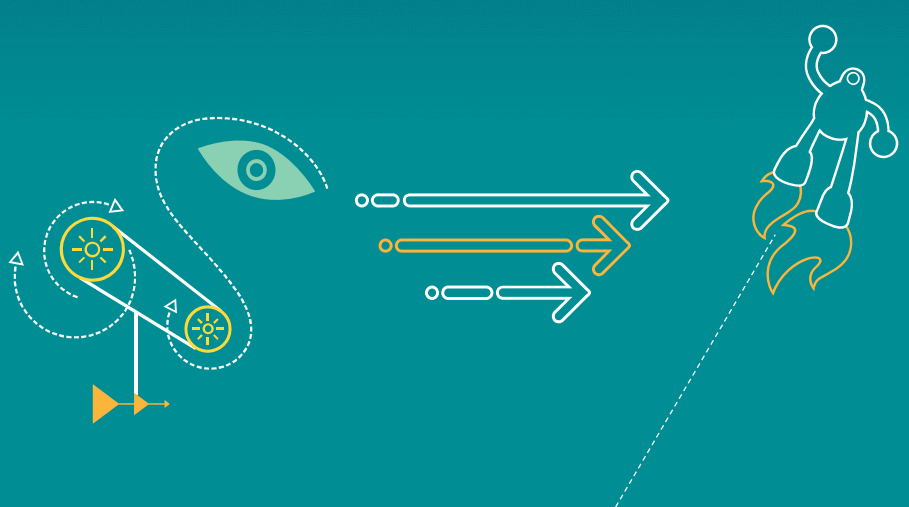
UIM Drivers-Based Log Analysis Overview

Qualcomm Technologies, Inc.



80-NE802-1 D

Confidential and Proprietary – Qualcomm Technologies, Inc.

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

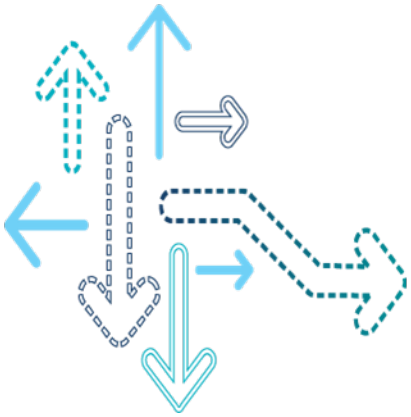
Revision History

Revision	Date	Description
A	January 2013	Initial release
B	January 2015	Updated NV Settings section
C	August 2015	Added Hotswap section and updated NV Settings section.
D	November 2016	Added slides 84-93; updated slides 7, 20, 21, 26, 30, 31, 35, 52, 57, 100-103, and 105.

Contents

Contents

Log Packets

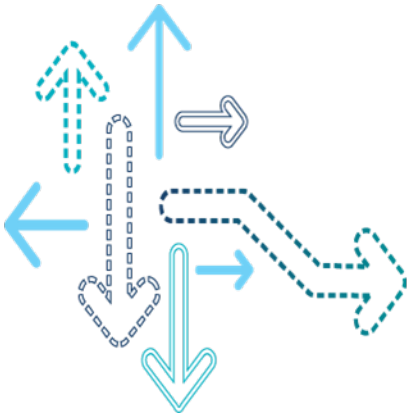


Log Packets

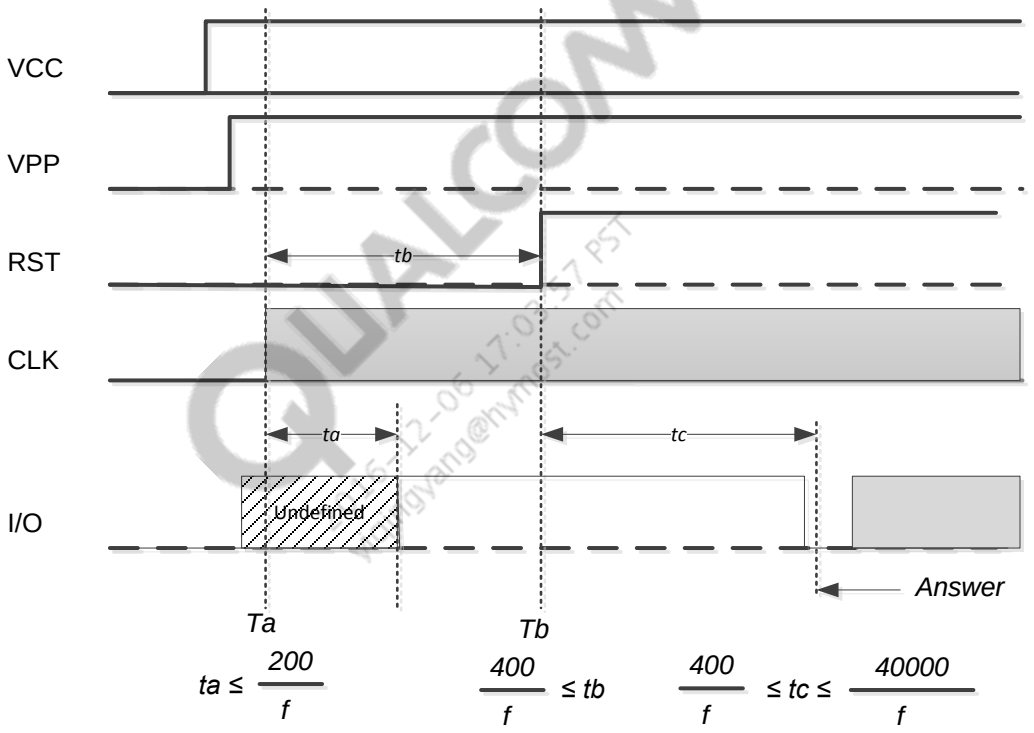
- The following log packets need to be enabled for collecting user identity module (UIM) modem logs:
  - Legacy
  - UIM
  - UMTS
  - GSM
  - Common→0x1098 UIM application protocol data unit (PDU)
  - For dual sim dual standby (DSDS) targets, check 0x14CE Internal – UIM DS Data for getting Slot 2 APDUs
  - For MPSS.AT.2.1 and later, check 0x19B7 for new APDU log packet

- The following RAT-specific log packets need to be enabled depending on scenario:
  - 1X default
  - LTE
  - CDMA system determination
  - Call manager
  - Multimode controller
  - 1xEV-DO
  - 1xEV-DO debug
  - Data services

Activation Sequence



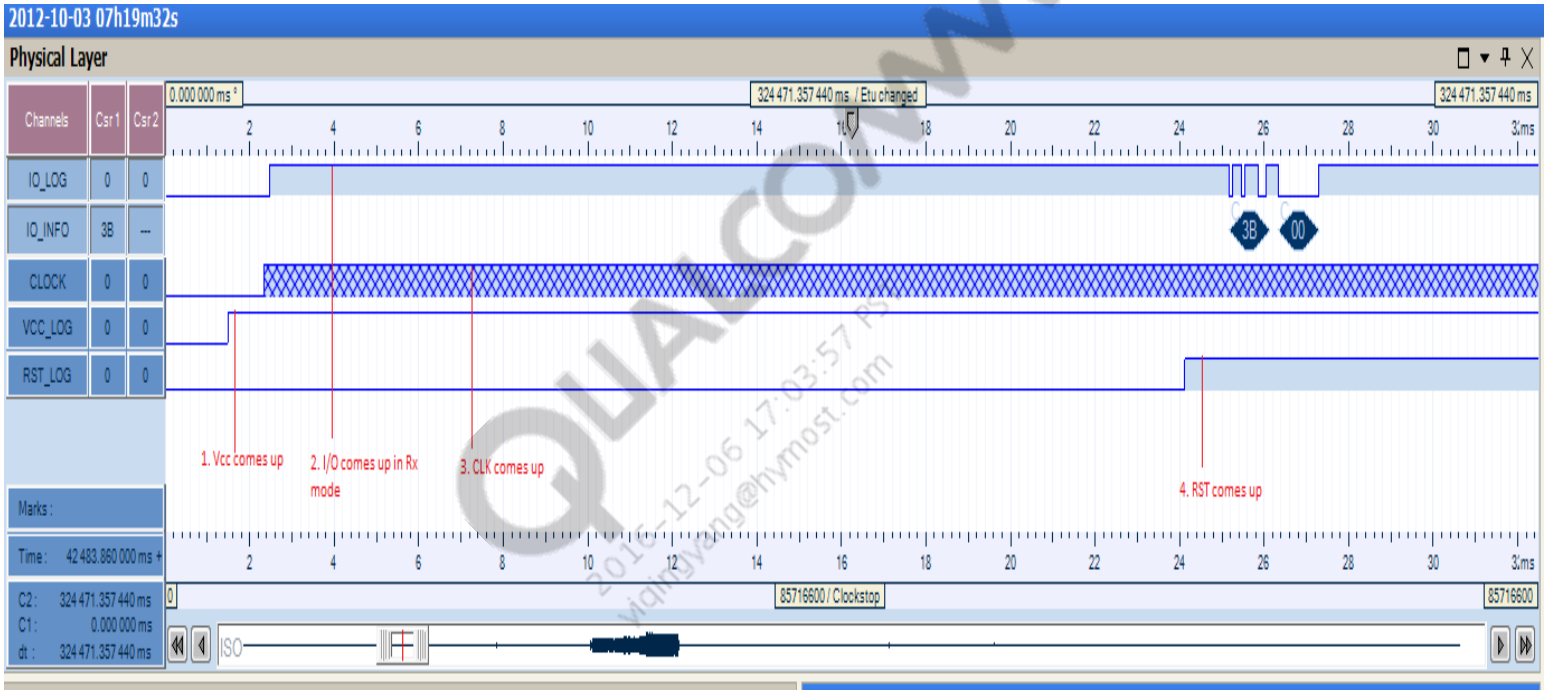
Activation Sequence – Diagram



Activation and cold reset

Note: See *ISO 7816 Part 3: Electronic Signals and Transmission Protocols* (ISO 7816 Part 3), Sections 5.2 and 5.3.2.

Activation Sequence – IT3 Logs



Activation Sequence – Code Snippet

```
FILE: uimdrv.c
Function : uim_power_up
void uim_power_up
(
  void
)
{
..
/* Vcc comes up */
UIM_POWER_ON ();

/* I/O comes up -  Next, place the I/O line in reception mode. */
  UIM_STOP_BREAK ();

/* CLK comes up - Setup the UIM clock based on available clock frequency. */
  uim_clock_control ( uim_clock_control_val, uim_drv_slot );

} /* end - uim_power_up */

FILE: uimgen,c
Function : uim_generic_command
 /* RST comes up - Reset the UIM card */
uim_reset( &uim_rsp_buf, *(uim_command_response_callback));
```


Activation Sequence – QXDM Logs

For getting power up logs – insert a sleep of 25s (rex_sleep(25000);) in uim_power_up function

MSG [00021/00] User Identity Module/Low 00:00:29.385 uimdrv.c 04434 uim_power_up

//Voltage Comes up

MSG [00021/02] User Identity Module/High 00:00:29.385 uimdrv.c 04441 uim power up @ 1.8 v

MSG [00021/02] User Identity Module/High 00:00:29.385 uimdrv.c 04559 uim clk freq 2

MSG [00021/02] User Identity Module/High 00:00:29.385 uimdrv.c 04566 uim clk freq is 3840000

//IO would have come up at this point

//CLK comes up

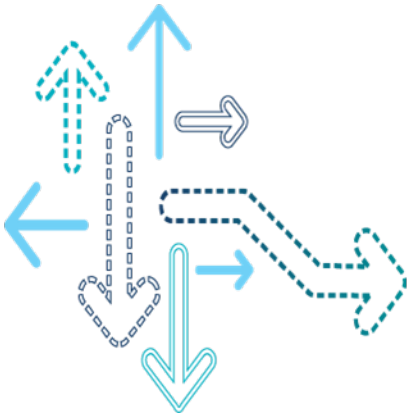
MSG [00021/02] User Identity Module/High 00:00:29.386 uimdrv.c 04662 Turned on the clock for RUIM

MSG [00021/02] User Identity Module/High 00:00:29.387 uimdrv.c 04668 Turned on the power

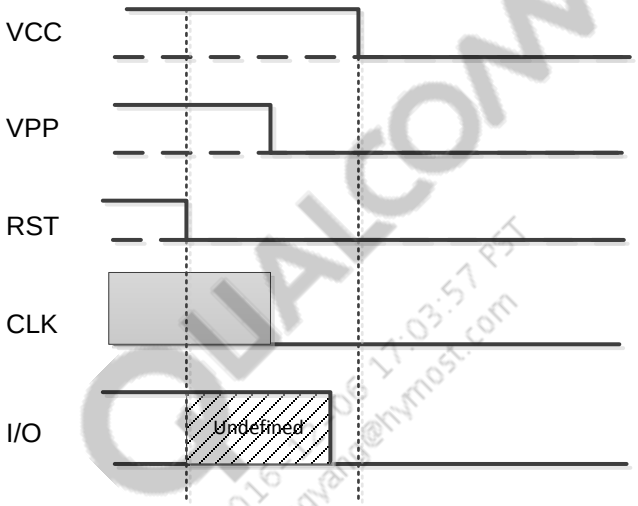
//RST comes up

MSG [00021/00] User Identity Module/Low 00:00:29.407 uimdrv.c 04761 uim_reset

Deactivation Sequence



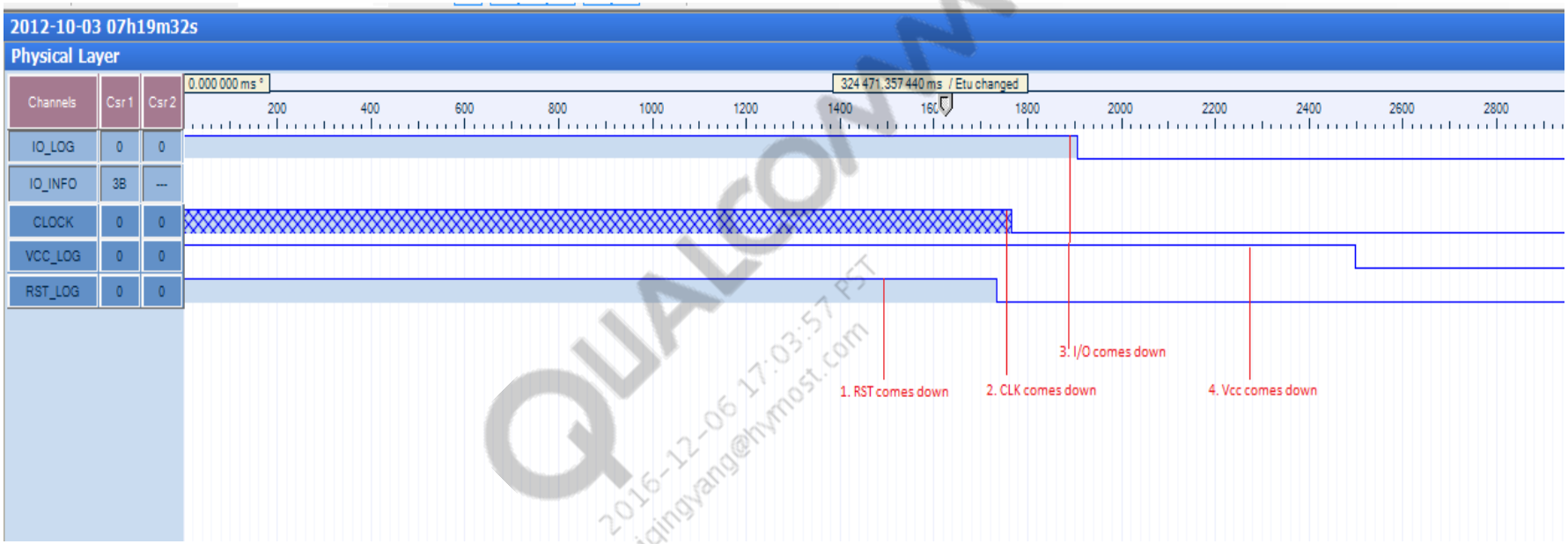
Deactivation Sequence – Diagram



- RST shall be put to state L
- CLK shall be put to state L (unless the clock is already stopped at state L)
- VPP shall be deactivated (if it has been activated)
- I/O shall be put to state A
- VCC shall be deactivated

Note: See *ISO 7816 Part 3: Electronic Signals and Transmission Protocols* (ISO 7816 Part 3), Section 5.4.

