT event
- Output COMM EE log to NVM upon CP asserts
- Output stream Log upon CP asserts
- Communicate with Msocket Driver for link down and link up during CP silent reset
- Output local asserts information

▶ **SEH Driver in Linux Kernel space**

- Listen on Comm WDT interrupt and report to EEH thread in user space
- Enable and disable Comm WDT interrupt, set the register to hold or release CP
- Memory map for AP-CP log error share memory so that EEH thread can access this part of memory

Reset type

- ▶ Reset type is defined as finalAction in the NVM EE Configuration file(EHandlerConfig_Linux.nvm)
- ▶ Generally, there are the following reset types:
 - Stall type (EE_STALL, EE_EXTERNAL): no recovery, just output the CP postmortem logs for validation/development
 - Full reset type (EE_RESET_GPIO_RESET): reset the whole system (both AP and CP)
 - Silent reset type (EE_RESET_COMM_SILENT): reset CP only

AT command server

- ▶ Mainly parse and handle AT commands, convert them to CCI requests and send to CP.
- ▶ All AT command are defined in shell_commands[] in telcontroller.c.
- ▶ Set Indication capability in gTelAtpDescConfig in telcontroller.c.
- ▶ Set Indication capability of serial port with AT*SPIND.

```
static utlAtCommand_T shell_commands[] = {
    utlDEFINE_EXTENDED_AT_COMMAND("+CREG", plusCREG_params, "+CREG: (0-2)", ciRegStatus, ciRegStatus),
    utlDEFINE_EXTENDED_AT_COMMAND("+CIND", plusCIND_params, "+CIND: (0,1)", ciNwModeInd, ciNwModeInd),
```

AT Command Handler Type

Request Type

- ▶ Action
- ▶ Get
- ▶ Set
- ▶ Test(to get usage string from syntax or syntax function)

Example

1. Action request : AT+CSQ
2. Get request : AT+CREG?
3. Set request : AT+CGACT=0
4. Test request : AT+CREG=?

AT Request

- ▶ utlDEFINE_BASIC_AT_COMMAND
 - set function
- ▶ utlDEFINE_EXTENDED_AT_COMMAND
 - syntax, get/set function
- ▶ utlDEFINE_EXACTION_AT_COMMAND
 - syntax, action function
- ▶ utlDEFINE_EXTENDED_VSYNTAX_AT_COMMAND
 - syntax function, get/set function
- ▶ utlDEFINE_EXACTION_VSYNTAX_AT_COMMAND
 - syntax function, action function
- ▶ utlDEFINE_EXTENDED_EXACTION_AT_COMMAND
 - syntax, action function

AT Example

```

▶ switch(op){
▶   case TEL_EXT_SET_CMD:                                /* AT*MODEMRESET= */
▶   {
▶       char buf[512]; INT16 len; int force;
▶       if(getExtString(parameter_values_p,0, (CHAR *)buf, sizeof(buf) - 1, &len, NULL) == TRUE){
▶           if(getExtValue( parameter_values_p,1, &force, 0, 1, 0) == TRUE){
▶               ret = ATRESP( atHandle, ATCI_RESULT_CODE_SUPPRESS, 0, NULL);
▶               if(cp_silent_reset_on_req(buf,force) != 0){
▶                   ret = ATRESP( atHandle, ATCI_RESULT_CODE_CME_ERROR, CME_UNKNOWN, NULL);
▶               }
▶               else
▶                   ret = ATRESP( atHandle, ATCI_RESULT_CODE_NULL, 0, "OK");
▶           }
▶           else
▶               ret = ATRESP( atHandle, ATCI_RESULT_CODE_CME_ERROR, CME_INVALID_PARAM, NULL);
▶       }
▶       break;
▶   }
▶   case TEL_EXT_ACTION_CMD:                             /* AT*MODEMRESET */
▶   case TEL_EXT_TEST_CMD:                               /* AT*MODEMRESET=? */
▶   case TEL_EXT_GET_CMD:                                /* AT*MODEMRESET?*/
▶   Default:{
▶       ret = ATRESP( atHandle, ATCI_RESULT_CODE_CME_ERROR, CME_OPERATION_NOT_SUPPORTED, NULL);
▶       break;}
▶ }

```

Thank You!