Deactivation Sequence – IT3 Logs



Deactivation Sequence – Code Snippet

```
FILE: uimdrv.c
Function : uim_power_down

void uim_power_down
(
  void
)
{
..
/* RST comes down -  Reset the UART receiver and disable */
   /* Set the reset active */
  UIM_ASSERT_RESET ();

  /* Wait for 100 clock cycles before turning the clock off. */
  uim_clk_busy_wait ((100*1000000)/uim_clk_freq[uim_drv_slot]);

 /* CLK comes down - Turn the clock to the UIM off */
  uim_clock_control ( UIM_CLOCK_LOW, uim_drv_slot );

 /* Wait for 100 clock cycles before setting I/O line low. */
  uim_clk_busy_wait ((100*1000000)/uim_clk_freq[uim_drv_slot]);

 /* I/O comes down - Set the I/O line Low */
  UIM_START_BREAK ();

 /* Wait for 100 clock cycles before turning power off. */
  uim_clk_busy_wait ((100*1000000)/uim_clk_freq[uim_drv_slot]);

  /* Vcc comes down - Turn off the UIM */
UIM_POWER_OFF ();
} /* end - uim_power_up */
```


Power Down – QXDM Logs

//MMGSDI receives power down request from its clients

MSG [00021/02] User Identity Module/High 00:03:13.012 mmgsdi.c 03846 Received MMGSDI_CARD_PDOWN_REQ

//MMGSDI sends power down command to UIM

MSG [00021/02] User Identity Module/High 00:03:13.012 mmgsdi_gen.c 00373 Sending POWER down command to UIM

MSG [00021/02] User Identity Module/High 00:03:13.013 mmgsdilib.c 03769 Successful queue of Card Pdown command 0x0

MSG [00021/02] User Identity Module/High 00:03:13.015 uimgen.c 03853 Received power down command

//Turning off UIM

MSG [00021/03] User Identity Module/Error 00:03:13.015 uimgen.c 05855 Turning off UIM because of POWER down cmd

//Since card is a 3V card – thus powering down at 3V

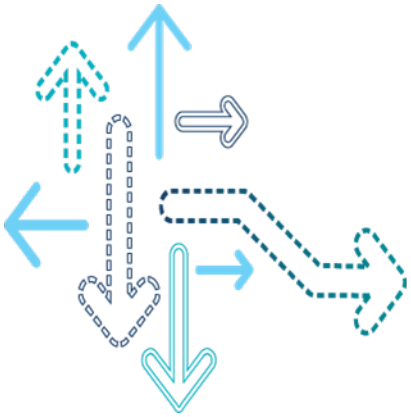
MSG [00021/02] User Identity Module/High 00:03:13.015 uimdrv.c 06553 uim power down @ 3 v

//uim_power_down function hit

MSG [00021/00] User Identity Module/Low 00:03:13.015 uimdrv.c 06578 uim_power_down

MSG [00021/03] User Identity Module/Error 00:03:13.041 gsdi_convert.c 02732 UIM_POWER_DOWN_CMD_NOTIFICATION_S ==> GSDI_CARD_ERR_POWER_DOWN_CMD_NOTIFICATION

ATR Sequence



ATR Sequence – IT3 Logs

ISO Protocol Layer

#	Sequence	Dir	Description	Value	Delay	Meaning
1	Cold reset		Event			Reset card interface
2	ATR	C	TS	3B		Direct convention
		C	T0	9E	12.00 etu	Following interface chars: TA1, TD1 Number of historical chars: 14
		C	TA1	95	12.00 etu	F = 512 D = 16
		C	TD1	80	12.00 etu	Following interface chars: TD2 Transfer protocol T=0
		C	TD2	1F	12.00 etu	Following interface chars: TA3 Interface character qualifier: T=15
		C	TA3	C3	12.00 etu	X = No preference U = Class A and B
		C	T1	80	12.00 etu	
		C	T2	31	12.00 etu	
		C	T3	E0	12.00 etu	
		C	T4	73	12.00 etu	
		C	T5	FE	12.00 etu	
		C	T6	21	12.00 etu	
		C	T7	1B	12.00 etu	
		C	T8	66	12.00 etu	
		C	T9	D0	12.00 etu	
		C	T10	00	12.00 etu	
		C	T11	6C	12.00 etu	
		C	T12	09	12.00 etu	
		C	T13	1A	12.00 etu	
		C	T14	00	30.80 etu	
		C	TCK	78	12.00 etu	Check character

Note: See *ISO 7816 Part 3: Electronic Signals and Transmission Protocols* (ISO 7816 Part 3), ATR bytes, Section 6.

ATR Sequence – QXDM Logs

2012 Oct 3 00:13:13.720 [00] 0x1FEB Extended Debug Message uimdrv.c 4761 L uim_reset

//UIM state is UIM_RESET_ST

2012 Oct 3 00:13:13.721 [00] 0x1FEB Extended Debug Message uimgen.c9393 L UIM generic state in uim_command 1

//Card responds back with the ATR Rx bytes

2012 Oct 3 00:13:13.786 [00] 0x1098 RUIM Debug

RX 3B 9E 95 80 1F C3 80 31 E0 73 FE 21 1B 66 D0 00 6C 09 1A

00:00:29.414

RX 00 78

// Command response signal received

2012 Oct 3 00:13:13.753 [00] 0x1FEB Extended Debug Message uim.c 4624 L Recd Command Response Signal

2012 Oct 3 00:13:13.753 [00] 0x1FEB Extended Debug Message uim.c 10862 L uim_process_card_response

//Successful cmd status, no SW bytes seen for ATR, 21 byte ATR response seen

2012 Oct 3 00:13:13.753 [00] 0x1FEB Extended Debug Message uim.c 5855 H cmd status 0x0, SW1 0x0, SW2 0x0, Response data length 0x15

ATR Sequence – QXDM Logs (cont.)

//Generic State pointer = UIM_RESET_ST

2012 Oct 3 00:13:13.753 [00] 0x1FEB Extended Debug Message uimgen.c 9533 H generic_state_ptr 0x1, uim_reselect_mf 0x0

//Baud Rate Fi/Di = 512/16 as seen in RUIM debug are supported

2012 Oct 3 00:13:13.753 [00] 0x1FEB Extended Debug Message uimdrv.c 17269 H FI and DI are supported

//UIM state in UIM_DELAY_AFTER_ATR_ST – delay added after receving ATR before sending next command

2012 Oct 3 00:13:13.754 [00] 0x1FEB Extended Debug Message uimgen.c 9393 L UIM generic state in uim_command 2

2012 Oct 3 00:13:13.754 [00] 0x1FEB Extended Debug Message uim.c 10862 L uim_process_card_response

2012 Oct 3 00:13:13.754 [00] 0x1FEB Extended Debug Message uim.c 5855 H cmd status 0x0, SW1 0x0, SW2 0x0, Response data length 0x15

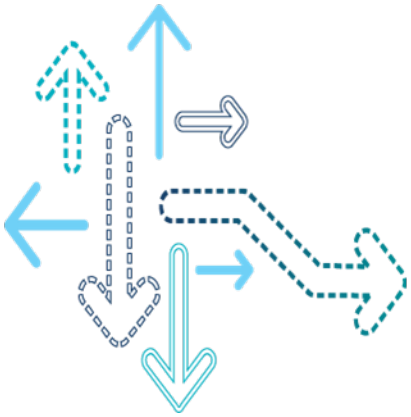
//Generic State pointer = UIM_DELAY_AFTER_ATR_ST

2012 Oct 3 00:13:13.754 [00] 0x1FEB Extended Debug Message uimgen.c 9533 H generic_state_ptr 0x2, uim_reselect_mf 0x0

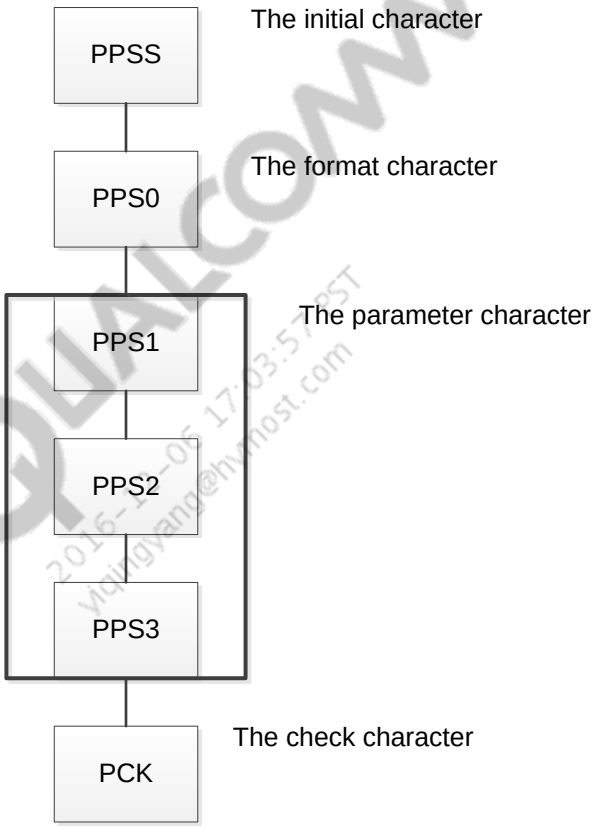
ATR Sequence – Code Snippet

```
FILE: uimdrv.c

Function : rx_isr_receive_atr :  This function runs within the context of the Rx ISR of the
UART.  This function is called when expecting an ATR from the UIM.  The ATR bytes are
processed as they come in.  If the ATR is received without conflict,    the UIM server is
notified.

LOCAL void rx_isr_receive_atr
(
  uim_drv_slot_type uim_drv_slot  /* Slot control variable */
)
{
..
…
}
```


PPS Procedure



PPS Procedure – Diagram



Structure of PSS request and response

Note: See *ISO 7816 Part 3: Electronic Signals and Transmission Protocols* (ISO 7816 Part 3), Section 7.3.

PPS Procedure – IT3 Logs

ISO Protocol Layer

#	Sequence	Dir	Description	Value	Delay	Meaning
3	PPS request	T	PPSS	FF	55.01 etu	Initial character
		T	PPS0	10	12.06 etu	Following parameter chars: PPS1 Transfer protocol: T=0
		T	PPS1	95	12.06 etu	F = 512 D = 16
		T	PCK	7A	12.06 etu	Check character
4	PPS response	C	PPSS	FF	12.01 etu	Initial character
		C	PPS0	10	12.00 etu	Following parameter chars: PPS1 Transfer protocol: T=0
		C	PPS1	95	12.00 etu	F = 512 D = 16
		C	PCK	7A	12.00 etu	Check character

PPS Procedure – QXDM Logs

//Send pps

2012 Oct 3 00:13:13.755 [00] 0x1FEB Extended Debug Message uimdrv.c 5042 L uim_send_pps

//UIM state in UIM_PPS_ST

2012 Oct 3 00:13:13.755 [00] 0x1FEB Extended Debug Message uimgen.c 9393 L UIM generic state in uim_command 3

2012 Oct 3 00:13:13.766 [00] 0x1FEB Extended Debug Message uim.c 4624 L Recd Command Response Signal

// Received the PPS related RX bytes

00:00:29.418 RX FF 10 95 7A

2012 Oct 3 00:13:13.766 [00] 0x1FEB Extended Debug Message uim.c 10862 L uim_process_card_response

//Success status for pps response

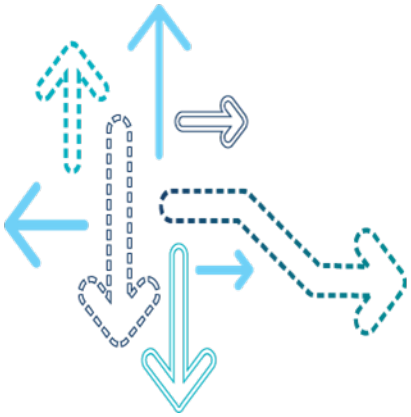
2012 Oct 3 00:13:13.766 [00] 0x1FEB Extended Debug Message uim.c 5855 H cmd status 0x0, SW1 0x0, SW2 0x0, Response data length 0x4

2012 Oct 3 00:13:13.766 [00] 0x1FEB Extended Debug Message uimgen.c 9533 H generic_state_ptr 0x3, uim_reselect_mf 0x0

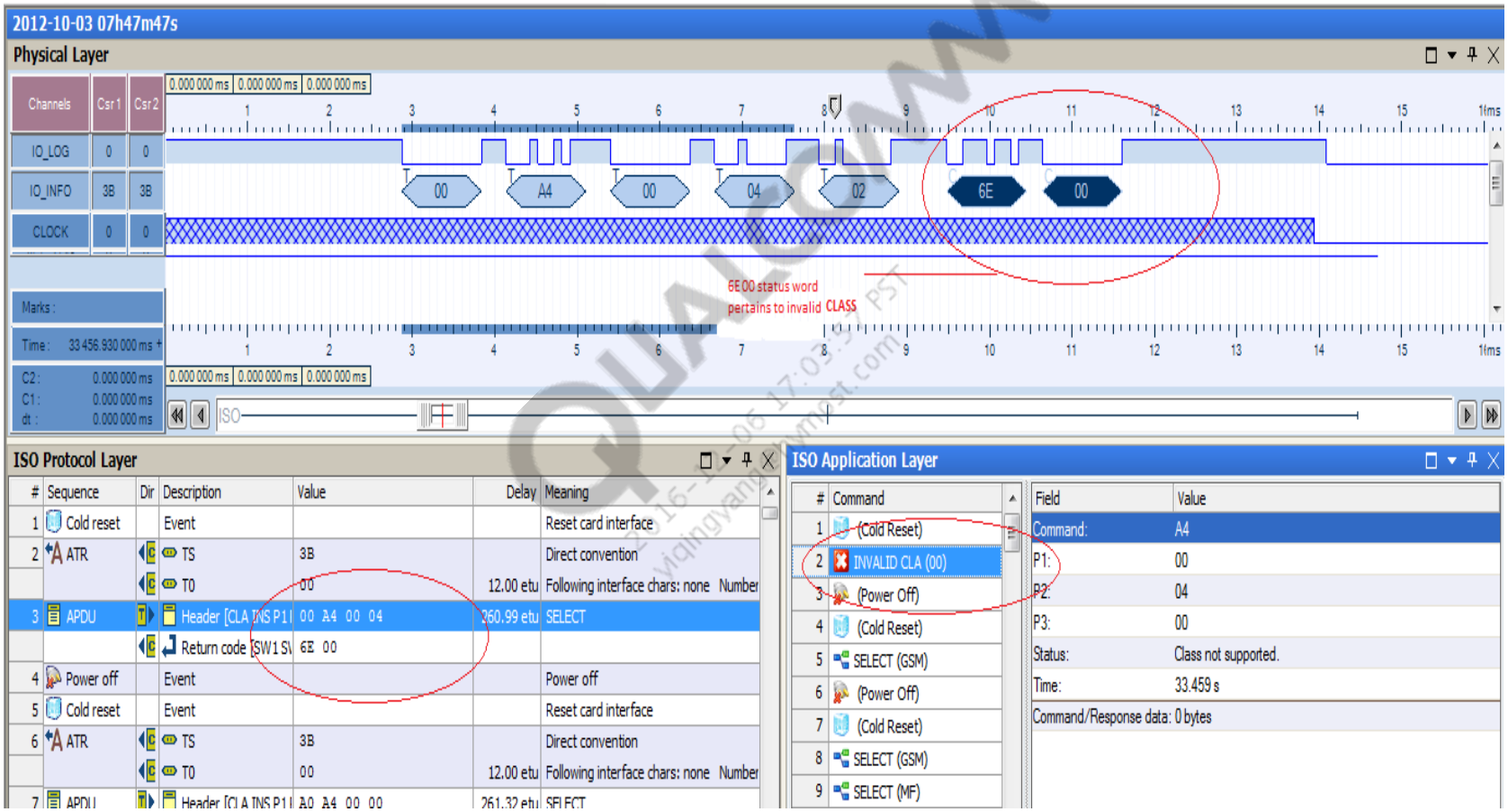
PPS Procedure − Code Snippet

```
FILE: uimdrv.c
Function : uim_send_pps :   This function sets up the transfer of a PPS request to the UIM.
This function starts the transfer, which, if all goes well, is finished by the uim_rx_isr.
LOCAL void uim_send_pps
(
  uim_pps_req_buf_type const *pps_request  /* This points to the PPS request and
                                           the PPS response buffer.           */
)
{
..
…
}

Function: RX_ISR_Receive_PPS
 This function runs within the context of the Rx ISR of the UART. This function is called
when expecting to receive a PPS response.  The PPS response bytes are processed as they come
in.  If the PPS response is received without conflict, the UIM server is notified.
LOCAL void rx_isr_receive_pps
(
  uim_drv_slot_type uim_drv_slot  /* Slot control variable */
)
{
..
..
}
```

UIM Class Switch



Class Switch – IT3 Logs



Class Switch – QXDM Logs

2012 Oct 3 00:27:29.977 [00] 0x1FEB Extended Debug Message uimdrv.c 4434 L uim_power_up

//Receive ATR

2012 Oct 3 00:27:30.027 [00] 0x1098 RUIM Debug

RX 3B 00

//Send Select Command

00:00:29.374

TX 00 A4 00 04 02

2012 Oct 3 00:27:30.027 [00] 0x1FEB Extended Debug Message uimgen.c 9393 L UIM generic state in uim_command 7

//Card responds back with invalid status 6E 00

2012 Oct 3 00:27:55.105 [00] 0x1098 RUIM Debug

RX 6E 00

2012 Oct 3 00:27:30.038 [00] 0x1FEB Extended Debug Message uim.c 5855 H SW1 0x6e,SW2 0x0, Response data length 0x0

2012 Oct 3 00:27:30.038 [00] 0x1FEB Extended Debug Message uim.c 4624 L Recd Command Response Signal

2012 Oct 3 00:27:30.038 [00] 0x1FEB Extended Debug Message uim.c 10862 L `uim_process_card_response

Class Switch – QXDM Logs (cont.)

//Response Status UIM_WRONG_CLASS

2012 Oct 3 00:27:30.038 [00] 0x1FEB Extended Debug Message	uim.c	5855	H	cmd status 0x13, SW1 0x0, SW2 0x0, Response data length 0x0
2012 Oct 3 00:27:30.038 [00] 0x1FEB Extended Debug Message	uimgen.c	9533	H	generic_state_ptr 0x7, uim_reselect_mf 0x0

//Reset UIM after card responds with 6e 00

2012 Oct 3 00:27:30.038 [00] 0x1FEB Extended Debug Message	uim.c	8536	H	Internal command to Reset the UIM for slot 0x1

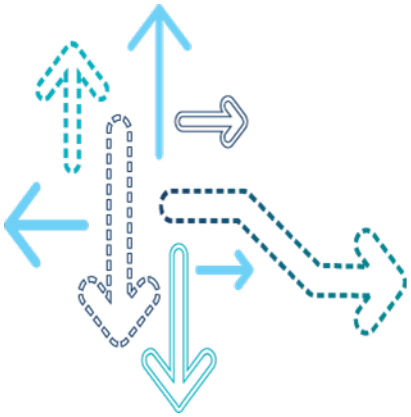
//Power Down UIM

2012 Oct 3 00:27:30.038 [00] 0x1FEB Extended Debug Message	uimdrv.c	6549	H	uim power down @ 1.8 v
2012 Oct 3 00:27:30.038 [00] 0x1FEB Extended Debug Message	uimdrv.c	6578	L	uim_power_down

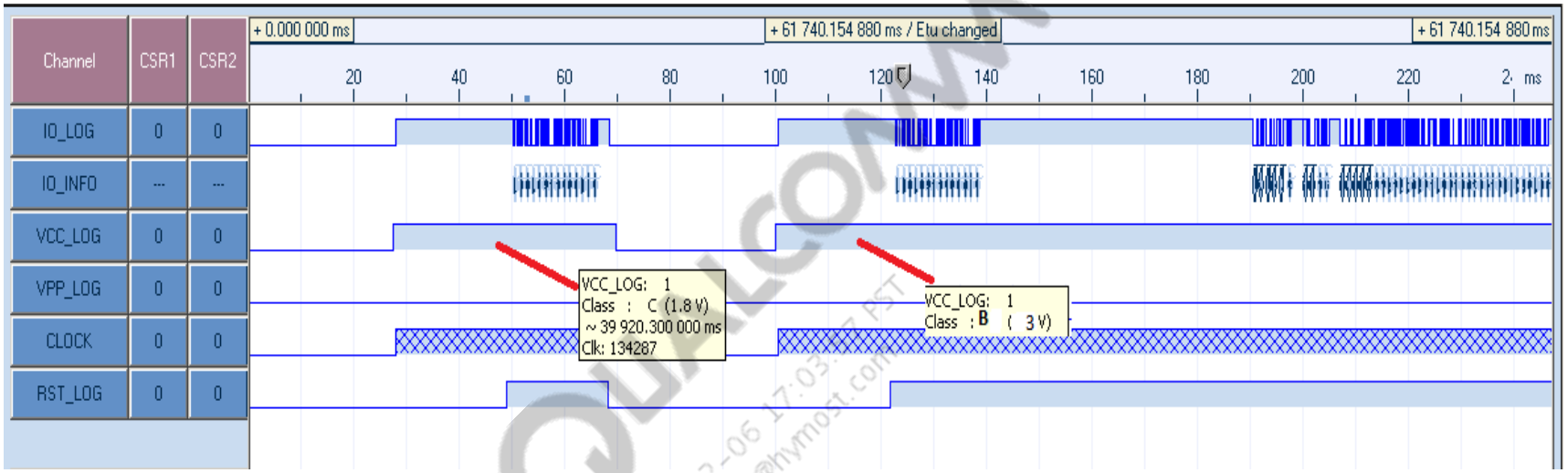
Class Switch – Code Snippet

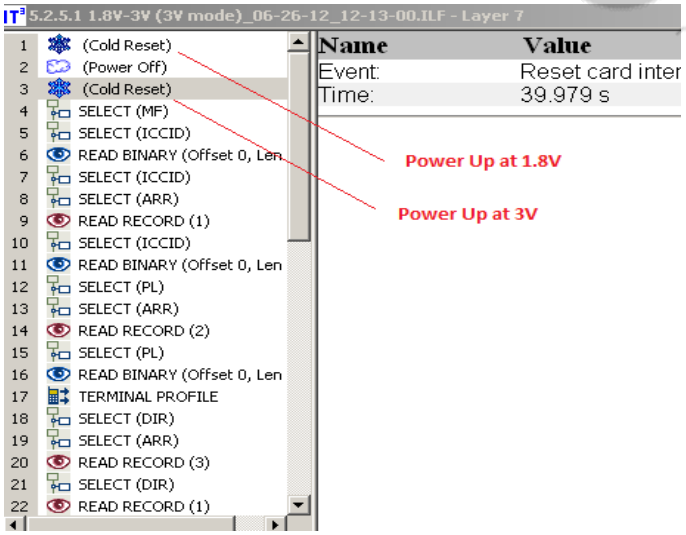
```
FILE: uimdrv.c
Function : rx_isr_sw1_cla_bad:   This function runs within the context of the Rx ISR of the UART.  This
  function is called when expecting the second status word after a class
  is bad status word.
```

Voltage Switch



Voltage Switch – IT3 Logs





Voltage Switch – QXDM Logs

2012 Oct 2	23:38:07.781	[00]	0x1FEB	Extended Debug Message	uimdrv.c	4434	L	uim_power_up
2012 Oct 2	23:38:07.781	[00]	0x1FEB	Extended Debug Message	uimdrv.c	4441	H	uim power up @ 1.8 v
2012 Oct 2	23:38:07.804	[00]	0x1FEB	Extended Debug Message	uimdrv.c	4761	L	uim_reset
2012 Oct 2	23:38:07.806	[00]	0x1FEB	Extended Debug Message	uimgen.c	9393	L	UIM generic state in uim_command 1
2012 Oct 2	23:38:07.825	[00]	0x1FEB	Extended Debug Message	uim.c	4624	L	Recd Command Response Signal

//card does not support 1.8V – TA1 of the card's ATR specifies the voltage class supported by card

//Even if ATR idoes not indicate, we can switch if the card does not respond at 1.8V

2012 Oct 2	23:38:07.825	[00]	0x1FEB	Extended Debug Message	uimgen.c	9766	L	Card does not support 1.8V
2012 Oct 2	23:38:07.825	[00]	0x1FEB	Extended Debug Message	uimgen.c	9771	H	Turning off the interface to start in 3V mode
2012 Oct 2	23:38:07.825	[00]	0x1FEB	Extended Debug Message	uimdrv.c	6549	H	uim power down @ 1.8 v

Voltage Switch – QXDM Logs (cont.)

2012 Oct 2 23:38:32.854 [00] 0x1FEB Extended Debug Message	uimdrv.c	4434	L	uim_power_up	

//UIM powers up at 3V

2012 Oct 2 23:38:32.854 [00] 0x1FEB Extended Debug Message	uimdrv.c	4445	H	uim power up @ 3 v	

Voltage Switch, ATR Received Specifies Voltage Class Supported – Code Snippet

```
FILE: uimgen.c

Function : uim_generic_command_response:    This procedure processes the response to a generic
command that has been

  received from the UIM.

voltage_class_indicator_scan  boolean checks if we have TA(i) after the first occurrence of T=15 in
TD(i-1)


uim_cmd_status_type uim_generic_command_response

(

 uim_rsp_buf_type *rsp_ptr,

 uim_cmd_type     *cmd

)

{

..

…. if ( voltage_class_indicator_scan == TRUE)

          {

            voltage_class_indicator_scan = FALSE;

            voltage_class_indicator = ( rx_value & UIM_UI_MASK );

            voltage_class_known_from_atr = TRUE;
```


Voltage Switch, ATR Received Specifies Voltage Class Supported – Code Snippet (cont.)

```
/* Determine the current voltage on the interface */
            switch (uim_current_voltage_class[uim_drv_slot])
            {
              case UIM_VOLTAGE_1_8V:
                {
                  /* Check if the card does not support this voltage */
                  if ((voltage_class_indicator & UIM_UI_1_8_UIM_UICC) !=
                      UIM_UI_1_8_UIM_UICC)
                  {
                    MSG_LOW_UIM ( uim_drv_slot, "Card does not support 1.8V",0,0,0);
                    /* Check if card supports other supported voltages */
                    if ((voltage_class_indicator & UIM_UI_3V_UIM_UICC) ==
                        UIM_UI_3V_UIM_UICC)
                    {
                      MSG_HIGH_UIM ( uim_drv_slot, "Turning off the interface to start in 3V mode",0,0,0);
#ifndef FEATURE_UIM_SUPPORT_DUAL_SLOTS
                      /* Switch to different voltage */
                      /* Power down the interface */
                      uim_power_down();
#else
                      /* Switch to different voltage */
                      /* Power down the interface */
                      uim_power_down(cmd->hdr.slot);
#endif
```

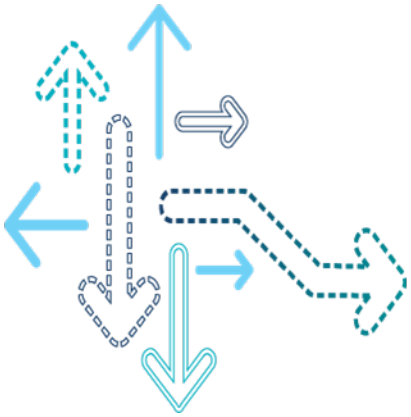

Voltage Switch, No ATR Received – Code Snippet

File: uim.c

Function: uim_process_cmd_error

```
        /* We will try the next voltage class only if the first_inst_class
        is GSM or we do not have the voltage class information from
        the ATR and we have a higher voltage available*/
        if ( ( uim_first_inst_class == NV_UIM_FIRST_INST_CLASS_GSM_SIM ) ||
             ( voltage_class_known_from_atr == FALSE )
           )
        {
              uim_timed_sleep(UIM_ISO7816_VOLTAGE_SWITCH_PWR_DOWN_DELAY);
           /* Try the next voltage class */
          uim_current_voltage_class[uim_drv_slot]++;
          /* Reset the error count */
          uim_static_cmd_ptr->hdr.cmd_count = 0;
        }
```


Command Cases



Case 1 Command

- A C-APDU of {CLA INS P1 P2} is passed from the terminal to the UICC (note that P3 of the C-TPDU is set to '00'). An R-APDU of {90 00} is returned from the UICC to the terminal.

	Terminal		UICC
C-TPDU	[CLA INS P1 P2 P3=00]	->	
		<-	90 00

2012 Sep 7 18:32:11.822 [A8] 0x1098 RUIM Debug

TX 80 F2 00 0C 00 // [CLA INS P1 P2 P3=00]

18:32:11.790

RX 90 00 // 90 00

Note: See *Universal Mobile Telecommunications System (UMTS); USIM Conformance Test Specification* (3GPP TS 31.122) ETSI TS 102.221.

Case 2 Command – Luicc < 256 Bytes

- In this first example, a C-APDU of {CLA INS P1 P2 Le = 00} is passed from the terminal to the UICC with Luicc < 256 bytes.

	Terminal		UICC
C-TPDU	[CLA INS P1 P2 P3=00]	->	
		<-	6C Luicc
C-TPDU	[CLA INS P1 P2 Luicc]	->	
		<-	INS [Data(Luicc)] 90 00

2012 Sep 7 18:30:06.158 [00] 0x1098 RUIM Debug

TX 80 F2 01 01 00 // [CLA INS P1 P2 P3=00]

18:30:06.157

RX 6C 12 // 6C Luicc

18:30:06.158

2012 Sep 7 18:30:06.168 [00] 0x1098 RUIM Debug

TX 80 F2 01 01 12 // [CLA INS P1 P2 Luicc]

18:30:06.162

RX F2 84 10 A0 00 00 00 87 10 02 F8 86 FF 92 89 05 0B 00 FF 90 00
// INS [Data(Luicc)] 90 00

Case 2 Command – Luicc = 256 Bytes

- C-APDU of {CLA INS P1 P2 Le = 00} is passed from the terminal to the UICC with Luicc = 256 bytes.

	Terminal		UICC
C-TPDU	[CLA INS P1 P2 P3=00]	->	
		<-	6C Luicc
C-TPDU	[CLA INS P1 P2 Luicc]	->	
		<-	INS [Data(Luicc)] 90 00

```
2012 Sep  7  18:30:13.272  [00]  0x1098  RUIM Debug
                                   TX        00 B0 00 00 00        // [CLA INS P1 P2 P3=00]
          18:30:13.269
                                   RX        B0 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01
  10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10
  02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02
  42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42
  98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98
  F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9
  FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF
  FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF
  FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF
  FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF
  FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF FF FF FF FF FF 98 03 42 09 01 10 02 42 98 F9 FF FF FF FF FF
  FF 90 00  // INS [Data(Luicc=256)] 90 00
```

Case 2 Command – Using '61' and '6C' Procedure Bytes

- A C-APDU of {CLA INS P1 P2 Le = 00} is passed from the terminal to the UICC with Luicc < 256 bytes. Where YY ≤ XX, an R-APDU of {[Data(YY + ZZ)] 90 00} is returned from the UICC to the terminal. The GET RESPONSE command is sent on the same logical channel as the C-TPDU.

	Terminal		UICC
C-TPDU	[CLA INS P1 P2 P3=00]	->	
		<-	6C Luicc
	[CLA INS P1 P2 Luicc]	->	
		<-	61 XX
GET RESPONSE	[0X C0 00 00 YY]	->	
		<-	C0 [Data[Luicc]] 61 ZZ
	[0X C0 00 00 ZZ]	->	
			C0 [Data(ZZ)] 90 00

2012 Sep 17 07:50:56.361 [00] 0x1098 RUIM Debug

TX 80 F2 01 01 00 // [CLA INS P1 P2 P3=00]

07:50:56.361

2012 Sep 17 07:50:56.435 [00] 0x1098 RUIM Debug

RX 6C 12 // 6C Luicc

07:50:56.373

TX 80 F2 01 01 12 // [CLA INS P1 P2 Luicc]

07:50:56.373

RX 61 0C // 61 XX

07:50:56.384

TX 00 C0 00 00 0C // [0X C0 00 00 YY]

07:50:56.384

RX C0 84 10 A0 00 00 00 87 10 02 FF 49 FF 61 06 // C0 [Data[Luicc]] 61 ZZ

07:50:56.410

TX 00 C0 00 00 06 // [0X C0 00 00 ZZ]

07:50:56.410

RX C0 FF 89 04 03 00 00 90 00 // C0 [Data(ZZ)] 90 00

Case 3 Command

- A C-APDU of {CLA INS P1 P2 Lc [Data(Lc)]} is passed from the terminal to the UICC. An R-APDU of {90 00} is returned from the UICC to the terminal.

	Terminal		UICC
C-TPDU	[CLA INS P1 P2 P3=Lc]	->	
		<-	[INS]
C-TPDU	[Data(Lc)]	->	
		<-	90 00

2012 Sep 7 18:29:59.764 [00] 0x1098 RUIM Debug TX 00 A4 00 0C 02 // [CLA INS P1 P2 P3=Lc]

18:29:59.764 RX A4 // [INS]

18:29:59.764

2012 Sep 7 18:29:59.767 [00] 0x1098 RUIM Debug TX 6F 08 // [Data(Lc)]

18:29:59.765 RX 90 00 // 90 00

Case 4 Command

- A C-APDU of {CLA INS P1 P2 Lc [Data (Lc)] Le = 00} is passed from the terminal to the UICC. An R-APDU of {[Data(Luicc)] 90 00} is returned from the UICC to the terminal. The GET RESPONSE command is sent on the same logical channel as the C-TPDU.

	Terminal		UICC
C-TPDU	[CLA INS P1 P2 P3=Lc]	->	
		<-	[INS]
C-TPDU	[Data(Lc)]	->	
		<-	90 00

2012 Sep 7 18:29:59.658 [00] 0x1098 RUIM Debug
TX 00 A4 00 04 02 // [CLA INS P1 P2 P3=Lc]
18:29:59.658
RX A4 // [INS]
18:29:59.658
2012 Sep 7 18:29:59.662 [00] 0x1098 RUIM Debug
TX 7F FF // [Data(Lc)]
8:29:59.659
RX 61 2E // 61 Luicc
18:29:59.662
2012 Sep 7 18:29:59.669 [00] 0x1098 RUIM Debug
TX 00 C0 00 00 2E // [0X C0 00 00 Luicc]
18:29:59.664
RX C0 62 2C 82 02 78 21 84 10 A0 00 00 00 87 10 02 FF FF FF FF 89 03 02 00 00 8A 01 05 8B 03 2F 06 0C C6 0C 90 01 60 83 01 01 83 01 0A 83 01 81 90 00 // C0 [Data[Luicc]] 90 00

Case 4 Command – Using '61' Procedure Byte

- A C-APDU of {CLA INS P1 P2 Lc [Data Lc] Le = 00} is passed from the terminal to the UICC. An R-APDU of {[Data(XX + YY)] 90 00} is returned from the UICC to the terminal. The GET RESPONSE command is sent on the same logical channel as the C-TPDU.

	Terminal		UICC
C-TPDU	[CLA INS P1 P2 P3=Lc]	->	
		<-	[INS]
	[Data(Lc)]	->	
		<-	61 XX
GET RESPONSE	[0X C0 00 00 XX]	->	
		<-	C0 [Data[Luicc]] 61 YY
	[0X C0 00 00 YY]	->	
			C0 [Data(YY)] 90 00

```
2012 Sep 17  07:50:55.326  [00]  0x1098  RUIM Debug
                                        TX      00 A4 08 04 02          // [CLA INS P1 P2 P3=Lc]
                07:50:55.326
2012 Sep 17  07:50:55.405  [00]  0x1098  RUIM Debug
                                        RX      A4                      // [INS]
                07:50:55.336
                                        TX      2F E2                   // [Data(Lc)]
                07:50:55.336
                                        RX      61 14                   // 61 XX
                07:50:55.344
                                        TX      00 C0 00 00 14          // [0X C0 00 00 XX]
                07:50:55.344
                                        RX      C0 62 1C 82 02 41 21 83 02 2F E2 A5 03 80 01 68 8A 01 05 8B 03 61 0A
                                                                        // C0 [Data[Luicc]] 61 YY
                07:50:55.379
                                        TX      00 C0 00 00 0A          // [0X C0 00 00 YY]
                07:50:55.379
                                        RX      C0 2F 06 01 80 02 00 0A 88 01 10 90 00    // C0 [Data(YY)] 90 00
```

Case 4 Command – With Warning Condition

- A C-APDU of {CLA INS P1 P2 Lc [Data Lc] Le = 00} is passed from the terminal to the UICC. An R-APDU of {[Data(Luicc)] 62 XX} is returned from the terminal to the UICC containing the data returned together with the warning status bytes. The GET RESPONSE command is sent on the same logical channel as the C-TPDU.

	Terminal		UICC
C-TPDU	[CLA INS P1 P2 P3=Lc]	->	
		<-	[INS]
	[Data(Lc)]	->	
		<-	62 XX
GET RESPONSE	[0X C0 00 00 00]	->	
		<-	6C Luicc
	[0X C0 00 00 Luicc]	->	
			C0 [Data(Luicc)] 90 00

2012 Sep 17 07:53:02.605 [00] 0x1098 RUIM Debug

TX 00 A4 00 04 02 // [CLA INS P1 P2 P3=Lc]

07:53:02.605

2012 Sep 17 07:53:02.696 [00] 0x1098 RUIM Debug

RX A4 // [INS]

07:53:02.615

TX 3F 00 // [Data(Lc)]

07:53:02.615

RX 62 00 // 62 XX

07:53:02.623

TX 00 C0 00 00 00 // [0X C0 00 00 00]

07:53:02.623

RX 6C 2B // 6C Luicc

07:53:02.634

TX 00 C0 00 00 2B // [0X C0 00 00 Luicc]

07:53:02.634

RX C0 62 29 82 02 78 21 83 02 3F 00 A5 03 80 01 68 8A 01 05 8B 03 2F 06 04 C6 12 90 01 F8 83 01 01 83 01 81 83 01 0A 83 01 0B 83 01 0C 90 00 // C0 [Data(Luicc)] 90 00

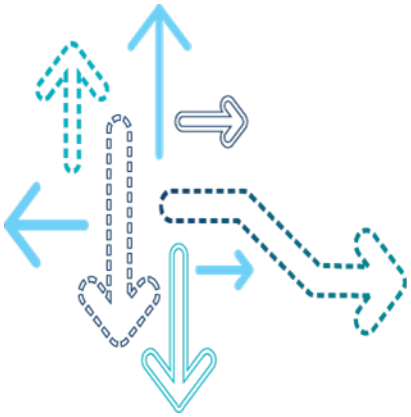
Case Commands – Code Snippet

- Each instruction case is assigned and used with the following enums. Use the following keywords for each case command to browse through related code.

```
File: uimdrv.h
typedef enum {
  UIM_INSTRN_CASE_1    = 0, /* Case 1 - No data exchanged */
  UIM_INSTRN_CASE_2    = 1, /* Case 2 - Response data only */
  UIM_INSTRN_CASE_3    = 2, /* Case 3 - Command data only */
  UIM_INSTRN_CASE_4    = 3, /* Case 4 - Response and command data */
  UIM_INSTRN_IFS_BLOCK = 4  /* Used to send IFS requests to the driver */
} uim_instrn_case_type;
```

- These instruction classes are further handled in:
  - File – uimdrv.c
  - Function – uim_send_apdu

Errors



Rx Break/Parity/Overflow Errors

- Rx break/parity/overflow errors are the common errors of hardware when interacting with the card. Since these errors are from hardware, it could only be solved from the hardware side. From the software side, we could only reduce the effects as much as possible.
- Rx break in IDLE.
  - When the Rx break appears when the UART is in the Idle state, it can be ignored. To ignore the Rx break in IDLE, we need to enable the feature FEATURE_UIM_DISABLE_RX_BREAKS_IN_IDLE.
  - QXDM message are:

//Rx break errors seen in Idle state

00:02:17.598 12612 uimdrv.c RX Break in IDLE
00:02:17.598 12612 uimdrv.c RX Break in IDLE
00:02:17.598 12612 uimdrv.c RX Break in IDLE

Rx Break/Parity/Overflow Errors

- Rx break in nonidle
  - When Rx break errors happen in nonidle, they cannot be ignored. The UIM driver will reset the card and go into recovery process when it receives the max number of Rx breaks.
  - QXDM message are:

MSG [00021/00] User Interface Module/Low 02:26:32.766 uimdrv.c 04203 uim_send_command
MSG [00021/00] User Interface Module/Low 02:26:32.767 uimgen.c 07646 UIM generic state in uim_command 33

//Rx break errors maxed out

MSG [00021/03] User Interface Module/Error 02:26:33.019 uimdrv.c 09662 maxed the Rx break error count

//ME timed out waiting for a correct card response

MSG [00021/03] User Interface Module/Error 02:26:33.019 uim.c 08756 Timed out on the command response

//Reset due to max Rx break errors seen

MSG [00021/03] User Interface Module/Error 02:26:33.019 uim.c 08831 Reset due to Rx break errors

Rx Break – IT3 logs

#	Sequence	Dir	Description	Value	Delay	Meaning
3030	PPS request	T	PPSS	FF	48.94 etu	Initial character
		T	PPS0	10	13.06 etu	Following parameter chars: PPS1 Transfer protocol: T=0
		T	PPS1	95	13.06 etu	F = 512 D = 16
		T	PCK	7A	13.06 etu	Check character
3031	PPS response	C	PPSS	FF	16.03 etu	Initial character
		C	PPS0	10	12.01 etu	Following parameter chars: PPS1 Transfer protocol: T=0
		C	PPS1	95	12.01 etu	F = 512 D = 16
		C	PCK	00	139.56 etu	Check character
3032	Unknown	T	Incoming data	00 00 00 A4 00	34.88 etu	
3033	Unknown	T	Incoming data	04 02	13.06 etu	
		C	Outgoing data	A4		
3034	Unknown	T	Incoming data	3F 00	57.41 etu	
		C	Outgoing data	61 23		
3035	APDU	T	Header [CLA INS P1 P2]	00 C0 00 00	58.66 etu	GET RESPONSE
		C	Outgoing data	62 21 82 02 78 21 83 02 3F 00 A5 07 80 01 71 83 02 5F 40 8A 01 05 8B 03 2F 06 02 C6 06 90 01 00 83 01 01		
		C	Return code [SW1 SW2]	90 00		

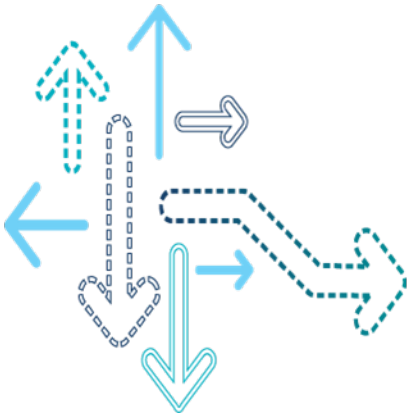
ISO Protocol Layer

#	Sequence	Dir	Description	Value	Delay	Meaning
3043	APDU	T	Header [CLA INS P1 P2]	00 88 00 81	181.81 etu	AUTHENTICATE
		T	Incoming data	10 7C 77 06 C7 6B 53 6E 7F B6 31 5F E2 57 E5 F0 46 10 69 6A CC 99 DF 21 00 00 C8 61 DB DE 7E 50 74		
		C	Return code [SW1 SW2]	5A 00		
3044	Unknown	C	Outgoing data	00 00 00	34.88 etu	
3045	Unknown	T	Incoming data	00 00	69.75 etu	
		C	Outgoing data	00		
3046	Unknown	T	Incoming data	00	23.25 etu	
3047	Power off		Event			Power off
3048	Cold reset		Event			Reset card interface

Rx Break Errors – Code Snippet

```
Get the UART status
   uart_status = UIM_READ_STATUS (), uart_misr_status = UIM_READ_MISR ();
Check if the status is RX Break
    if ((uart_status & (MSMU_SR_RXBREAK))|| (uart_misr_status & MSMU_ISR_RXBREAK))
Increase the counter of RX Breaks
   uim_tot_rx_break_error_count++;
If the counter exceeds UIM_MAX_RX_BREAK_ERR_COUNT, trigger a card error to upper layer
   module.
File:uimdrv.c
Function: uim_rx_error
   if ( uim_rx_break_error_count > UIM_MAX_RX_BREAK_ERR_COUNT )
         {
        ……
            MSG_ERROR_DS ( uim_drv_slot,"Maxed the Rx break error count",0,0,0);

            /* Disable the receiver to avoid interrupts */
            uim_reset_uart(uim_drv_slot);
            /* Set the command response timeout signal */
            (void) rex_set_sigs( &uim_tcb, UIM_CMD_RSP_TIMEOUT_SIG );//card error is triggered
     after set the siginal    UIM_CMD_RSP_TIMEOUT_SIG
```


Command Response Timeout



Command Response Timeout Overview

- Timeout for ATR
  - The response to the reset is expected from the UIM before 40,000 UIM clock cycles; otherwise, the reset failed. The UIM driver does not process a timer for this activity. This time is defined as UIM_ATR_TIMEOUT_MS. The lowest voltage class that is supported is started first. In this case, the process starts with Class C (1.8 V) and waits for the initial work waiting time for the ATR. If there is a timeout because an ATR was not received, or a corrupted ATR was received three times, it is assumed that the card does not support that voltage class and the process switches to the next available voltage class, i.e., Class B (3.0 V).
- Timeout for Command Response
  - UIM_RSP_TIMEOUT_SIG – This is a signal associated with the UIM_TIMER that is set when a command is sent to the UIM that requires a response. This signal gets set upon the expiration of the timer.
  - UIM_TRANSACTION_SIG – This signal is set when some task votes on power-down of the UIM. This is mainly used for handling infinite NULL byte scenario.
- See *Universal Mobile Telecommunications System (UMTS); USIM Conformance Test Specification* (3GPP TS 31.122); per specification, the value of the WWT shall not exceed 960 × WI × Fi/f. If this is exceeded, a timeout is declared within UIM code.

Command Response Timeout – QXDM Logs

//UIM received Poll command

23:49:28.736 uimgen.c 05464 Setting UIM POLL timer following a poll
23:49:28.736 uimdrv.c 05657 uim_send_command
23:49:28.737 uimgen.c 08790 UIM generic state in uim_command 11

//UIM response time out signal

23:49:30.021 uim.c 10634 UIM received response time out signal 0x8 0x3065fd
23:49:30.021 uim.c 10707 Timed out on the command response
23:49:30.021 uim.c 10958 UIM timeout in internal command

//UIM powering down and entering recovery

23:49:30.508 uimdrv.c 06369 uim_power_down
23:49:30.510 uimdrv.c 16777 uim_clk_busy_wait large value 2000
23:49:30.525 uimgen.c 02573 Received internal Wakeup command, UIM Entering Recovery

//UIM power up and reset

23:49:30.525 uimdrv.c 04291 uim power up at 1.8 V
23:49:30.525 gstk.c 07988 Received Recovery Sig – Done
23:49:30.547 uimdrv.c 04587 uim_reset

Command Response Timeout – QXDM Logs (cont.)

//Time out

23:49:33.450	uim.c	10634	HOTSWAP – UIM received response time out signal 0x8 0x3065fd

//UIM Power down

23:49:33.451	uimdrv.c	06340	uim power down at 1.8 V
23:49:33.451	uimdrv.c	06369	uim_power_down

//This means we either received no ATR or a corrupted ATR and ultimately we power down.

23:49:33.490	uim.c	11046	Bad ATR in WAKE_UP_F -> Resetting + UIM POLL timer reset
23:49:33.490	uim.c	11103	Entering UIM_MAX_NUM_ATTEMPTS
23:49:33.490	uimdrv.c	06340	uim power down at1.8 V
23:49:33.490	uimdrv.c	06369	uim_power_down
3:49:33.493	mmgsdi.c	06710	GSDI_UIM_ERROR_SIG received
23:49:33.493	mmgsdi_evt.c	02258	MMGSDI_CARD_ERR_NO_ATR_RCVD_AFTER_RESET

Command Response Timeout – Code Snippet

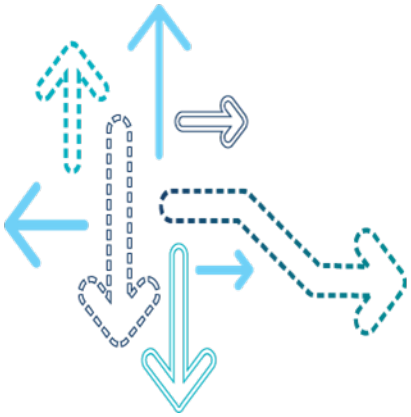
File: uim.c
Function: uim_task

```
.....
      if  ((uim_card_status           == UIM_CARD_REMOVED)/* ||
           (uim_util_query_card_status == UIM_CARD_REMOVED) ||
           (uim_hotswap_debounce_in_progress == TRUE)*/)
…
 /* The status of UIM is set to Removed state */
        uim_status  = UIM_ERR_S;
….
….
 if ((rex_signals_mask & UIM_UNSOLICITED_RSP_SIG) != 0)
     {
       (void) rex_clr_sigs( &uim_tcb, UIM_UNSOLICITED_RSP_SIG );
       ERR ("Received un-solicited byte", 0, 0, 0);
     }
     else
     {
       ERR ("Timed out on the command response", 0, 0, 0);
       (void) rex_clr_sigs( &uim_tcb, UIM_CMD_RSP_TIMEOUT_SIG );
       (void) rex_clr_sigs( &uim_tcb, UIM_TRANSACTION_SIG );
     }
```


Command Response Timeout – Code Snippet (cont.)

```
File: uim.c
Function: uim_task
…..
       if  ((uim_card_status            == UIM_CARD_REMOVED)/* ||
             (uim_util_query_card_status == UIM_CARD_REMOVED) ||
             (uim_hotswap_debounce_in_progress == TRUE)*/)
…
 /* The status of UIM is set to Removed state */
         uim_status  = UIM_ERR_S;
….
….
 if ((rex_signals_mask & UIM_UNSOLICITED_RSP_SIG) != 0)
      {
        (void) rex_clr_sigs( &uim_tcb, UIM_UNSOLICITED_RSP_SIG );
        ERR ("Received un-solicited byte", 0, 0, 0);
      }
      else
      {
        ERR ("Timed out on the command response", 0, 0, 0);
        (void) rex_clr_sigs( &uim_tcb, UIM_CMD_RSP_TIMEOUT_SIG );
        (void) rex_clr_sigs( &uim_tcb, UIM_TRANSACTION_SIG );
      }
```

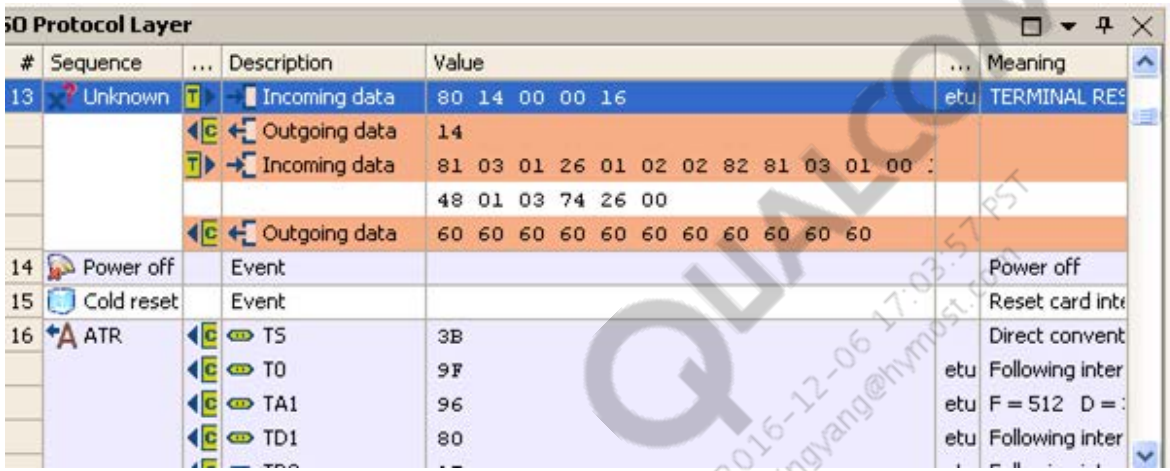
FEATURE_UIM_STOP_INFINITE_NULL

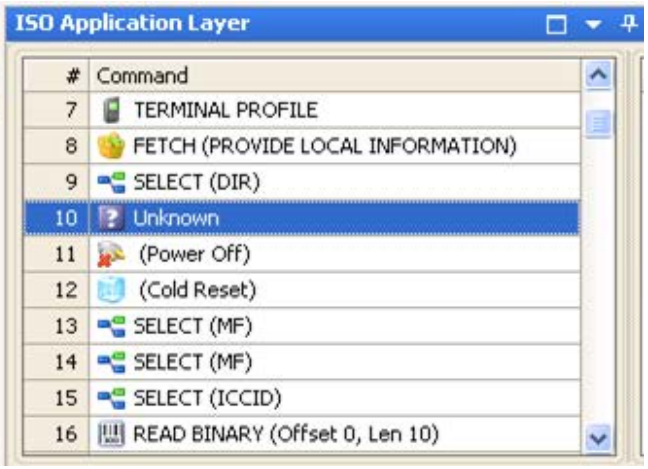


FEATURE_UIM_STOP_INFINITE_NULL Overview

- Some erroneous cards send infinite NULL bytes in a nonstop fashion while processing the command.
- This feature was developed to enable UIM software to reset the card interface if a card continuously sends NULL procedure bytes.
- If we disable this macro, Terminal will not reset the card interface if a card continuously sends NULL procedure bytes. Disabling this macro will have no side effects.

FEATURE_UIM_STOP_INFINITE_NULL – IT3 Logs





Reset after received 10 NULL bytes

FEATURE_UIM_STOP_INFINITE_NULL – QXDM Logs

TX 80 14 00 00 16

00:07:10.898

RX 14

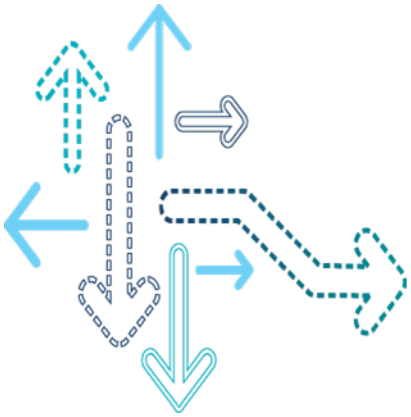
TX 81 03 01 26 01 02 02 82 81 03 01 00 14 08 3A 55 48 01 03 74 26 00

//NULL bytes

RX 60 60 60 60 60 60 60 60 60 60 60 3B 9F 96 80 1F C7

MSG	[00021/02] User Identity Module/High	00:07:32.485	uim.c	06136	Internal command to Reset the UIM
MSG	[00021/00] User Identity Module/Low	00:07:32.500	uimdrv.c	03110	uim_power_up
MSG	[00021/00] User Identity Module/Low	00:07:32.522	uimdrv.c	03377	uim_reset

FEATURE_HANDLE_UNKNOWN_ACK_BYTE



FEATURE_HANDLE_UNKNOWN_ACK_BYTE Overview

- This feature was developed for a nonspec-compliant card which could not be recognized if NV 896 was set to 1. In the case of some older GSM card, the Status byte (SW1 SW2) would return an unknown acknowledgement byte' and not a known byte that makes the card get recognized.

NV#896 (NV_UIM_FIRST_INST_CLASS_I):-
NV 896 -> 0 for 2G Card
NV 896 -> 1 for 3G Card

This card responded for the first 'Select'/'Get Response' command in the following way:
APDU command:–
TX 00 A4 00 04 02
RX A4 - Correct ACK received (Correct behavior would be to get a status word bytes as 6E 00 i.e. CLA not supported at this point since it's a GSM card, but somehow this card responds with the correct ACK at this point)

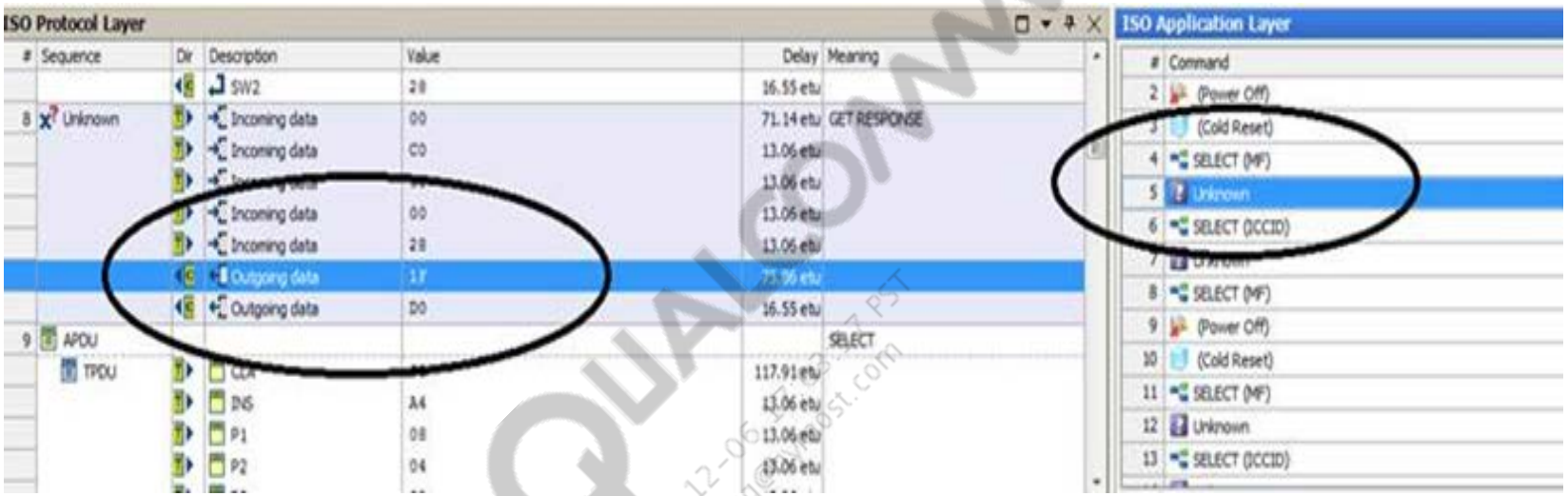
TX 3F 00
RX 61 28
TX 00 C0 00 00 28
RX 1F D0 - Here card returns an incorrect response for the GET RESPONSE command

FEATURE_HANDLE_UNKNOWN_ACK_BYTE Overview (cont.)

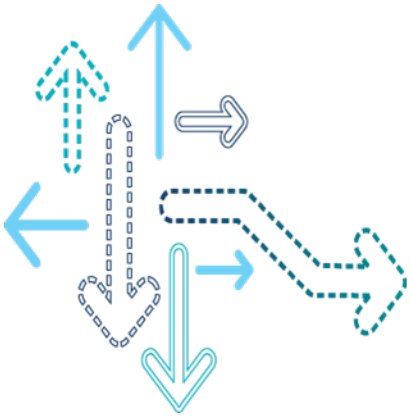
- As a workaround for this nonspec-compliant card, FEATURE_HANDLE_UNKNOWN_ACK_BYTE was developed which enabled the CLA to be toggled from USIM to GSM when an unknown ack byte (in this case 1F D0) was returned. Once the CLA is toggled on receiving the unknown ack byte, the card responds correctly.
- Code snippet

```
(@Function RX_ISR_PROC_PROCEDURE_BYTES_DM)
#ifdef FEATURE_HANDLE_UNKNOWN_ACK_BYTE
                        /* Toggle Instruction class for one more try */
                        uim_toggle_instrn_class = !uim_toggle_instrn_class;
                        return;
#endif /* FEATURE_HANDLE_UNKNOWN_ACK_BYTE */
```


FEATURE_HANDLE_UNKNOWN_ACK_BYTE – IT3 Logs

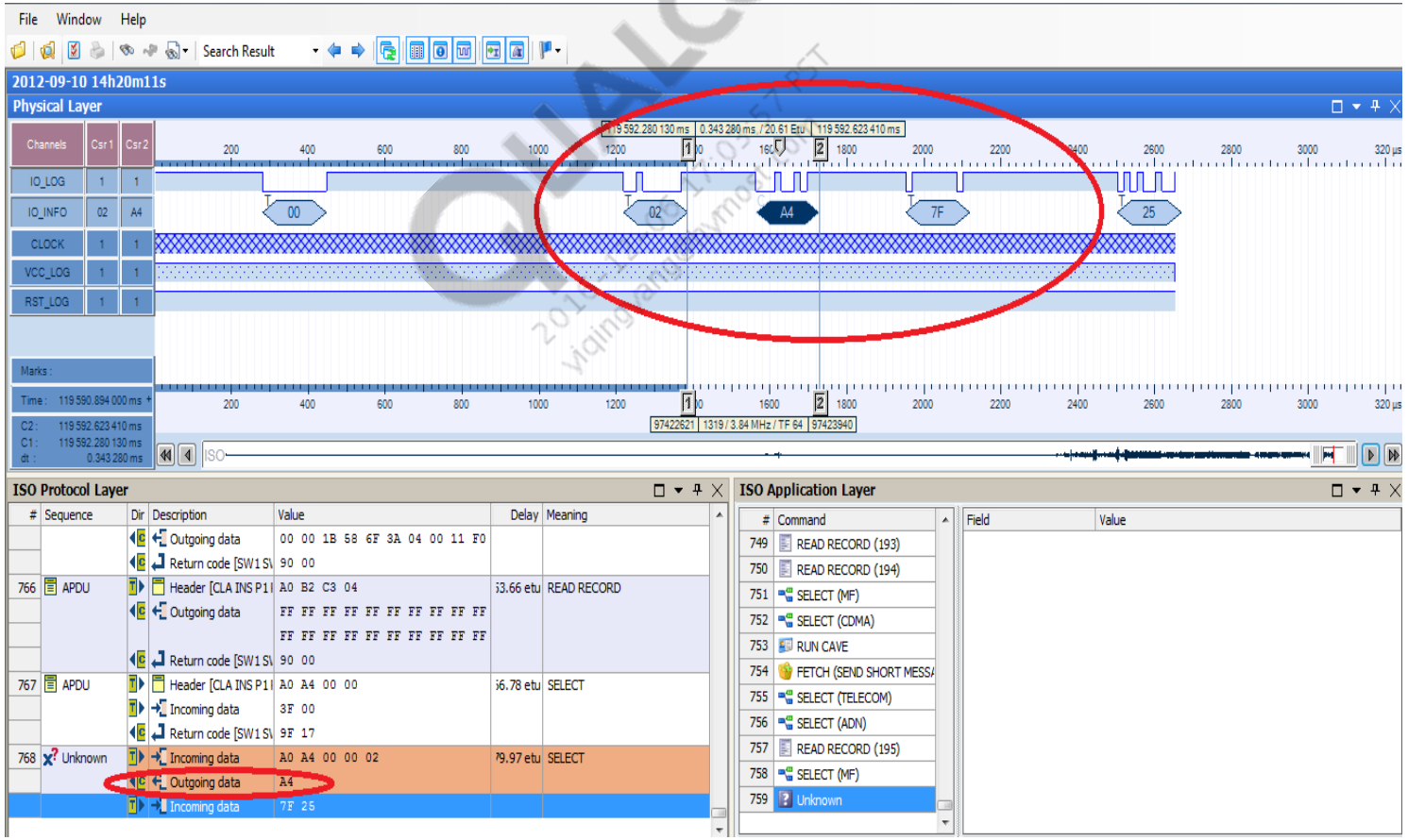


FEATURE_UIM_AUTH_CDMA_DF_FIRST

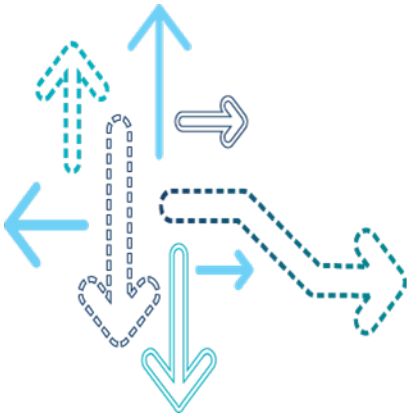


FEATURE_UIM_AUTH_CDMA_DF_FIRST Overview

- Some cards have GSM and RUIM application. GSM application will be selected and MMGSDI initializes files in the GSM application at first. Some cards do not respond well if the DF_CDMA is not selected prior to DF_GSM. Once we enable this feature, UIM will select DF_CDMA first.



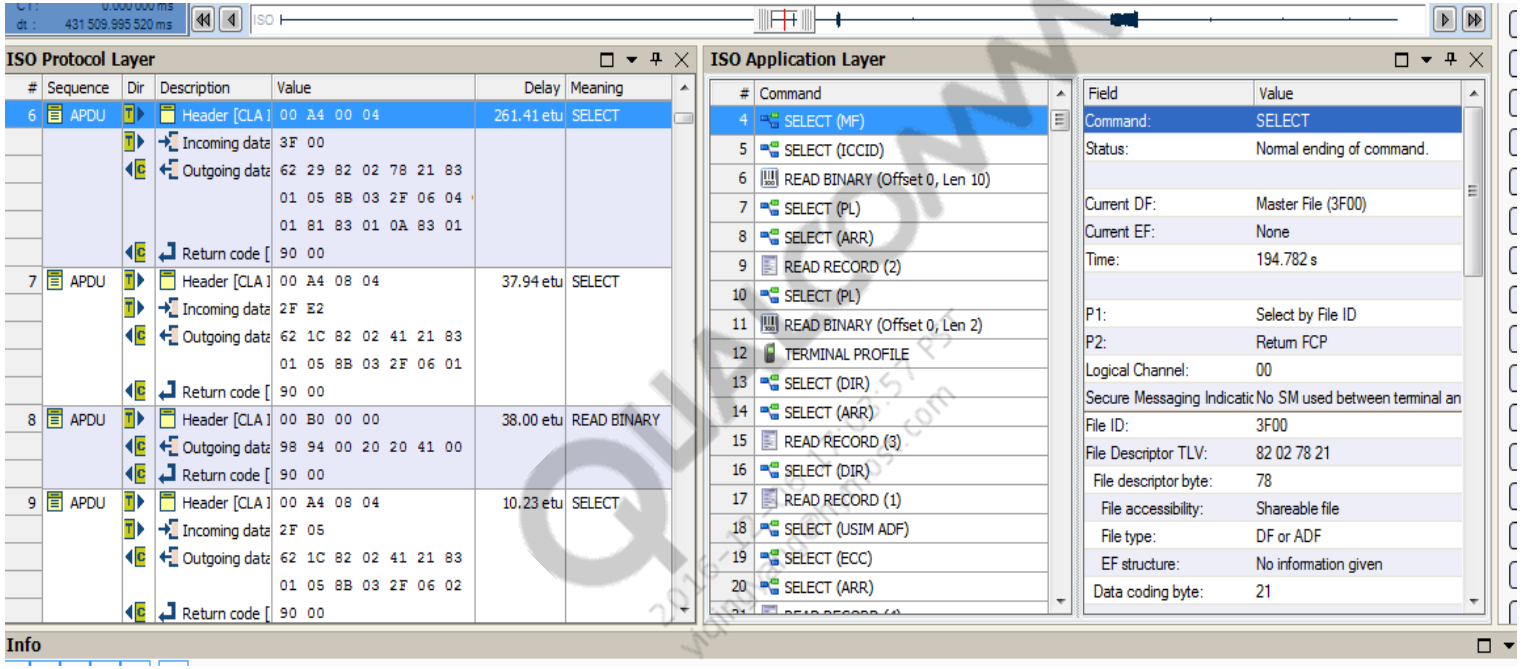
Directory Maintenance



Directory Maintenance

- The UIM server keeps track of the last MF, ADF, DF, and EF accessed. This is required for checking to see if the DF indicated in response to the STATUS command is the same as that which was indicated in the previous response. It is also used to optimize command processing of the ME.
- It also keeps track of the last selected DF so that for selecting all EFSs under one particular DF, the DF does not have to be selected again. The last selected DF1, DF2, DF3, and EF are maintained in a data structure to aid in directory maintenance and are initialized to UIM_NO_FILE_SEL (0xFFFF) on task startup. When a first-level DF is selected, the second-level DF and EF are cleared. Similarly, when a second-level DF is selected, the last selected EF is cleared. The ADF is treated as a first-level DF. When the last selected DF does not match the requested DF, the entire path is selected starting from the MF. The most optimal path is selected based on the current directory, and a redundant selection is not made. The last selected directory structure is maintained separately for each slot in the case of dual slots.

Directory Maintenance – Selecting Files Under MF



Selecting the Master File (3F00)

1980 Jan 6 00:29:38.998 [00] 0x1098 RUIM Debug

RX A4

00:29:38.990
TX 3F 00

00:29:38.990
RX 90 00

Directory Maintenance – Selecting Files Under MF (cont.)

Selecting ICCID (2FE2)

1980 Jan 6 00:29:51.169 [00] 0x1098 RUIM debug

RX A4

00:29:51.111
TX 2F E2

00:29:51.113
RX 61 1E

00:29:51.119
TX 00 C0 00 00 1E

00:29:51.120
RX C0 62 1C 82 02 41 21 83 02 2F E2 A5 03 80 01 68 8A 01 05 8B 03 2F 06 01 80 02 00 0A 88 01 10 90 00

Selecting EF ARR within MF (2F06)

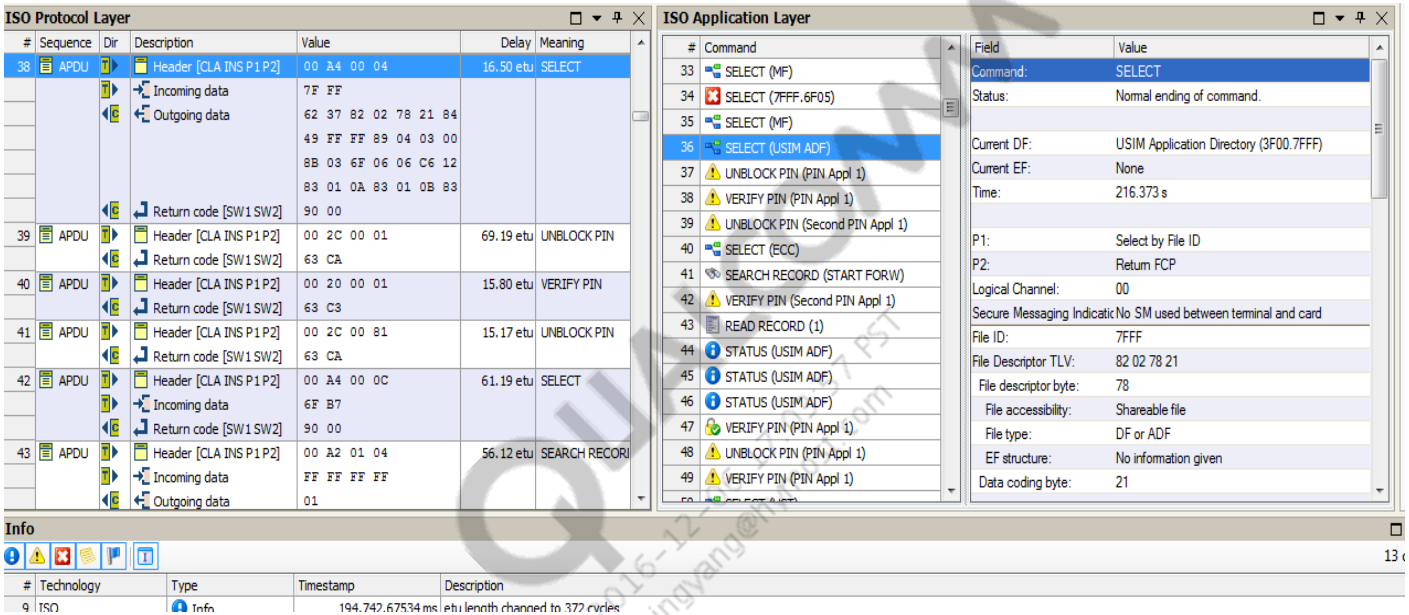
1980 Jan 6 00:29:51.185 [00] 0x1098 RUIM debug

RX A4

00:29:51.179
TX 2F 06

00:29:51.179
RX 90 00

Directory Maintenance – Selecting Files Under USIM ADF



Selecting USIM application directory\

1980 Jan 6 00:29:39.320 [00] 0x1FEB Extended Debug Message
uim.c 4495 L ADF selected
1980 Jan 6 00:29:39.320 [00] 0x1FEB Extended Debug Message
uim.c 4497 L Last sel DF2,EF ffff ffff

Selecting the EF ECC (6FB7) from the USIM application directory

1980 Jan 6 00:29:39.320 [00] 0x1FEB Extended Debug Message
uim.c 4507 L New path ffff ffff 6fb7
1980 Jan 6 00:29:39.338 [00] 0x1098 RUIM Debug
RX A4
00:29:39.330
TX 6F B7
00:29:39.330
RX 90 00

Directory Maintenance – Selecting Files Under USIM ADF (cont.)

Selecting EF UST (6f38) from the USIM application directory

1980 Jan 6 00:29:47.230 [00] 0x1FEB Extended Debug Message
uim.c 4495 L ADF selected

Drop count = 0

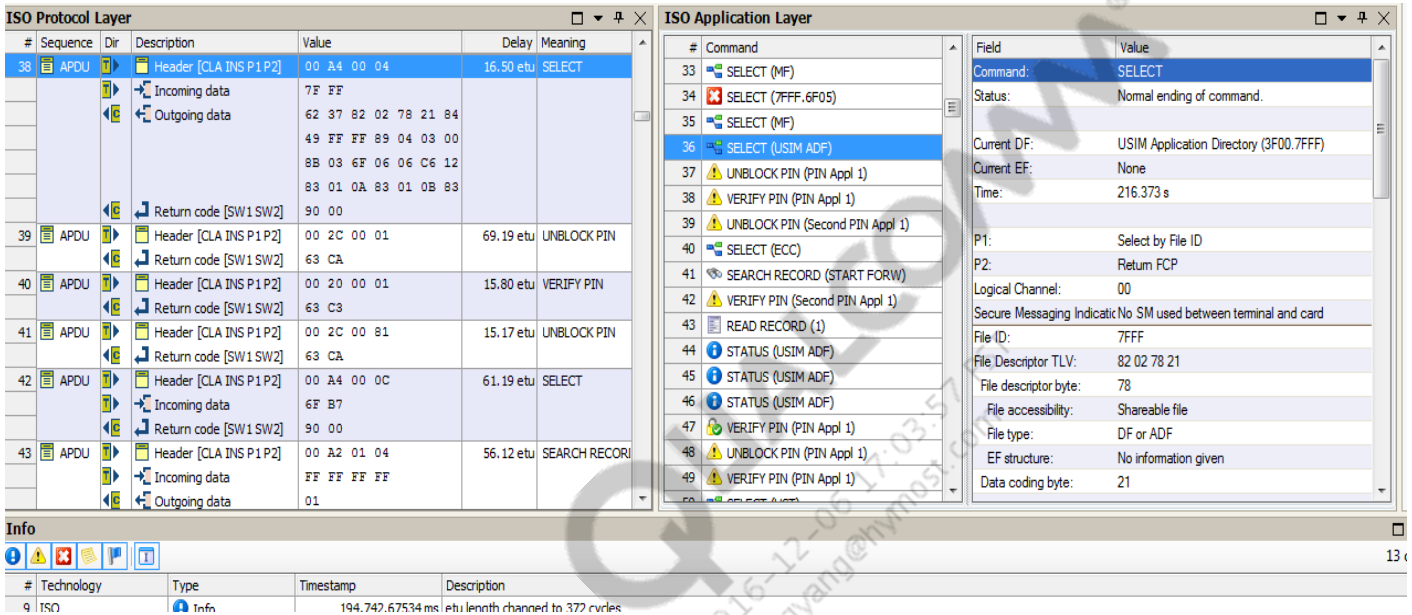
1980 Jan 6 00:29:47.230 [00] 0x1FEB Extended Debug Message
uim.c 4497 L Last sel DF2,EF ffff 6fb7

Drop count = 0

1980 Jan 6 00:29:47.230 [00] 0x1FEB Extended Debug Message
uim.c 4507 L New path ffff ffff 6f38

1980 Jan 6 00:29:47.296 [00] 0x1098 RUIM Debug
RX A4
00:29:47.240
TX 6F 38
00:29:47.240
RX 61 1E

Directory Maintenance – Selecting Files Under USIM ADF (cont.)



Selecting USIM application directory

1980 Jan 6 00:29:39.320 [00] 0x1FEB Extended Debug Message
uim.c 4495 L ADF selected
1980 Jan 6 00:29:39.320 [00] 0x1FEB Extended Debug Message
uim.c 4497 L Last sel DF2,EF ffff ffff

Selecting the EF ECC (6FB7) from the USIM application directory

1980 Jan 6 00:29:39.320 [00] 0x1FEB Extended Debug Message
uim.c 4507 L New path ffff ffff 6fb7
1980 Jan 6 00:29:39.338 [00] 0x1098 RUIM Debug
RX A4
00:29:39.330
TX 6F B7
00:29:39.330
RX 90 00

Directory Maintenance – Code Snippets

- The UIM provides the ability to internally select the required DF before processing external commands that require a particular DF to be selected using UIM_SELECT_F command.
- The last selected DF1, DF2, DF3, and EF are maintained in a data structure to aid in directory maintenance and are initialized to UIM_NO_FILE_SEL (0xFFFF) on task startup.

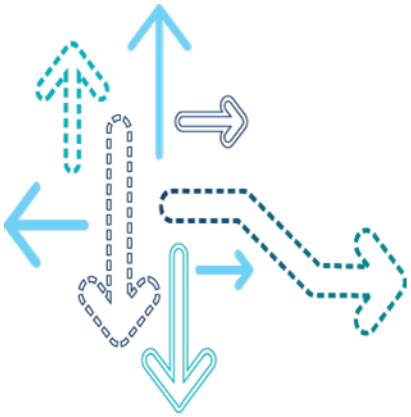
uim_last_sel_dir

```
typedef struct {
uim_df1_type              df1;              /* Last first level DF selected */
uim_dir_type              df2;              /* Last second level DF selected */
uim_dir_type              df3;              /* Last third level DF selected */
uim_dir_type              ef;               /* Last EF selected */
uim_file_type             df1_type;         /* File type of last selected DF */
boolean                   ef_sel;           /* Check to see if an EF has been selected under DF */

#ifdef                    curr_app;         Current application whose session is open even if we
   are not inside that AID of that Application */

FEATURE_UIM_USIM
uim_aid_type
#endif
curr_app;
Current application whose session is open even if we are not inside that AID of that Application */
uim_last_sel_ef_type ef_info;
} uim_last_sel_dir;
```


Directory Maintenance – Code Snippets (cont.)

- The function boolean uim_directory_current() takes the command as input and determines if the current directory to execute the command is valid. This function returns a Boolean as the result. Also, this function makes a call to the function uim_set_select_path(), which determines the optimal path when the directory is not current. If the current directory is not valid, the external command will be requeued in the front of the command queue, a path list will be built depending on the directory to be accessed, and an internal command to SELECT the directory will be generated.

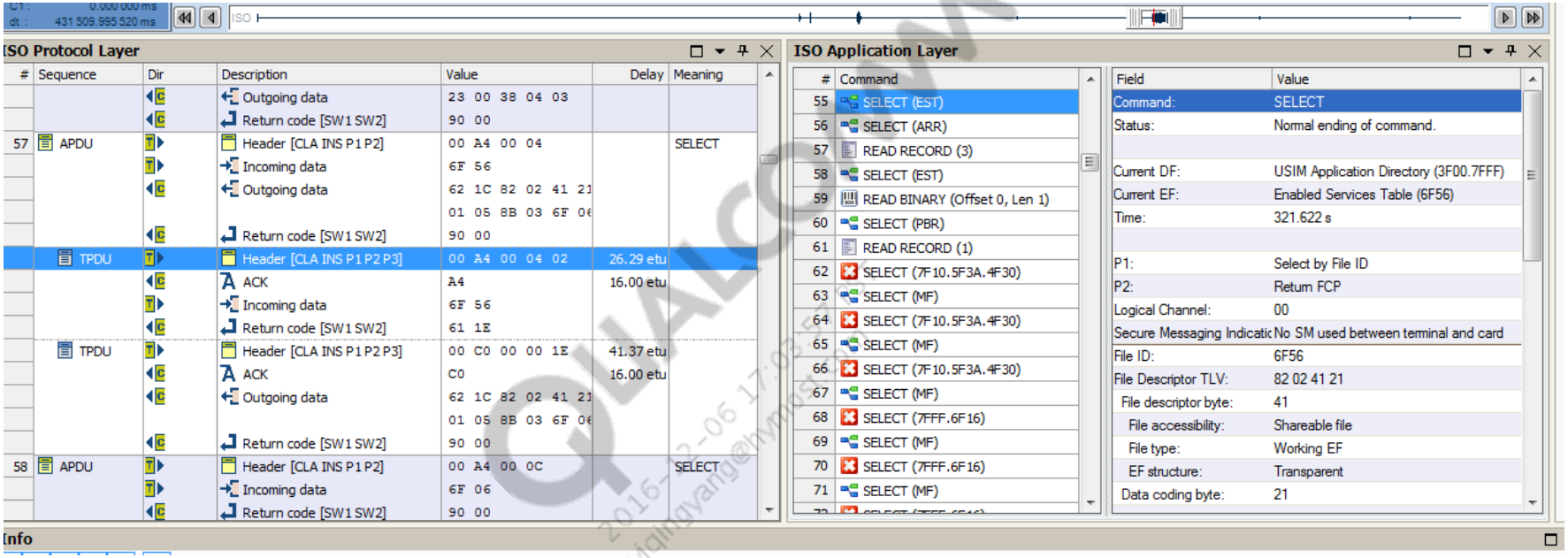
APDU vs. TPDU



APDU and TPDU Overview

- Application Protocol Data Unit (APDU)
  - The application protocol consists of an ordered set of exchanges between the application layer and the transport layer of the terminal. Application protocols are defined in subsequent parts of this document.
  - Each step in an application layer exchange consists of a command-response pair, where the application layer of the terminal sends a command to the UICC via the transport layer of the terminal, and the UICC processes it and sends a response to the application layer of the terminal using the transport layer of the UICC and the transport layer of the terminal. Each specific command (C-APDU) has a specific response (R-APDU).
- Transfer Protocol Data Unit (TPDU)
  - The mapping of the C-APDU onto the T = 0 command header is dependent upon the case of the command. The mapping of the data (if present) and status returned by the UICC onto the R-APDU is dependent upon the length of the data returned.
  - Procedure bytes '61XX' and '6CXX' are returned by the UICC to control exchanges between the transport layer of the terminal and the UICC, and should never be returned to the application layer of the terminal. Command processing in the UICC is not complete if it has returned procedure bytes '61XX' or '6CXX'.

IT3 Logs



QXDM Logs

QXDM LOG SNIPPETS

// Select TPDU

00:29:47.415

TX 00 A4 00 04 02

00:29:47.419

1980 Jan 6 00:29:47.485 [00] 0x1098 RUIM Debug

RX A4

00:29:47.429

TX 6F 56

00:29:47.429

RX 61 1E

//Get Response TPDU

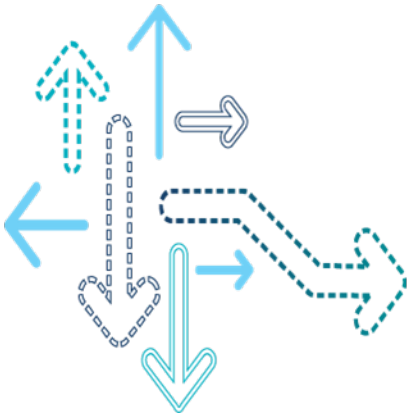
00:29:47.435

TX 00 C0 00 00 1E

00:29:47.435

RX C0 62 1C 82 02 41 21 83 02 6F 56 A5 03 80 01 68 8A 01 05 8B 03 6F 06 03 80 02 00 01 88 01 28 90 00

New APDU sample - 0x19B7



New APDU sample - 0x19B7

// COLD RESET

1980 Jan 6 03:06:51.261 [B0] 0x19B7 UIM APDU

Version = 1
Sequence Number = 1
Slot Id = SLOT_1
Message Type = COLD_RESET
Cold Reset {
    Clock = CLOCK_4P8_MHz
    Voltage = VOLTAGE_1P8
}

// WARM RESET

1980 Jan 6 03:06:51.261 [B0] 0x19B7 UIM APDU

Version = 1
Sequence Number = 1
Slot Id = SLOT_1
Message Type = WARM_RESET
Cold Reset {
    Clock = CLOCK_4P8_MHz
    Voltage = VOLTAGE_1P8
}

// POWER OFF

1980 Jan 6 03:06:51.248 [1B] 0x19B7 UIM APDU

Version = 1
Sequence Number = 0
Slot Id = SLOT_1
Message Type = POWER_OFF
Power Off {
    Clock = CLOCK_00_MHz
    Voltage = VOLTAGE_0P0
}

New APDU sample - 0x19B7 (cont.)

// ATR Sequence

1980 Jan 6 03:06:51.290 [3B] 0x19B7 UIM APDU

Version = 1
Sequence Number = 2
Slot Id = SLOT_2
Message Type = RX
RX Data = 3B

1980 Jan 6 03:06:51.370 [48] 0x19B7 UIM APDU

Version = 1
Sequence Number = 3
Slot Id = SLOT_2
Message Type = RX
RX Data = 9F 94 80 1F C7 80 31 E0 73 FE 21 1B 57 37 86 60 A3 02 88 51 4B

New APDU sample - 0x19B7 (cont.)

// PPS Sequence – QXDM Logs

1980 Jan 6 03:06:51.371 [D5] 0x19B7 UIM APDU

Version	= 1
Sequence Number	= 4
Slot Id	= SLOT_2
Message Type	= TX
TX Data	= FF 10 94 7B

1980 Jan 6 03:06:51.381 [9C] 0x19B7 UIM APDU

Version	= 1
Sequence Number	= 5
Slot Id	= SLOT_2
Message Type	= RX
RX Data	= FF 10 94 7B

New APDU sample - 0x19B7 (cont.)

// Case 1 Command

1980 Jan 6 03:07:19.489 [A2] 0x19B7 UIM APDU

Version = 1
Sequence Number = 642
Slot Id = SLOT_2
Message Type = TX
TX Data = 80 F2 00 0C 00

1980 Jan 6 03:07:19.491 [6B] 0x19B7 UIM APDU

Version = 1
Sequence Number = 643
Slot Id = SLOT_2
Message Type = RX
RX Data = 90 00

New APDU sample - 0x19B7 (cont.)

// Case 2 Command

1980 Jan 6 03:06:54.705 [9C] 0x19B7 UIM APDU

Version = 1
Sequence Number = 264
Slot Id = SLOT_2
Message Type = TX
TX Data = 80 F2 01 01 12

1980 Jan 6 03:06:54.725 [FE] 0x19B7 UIM APDU

Version = 1
Sequence Number = 265
Slot Id = SLOT_2
Message Type = RX
RX Data = F2 84 10 A0 00 00 00 87 10 02 FF 91 FF 01 89 B0 00 01 00 90 00

New APDU sample - 0x19B7 (cont.)

// Case 3 Command

1980 Jan 6 03:06:51.539 [71] 0x19B7 UIM APDU

Version = 1
Sequence Number = 28
Slot Id = SLOT_2
Message Type = TX
TX Data = 80 10 00 00 21

1980 Jan 6 03:06:51.542 [10] 0x19B7 UIM APDU

Version = 1
Sequence Number = 29
Slot Id = SLOT_2
Message Type = RX
RX Data = 10

New APDU sample - 0x19B7 (cont.)

// Case 3 Command

1980 Jan 6 03:06:51.542 [10] 0x19B7 UIM APDU

Version = 1
Sequence Number = 30
Slot Id = SLOT_2
Message Type = TX
TX Data = FF FF FF FF 7F 9F 00 DF FF 03 02 1F E2 00 00 00 C3 FB 00 07 04 11 78 00 71 01 00 00 00 38 02 80 03

1980 Jan 6 03:06:51.766 [42] 0x19B7 UIM APDU

Version = 1
Sequence Number = 31
Slot Id = SLOT_2
Message Type = RX
RX Data = 91 39

New APDU sample - 0x19B7 (cont.)

// Case 4 Command

1980 Jan 6 03:06:51.769 [50] 0x19B7 UIM APDU
Version = 1
Sequence Number = 34
Slot Id = SLOT_2
Message Type = TX
TX Data = 00 A4 08 04 02

1980 Jan 6 03:06:51.771 [96] 0x19B7 UIM APDU
Version = 1
Sequence Number = 35
Slot Id = SLOT_2
Message Type = RX
RX Data = A4

1980 Jan 6 03:06:51.771 [96] 0x19B7 UIM APDU
Version = 1
Sequence Number = 36
Slot Id = SLOT_2
Message Type = TX
TX Data = 2F 00

New APDU sample - 0x19B7 (cont.)

// Case 4 Command

1980 Jan 6 03:06:51.784 [23] 0x19B7 UIM APDU
Version = 1
Sequence Number = 37
Slot Id = SLOT_2
Message Type = RX
RX Data = 61 1C

1980 Jan 6 03:06:51.784 [23] 0x19B7 UIM APDU
Version = 1
Sequence Number = 38
Slot Id = SLOT_2
Message Type = TX
TX Data = 00 C0 00 00 1C

1980 Jan 6 03:06:51.791 [FE] 0x19B7 UIM APDU
Version = 1
Sequence Number = 39
Slot Id = SLOT_2
Message Type = RX
RX Data = C0 62 1A 82 05 42 21 00 28 01 83 02 2F 00 8A 01 05 8B 03 2F 06 01 80 02 00 28 88 01 F0 91 39

Code Snippet

- Function – uim_send_apdu
  - This function starts the transfer of an APDU to the UIM and if successful is finished by the uim_rx_is.
- Function – uim_send_command
  - This function starts the process of sending an APDU command to the UIM. Once the APDU command is sent, the UIM driver expects the APDU response. The UIM driver can initiate a GET RESPONSE command as the result of an APDU response. The GET RESPONSE command results in response data received by the UIM driver. Also, the APDU response can indicate the need for a FETCH command. The UIM driver also handles FETCH commands. This function sets up the APDU command header, command data buffer, command data size, response data buffer, response data size, status buffer, and response callback routine. It sets the appropriate UIM driver state, then sends the first byte of the APDU header.
- Function – RX_ISR_SW1_RESP_END_DM
  - This function runs within the context of the Rx ISR of the UART; it is called when expecting the second status word after a response end status word.

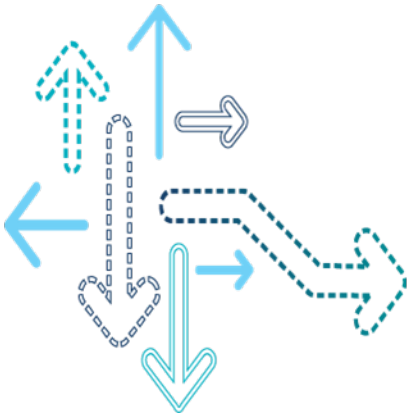
```
LOCAL void rx_isr_sw1_resp_end_dm
(
  uim_drv_slot_type uim_drv_slot,  /* Slot control variable */
  byte sw2
)
{
 /* Indicate APDU result */
      resp_buf[uim_drv_slot]->cmd_status = UIM_DONE;
   /* Send the GET RESPONSE command */
      uim_send_apdu ( &cmd_req_ptr[uim_drv_slot] );
```

Code Snippet (cont.)

- FUNCTION – UIM_GENERIC_COMMAND_RESPONSE
  - This procedure processes the response to a generic command that has been received from the UIM.

```
case UIM_GET_RESPONSE_ST:
      {
        MSG_HIGH_UIM ( uim_drv_slot,"Sending Get Response", 0,0,0);
        if ((cmd_ptr->hdr.protocol == UIM_UICC) ||
            (cmd_ptr->hdr.protocol == UIM_WCDMA))
        {
          uim_req_buf_static_ptr->apdu_hdr.uim_class =
            cmd_ptr->stream_iso7816_apdu.cmd_data[UIM_7816_APDU_CLASS_OFFSET] &
  UIM_GET_RESPONSE_CLA_MASK;
        }        else
       {
         uim_req_buf_static_ptr->apdu_hdr.uim_class = 0xA0;
       }
        uim_req_buf_static_ptr->apdu_hdr.instrn = GET_RESPONSE;
        uim_req_buf_static_ptr->instrn_case = UIM_INSTRN_CASE_2;
        uim_req_buf_static_ptr->apdu_hdr.p1 = 0x00;
        uim_req_buf_static_ptr->apdu_hdr.p2 = 0x00;
        uim_req_buf_static_ptr->apdu_hdr.p3 = uim_get_resp_sw2;
        uim_send_command(uim_req_buf_static_ptr);
        /* Move to Stream APDU state to process the response */
        ++uim_generic_state_ptr; }
```

Hotswap



Description

- The term hotswap denotes the ability to insert or remove a SIM card from the mobile equipment, thereby losing and reacquiring service without requiring a device reboot.
- Hotswap enablement and the card detect GPIO configuration are controlled by EFS 70210.
- The card detect GPIO is configured by UIM initialization software, as shown in Table 2-1 and Table 2-2. These runtime-configurable items are part of EFS item 70210. The following is an example:

Table 2-1 Card detect GPIO configuration on MSM8974 for SLOT1

Enable UIM1 hotswap	True
UIM1 hotswap polarity	ACTIVE_HIGH
UIM1 card detect GPIO num	100
UIM1 card detect GPIO function selection	0
UIM1 card detect GPIO drv strength	2 mA
UIM1 card detect pull setting	No pull

Table 2-2 Card detect GPIO configuration on MSM8974 for SLOT2

Enable UIM2 hotswap	False
UIM2 hotswap polarity	ACTIVE_HIGH
UIM2 card detect GPIO num	52
UIM2 card detect GPIO function selection	0
UIM2 card detect GPIO drv strength	2 mA
UIM2 card detect pull setting	No pull

Description (cont.)

- The UIM controller hardware debounces the card detect line. The debounce time value is set to three sleep clocks (sleep clock frequency = 32.768 kHz). This hardware debounce time value is not configurable.
- The software also debounces the card detect line to prevent the software from entering a bad state due to multiple interrupts coming from a bad switch. The debounce values are controlled by NV 66050.
- When the hotswap ISR is triggered, the debounce process starts and the UIM repeatedly sends a card status query to that slot on a fixed time interval of approximately 0.1 ms. The client can configure that number of samples through NV 66050. The default value is 20 samples to detect card insertion, one sample to detect card removal, and 10 samples for a maximum debounce retry counter for an unstable scenario.
- Debounce ends successfully when the UIM receives the same status continuously for all sample counts for that hotswap interrupt. Any mismatch during debouncing again restarts the sampling count from 1. This process ends upon a successful result or when the UIM reaches its maximum debounce retry count.
- The time taken for card detection depends on the number of samples and fixed time intervals, which is approximately 0.1 ms for most targets.
- The sampling logic is based on the number of allowed samples for card detection and the number of retrying counts for this debounce. There is a different number for the allowed sample count for card inserted and card removed detection, but the maximum debounce retry is the same for both queries.

Log Analysis

Successful card insertion detection

Debounce for card insertion interrupt

25:00:00:46.246 17925 uimdrv.c HOTSWAP: uim_hotswap_clear_hotswap_timer

25:00:00:46.247 17727 uimdrv.c HOTSWAP: Start, For CARD INSERTED Interrupt

25:00:00:46.348 18126 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x14, RT Status CARD INSERTED

25:00:00:46.450 18126 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x2, Max Count 0x14, RT Status CARD INSERTED
25:00:00:46.552 18126 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x3, Max Count 0x14, RT Status CARD INSERTED

25:00:00:46.653 18126 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x4, Max Count 0x14, RT Status CARD INSERTED
25:00:00:46.755 18126 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x5, Max Count 0x14, RT Status CARD INSERTED
25:00:00:46.856 18126 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x6, Max Count 0x14, RT Status CARD INSERTED

:

:

25:00:00:48.177 18126 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x13, Max Count 0x14, RT Status CARD INSERTED

25:00:00:48.280 18126 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x14, Max Count 0x14, RT Status CARD INSERTED

25:00:00:48.280 18146 uimdrv.c HOTSWAP: Debounce logic End Successfully for CARD INSERTED interrupt

Sending card inserted request to UIM task

25:00:00:48.280 18148 uimdrv.c HOTSWAP: Processing card inserted

25:00:00:48.280 14189 uim.c HOTSWAP: Card inserted queue count 0x0 25:00:00:48.280 14203 uim.c 19
HOTSWAP:uim_hotswap_send_card_inserted_cmd is send successfully to uim for 0x1

25:00:00:48.280 18200 uimdrv.c HOTSWAP: Process end

Log Analysis (cont.)

UIM processing of card insertion request and flush pending commands before resetting the card

25:00:00:48.285 4425 uim.c uim_flush_command for slot 0x1 and flush type 0x1

25:00:00:48.285 4455 uim.c Flushing 0x0 commands, uim status is 0x7

25:00:00:48.286 4548 uim.c before flushing Total CMD count 0x0, SLOT1 CMD count 0x0 SLOT2 CMD count 0x0

25:00:00:48.286 4550 uim.c After flushing Total CMD count 0x0 , Removed count 0x0 for slot 0x1

25:00:00:48.286 14599 uim.c HOTSWAP: executing UIM_HOTSWAP_CARD_INS_F for slot 0x1

Sending the Reset command for card insertion request

25:00:00:48.286 3643 uimgen.c Received Reset command for UIM_HOTSWAP_CARD_INS_F

UIM Powerup state

25:00:00:48.290 3602 uimdrv.c uim_hotswap_reconfig_uim for slot 0x0

25:00:00:48.290 4359 uimdrv.c uim_power_up

25:00:00:48.290 4366 uimdrv.c uim power up @ 1.8 v

25:00:00:48.315 9136 uimgen.c UIM generic state in uim_command 0

25:00:00:48.315 10573 uim.c uim_process_card_response

25:00:00:48.315 9275 uimgen.c generic_state_ptr 0x0, uim_reselect_mf 0x0

UIM Reset state

25:00:00:48.315 4674 uimdrv.c uim_reset

25:00:00:48.316 9136 uimgen.c UIM generic state in uim_command 1

25:00:00:48.547 4580 uim.c Recd Command Response Signal

25:00:00:48.547 10573 uim.c uim_process_card_response

Log Analysis (cont.)

UIM Delay After ATR state

25:00:00:48.548 17094 uimdrv.c FI and DI are supported
25:00:00:48.550 9136 uimgen.c UIM generic state in uim_command 2
25:00:00:48.550 10573 uim.c uim_process_card_response
25:00:00:48.550 9275 uimgen.c generic_state_ptr 0x2, uim_reselect_mf 0x0

UIM PPS state

25:00:00:48.550 4954 uimdrv.c uim_send_pps
25:00:00:48.550 9136 uimgen.c UIM generic state in uim_command 3
25:00:00:48.683 4580 uim.c Recd Command Response Signal
25:00:00:48.685 10573 uim.c uim_process_card_response
25:00:00:48.685 9275 uimgen.c generic_state_ptr 0x3, uim_reselect_mf 0x0

Update Operational Parameters state

25:00:00:48.685 10440 uimgen.c The UIM is operating under T=0x0
25:00:00:48.685 5043 uimdrv.c uim_update_op_params
25:00:00:48.711 9136 uimgen.c UIM generic state in uim_command 5
25:00:00:48.712 10573 uim.c uim_process_card_response
25:00:00:48.712 9275 uimgen.c generic_state_ptr 0x5, uim_reselect_mf 0x0

Check Characteristics state

25:00:00:48.712 5733 uimdrv.c uim_send_command
25:00:00:48.713 9136 uimgen.c UIM generic state in uim_command 7
25:00:00:49.092 4580 uim.c Recd Command Response Signal
25:00:00:49.092 10573 uim.c uim_process_card_response
25:00:00:49.092 9273 uimgen.c SW1 0x90,SW2 0x0, status 0x0
25:00:00:49.093 9275 uimgen.c generic_state_ptr 0x7, uim_reselect_mf 0x0

Log Analysis (cont.)

Select state

25:00:00:49.093 5733 uimdrv.c uim_send_command

25:00:00:49.095 9136 uimgen.c UIM generic state in uim_command 10

25:00:00:49.135 4580 uim.c Recd Command Response Signal

25:00:00:49.135 10573 uim.c uim_process_card_response

25:00:00:49.135 9273 uimgen.c SW1 0x90,SW2 0x0, status 0x0

25:00:00:49.135 9275 uimgen.c generic_state_ptr 0xa, uim_reselect_mf 0x0

Read ICCID state

25:00:00:49.135 5733 uimdrv.c uim_send_command

25:00:00:49.165 4580 uim.c Recd Command Response Signal

25:00:00:49.166 10573 uim.c uim_process_card_response

25:00:00:49.166 9273 uimgen.c SW1 0x90,SW2 0x0, status 0x0

25:00:00:49.166 9275 uimgen.c generic_state_ptr 0x9, uim_reselect_mf 0x0

25:00:00:49.166 9136 uimgen.c UIM generic state in uim_command 55

Sending the Link Establish command to the GSDI

25:00:00:49.166 8744 uim.c Sending the COMM LINK ESTABLISHED to GSDI

25:00:00:49.166 8796 uim.c UIM link established with card over legacy ISO interface

Log Analysis (cont.)

- **Successful card removal detection**

Card removed while the UIM command is processed

25:00:00:39.035 12016 uim.c Clock started
25:00:00:39.035 4981 uim.c Last DF1,DF2,EF 7f10 ffff 6f06
25:00:00:39.035 4985 uim.c New path ffff ffff 6f49
25:00:00:39.035 3067 uimgen.c Received UIM_CACHED_SEL_F command
25:00:00:39.035 5733 uimdrv.c uim_send_command
25:00:00:39.036 9136 uimgen.c UIM generic state in uim_command 10

Rx break error and timeout due to card removal

25:00:00:39.087 15902 uimdrv.c Maxed the Rx break error count
25:00:00:39.092 12454 uim.c HOTSWAP: UIM received response time out signal
25:00:00:39.092 12455 uim.c Uim internal hotswap detection
25:00:00:39.092 17925 uimdrv.c HOTSWAP: 16 uim_hotswap_clear_hotswap_timer
25:00:00:39.092 17732 uimdrv.c HOTSWAP: Start, For CARD REMOVED Interrupt
25:00:00:39.093 12540 uim.c Timed out on the command response
25:00:00:39.093 12618 uim.c Reset due to rx break errors
25:00:00:39.093 12746 uim.c UIM timeout in external command
25:00:00:39.165 2762 uim.c Starting to log the timeout Information

Log Analysis (cont.)

Debounce for card removal

25:00:00:39.193 18126 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x1, RT Status CARD INSERTED

25:00:00:39.193 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x1

25:00:00:39.295 18126 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x1, RT Status CARD INSERTED

25:00:00:39.295 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x2

25:00:00:39.395 18126 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x1, RT Status CARD INSERTED

25:00:00:39.395 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x3

25:00:00:39.496 18131 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x1, RT Status CARD REMOVED

25:00:00:39.496 18153 uimdrv.c HOTSWAP: Debounce logic End Successfully for CARD REMOVED interrupt

Notify card removal and send card removal request to UIM

25:00:00:39.496 18154 uimdrv.c HOTSWAP: Processing card removal

25:00:00:39.496 14130 uim.c HOTSWAP: Card removed queue count 0x0

25:00:00:39.496 14138 uim.c HOTSWAP: Reporting error condition with UIM_CARD_REMOVED_S for slot 0x1

25:00:00:39.496 4267 uim.c uim_notify_error status 0x1d for slot1

25:00:00:39.496 14149 uim.c HOTSWAP: uim_hotswap_send_card_removed_cmd is successfully sent to uim for 0x1

25:00:00:39.497 18200 uimdrv.c HOTSWAP: Process end

MMGSDI sees the card removal interrupt

59:00:00:39.505 7838 mmgsdi.c Received event: 0x1 in mmgsdi_evt_cb

59:00:00:39.516 7838 mmgsdi.c Received event: 0xd in mmgsdi_evt_cb

UIM enter into recover due to command response time out

25:00:00:40.493 3413 uim.c End logging information to UimReset.Txt

25:00:00:40.495 12850 uim.c Reset after timeout Rx-state 0x1 Tx- state 0x3

25:00:00:40.495 8268 uim.c Internal command to Reset the UIM for slot 0x1

Log Analysis (cont.)

Recovery is stopped in the middle due to card removal interrupt and also starts an internal hotswap debounce for card insertion

25:00:00:40.495 8316 uim.c uim_reset_uim is not allowed, card is not inserted yet for this slot

25:00:00:40.495 8333 uim.c Uim internal hotswap detection

25:00:00:40.495 17925 uimdrv.c HOTSWAP: uim_hotswap_clear_hotswap_timer

25:00:00:40.495 17727 uimdrv.c HOTSWAP: Start, For CARD INSERTED Interrupt

25:00:00:40.495 8373 uim.c Card is not detected on slot

UIM received card removal request and flush Pending command

25:00:00:40.495 4425 uim.c uim_flush_command for slot 0x1 and flush type 0x1

25:00:00:40.496 4455 uim.c Flushing 0x1 commands, uim status is 0x7

25:00:00:40.496 4548 uim.c before flushing Total CMD count 0x1, SLOT1 CMD count 0x0 SLOT2 CMD count 0x1

25:00:00:40.496 4550 uim.c After flushing Total CMD count 0x1 , Removed count 0x0 for slot 0x1

25:00:00:40.500 14798 uim.c HOTSWAP: executing UIM_HOTSWAP_CARD_REM_F for slot 0x1

UIM powerdown due to card removal

25:00:00:40.501 3816 uimgen.c Received power down command

25:00:00:40.501 5797 uimgen.c Turning off UIM because of POWER down cmd

25:00:00:40.501 6424 uimdrv.c uim power down @ 1.8 v

25:00:00:40.505 6453 uimdrv.c uim_power_down

25:00:00:40.521 9136 uimgen.c UIM generic state in uim_command 41

25:00:00:40.521 12264 uim.c CMD_RSP Sig Rcvd uim_status=0x7, uim_st_bf_us=0x1, cmd_ptr=0x44ce0860

25:00:00:40.521 12115 uim.c SIM in power down state

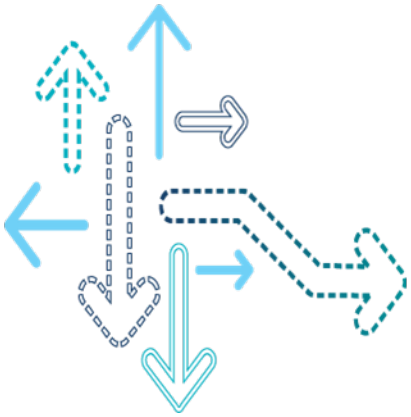
Log Analysis (cont.)

Unstable result for card insertion request

25:00:00:40.597 18131 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x14, RT Status CARD REMOVED
25:00:00:40.597 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x1
25:00:00:40.698 18131 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x14, RT Status CARD REMOVED
25:00:00:40.698 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x2
25:00:00:40.800 18131 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x14, RT Status CARD REMOVED
25:00:00:40.800 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x3
25:00:00:40.902 18131 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x14, RT Status CARD REMOVED
25:00:00:40.902 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x4
25:00:00:41.003 18131 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x14, RT Status CARD REMOVED
25:00:00:41.003 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x5
25:00:00:41.105 18131 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x14, RT Status CARD REMOVED
25:00:00:41.105 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x6
25:00:00:41.207 18131 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x14, RT Status CARD REMOVED
25:00:00:41.207 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x7
25:00:00:41.308 18131 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x14, RT Status CARD REMOVED
25:00:00:41.310 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x8

25:00:00:41.410 18131 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x14, RT Status CARD REMOVED
25:00:00:41.410 18193 uimdrv.c HOTSWAP: Debounce logic Maximum Retry 0xa Current Count 0x9
25:00:00:41.512 18131 uimdrv.c HOTSWAP: Debounce logic Sample Count 0x1, Max Count 0x14, RT Status CARD REMOVED
25:00:00:41.512 18179 uimdrv.c HOTSWAP: Debounce logic End, Unstable status for CARD INSERTED interrupt
25:00:00:41.512 18200 uimdrv.c HOTSWAP: Process end

NV Settings



NV Settings

- 855 – RTRE Configuration
  - 0 – RUIM Only
  - 1 – NV only
  - 2 – Fallback to NV
- 896 – Instruction class setting
  - 1 – 3G inst class
  - 0 – 2G inst class
- 6907 – DSDS hardware config, applicable DSDS targets only, MSM7xxx and MSM8x25 DSDS build
  - 0 – Single SIM hardware
  - 1 – Dual SIM hardware
- 67353 – Hotswap support present or not
  - Set to 1 – Hotswap support present

NV Settings (cont.)

- 67232 – Logic level on the card detection line
  - This NV item is a 2-byte value.
    - Byte 0 – UIM1 card detection polarity
    - Byte 1 – UIM2 card detection polarity
  - The possible values for each byte are:
    - 0 – ACTIVE LOW
    - 1 – ACTIVE HIGH
- 66050 – ME hotswap config
  - Sets the value for the number of samples for insertion/removal and debounce-related parameters. The UIM controller hardware debounces the card detect line. The debounce time value is set to three sleep clocks (sleep clock frequency = 32.768 kHz). This hardware debounce time value is not configurable.
  - The software also debounces the card detect line to prevent the software from entering a bad state due to multiple interrupts coming from a bad switch
    - The following are the default values:
      - num_of_sample_fpr_insertion – 20
      - num_of_sample_fpr_removal – 1
      - maximum_debounce_retry_counter – 10
      - auxiliary_period_for_card_detect – 0

NV Settings (cont.)

- 4205 – UIM config params
  - This NV can be used to add delay at UIM powerup. This is useful in capturing powerup logs which often get missed if this NV is not set properly.
  - Set the below parameter under NV 4205 to a value < 8 sec to be able to capture powerup logs.
    - uim-config parameter[57]
- 67330 – UIM features status list
  - The indexes under this NV are listed below. Some of these indexes can be used and recommended based on the issue and based on the need.
  - handle_no_atr_in_40000_clk_cycles -> Index 1
  - log_to_efs -> index 2
  - disable_recovery_upon_infinite_null -> index 3
  - debug_log -> index 4
    - This boolean is used to disable APDU logging
    - 1 – TRUE, enable APDU logging (along with this 0x1098 and 0x14CE log packets need to be enabled in QXDM to capture the logging in QXDM)
    - 0 – FALSE (default), disable APDU logging

NV Settings (cont.)

- 67330 – UIM features status list (cont.)
  - rpt_bad_sw_on_poll -> index 5
    - This NV is disabled by default. When enabled, it allows the UIM to generate a poll error on reception of bad status word on poll request.
  - handle_iccid_read_failure -> index 6
    - This NV is disabled by default. It is used to continue card powerup even if the READ on ICCID fails.
  - support_no_iccid -> index 7
    - This NV is enabled by default. It helps the software continue with card powerup even if the SELECT operation for the ICCID file fails. Reasons for this could be ICCID file is absent or missing or a buggy card not letting us SELECT ICCID.
  - min_tpl_iccid_support -> index 8
    - This NV is disabled by default. When enabled, it allows the UIM to reattempt the TP with minimum length (0x09) on TP failure. It also stores the ICCID and TP length so that at next powerup the UIM will send the minimum TP to that card to avoid the TP failure.

NV Settings (cont.)

- 67330 – UIM features status list (cont.)
  - handle_unknown_proc_bytes_as_cmd_timeout -> Index 9
    - This NV is disabled by default. If enabled, it allows the UIM to trigger the recovery on reception of bad status word from the card.
  - interface_not_used -> index 10
    - This NV is disabled by default. If enabled, the UIM interface is disabled and there is no configuration of UART, GPIO, PMIC, NPA node, and no SIM/card communication.
  - log_apdu_to_efs -> index 11
    - This NV is enabled by default to capture the APDU logging in F3 messages. If disabled, the APDU message will not capture in the F3 log.
  - no_switch_inst_on_wwt_expiry -> index 12
    - This NV is disabled by default. If enabled, it allows the UIM to send the status command before processing the Authentication request.
  - send_status_before_auth -> index 13
    - This NV is disabled by default. If enabled, it allows the UIM to send the status command before processing the Authentication request.

NV Settings (cont.)

- 67330 – UIM features status list (cont.)
  - try_default_baud_rate_for_f372_d12_card -> index 14
  - cold_reset_due_to_card_switch -> index 15
  - Sm_prefer_slot1 -> index 16
    - When the subscription manager is present (for using two cards but only publishing one card's presence to the rest of the modem), this NV defines the preferred slot to be used.
      - 1 - "TRUE", Slot 1 is the preferred slot.
      - 0 - "FALSE",-Slot 2 is the preferred slot
    - If the preferred slot has a card, that card's information is published. If the card in the preferred slot is absent, the default (soldiered always present) SIM's information is published.
  - uim_use_dual_ldo -> index 17
    - 0 – Not active, use only one LDO; code follows normal execution path
    - 1 – Use two LDOs; code powers on/off the second LDO whenever the first is changed

NV Settings (cont.)

- 67330 – UIM features status list (cont.)
  - uim_polling_only_at_polling_timer_expiry -> index 18
    - This boolean NV decides whether or not to poll the card only at the expiry of the polling timer.
      - 1 – TRUE, the card is polled only at the expiry of the polling timer; if within a traffic channel and a polling request is received, poll the card only at the expiry of the polling timer.
      - 0 – FALSE (default), card is polled even if the polling timer has not expired.
  - uim_set_clk_freq_at_4_8_MHz -> Index 19
    - This boolean NV is used to decide whether or not to enable the feature to use 4.8 MHz UIM Reference CLK.
      - 1 – TRUE (default), UIMDRV would attempt to power up UIM with 4.8 MHz default clk; based on negotiations with the card, it would continue at 4.8 MHz or switch to 3.84 MHz.
      - 0 – FALSE, UIMDRV would attempt to power up UIM with a 3.84 MHz default clk.

NV Settings (cont.)

- 67330 – UIM features status list (cont.)
  - uim_handle_tc1_byte_for_extra_guard_time -> index 20
    - This boolean NV allows UIM to use the extra guard time given by the card in TC1 byte. TC1 byte of the ATR provides the value of N which defines the extra guard time to be used between successive characters.
      - 1 – TRUE – Use TC1 byte for guard time calculation
      - 0 – FALSE – Do not use the TC1 byte for guard time calculation
  - uim_enable_sim_mode_change_via_warm_reset -> Index 21
    - This boolean NV allows UIM to trigger warm reset to move card from specific mode (TA2 byte Present in ATR) to negotiable mode (TA2 byte absent), where PPS selection is allowed.
      - 1 – TRUE - Allow UIM to do warm reset to change the SIM mode
      - 0 – FALSE - Do not allow UIM to do warm reset to change the SIM mode

NV Settings (cont.)

- 67330 – UIM features status list (cont.)
  - uim_explicit_mf_adf_Selection -> Index 22
    - This boolean NV is used for buggy cards which require an MF selection on channel 0 in the following cases:
      - After a successful location status envelope
      - After a successful terminal response command
      - When an AID selection fails on a nonzero channel
        - 1 – TRUE, MF selection on channel 0
        - 0 – FALSE (default), no MF selection on channel 0
  - uim_boot_up_inverse_convention -> Index 23
    - This boolean NV is used whenever the UIM driver needs to boot up in inverse convention.
      - 1 – TRUE, boot up in inverse convention.
      - 0 – FALSE (default), boot up in direct convention
  - uim_enable_recovery_on_bad_status_words -> Index 24
    - This boolean NV is used whenever the UIM driver needs to do a recovery upon receiving bad status words.
      - 1 – TRUE, enable recovery upon bad status words
      - 0 – FALSE (default), disable recovery upon bad status words

NV Settings (cont.)

- 67330 – UIM features status list (cont.)
  - uim_attempt_pup_3v_from_nth_recovery -> Index 25
    - This uint8 NV is used to set if the UIM driver needs to attempt powerup with 3 V from the nth recovery
      - 0 (default) – Disable this feature
      - 1 – Attempt powerup with 3 V (if card supports) from the first recovery
      - n – Attempt powerup with 3 V (if card supports) from the nth recovery
    - Version of the NV item (stored in the first field: version) is changed from 7 to 8 with this update.
  - UIMDRV_FEATURE_LE_SUPPORT_FOR_7816_STREAM_APDU -> Index 26
    - Version of the NV item (stored in the first field: version) is changed from 8 to 9 with this update.
  - disable_card_status_check_at_power_up -> Index 29 (can be seen in version 10
    qxdm >= 3.14.917)
    - This uint8 NV is used to disable/enable card status check at power up if Hotswap is enabled.
      - 0 (default) – Enable card status check at powerup
      - 1 – Disable card status check at powerup

NV Settings (cont.)

- 70210 – UIM HW Config
  - These runtime-configurable items are part of the UIM HW CONFIG EFS item (70210). You can enable Slot 1/Slot 2 or disable either Slot 1 or Slot 2 using this NV item. You can also enable Hotswap for Slot 1/Slot 2 using this NV item.
  - The following image provides details on how to enable Slot 1.

Fields

EFS item: 70210 (UIM HW Config)

Input	Value	Name (Partial)	Size	Type
0	VER_0	version	8	Signed Enum
0	0	hw_config.UIM_BATT_ALARM_GPIO_NUM	16	UINT16
0	0	hw_config.UIM_BATT_ALARM_GPIO_FUNC_SELECTION	8	UINT8
0	2MA	hw_config.UIM_BATT_ALARM_GPIO_DRV_STRENGTH	8	Signed Enum
0	NO_PULL	hw_config.UIM_BATT_ALARM_PULL_SETTING	8	Signed Enum
0	FALSE	hw_config.UIM[0].DISABLE_UIM	8	Signed Enum
0	FALSE	hw_config.UIM[0].ENABLE_UIM_HOTSWAP	8	Signed Enum
1	ACTIVE_HIGH	hw_config.UIM[0].UIM_HOTSWAP_POLARITY	8	Signed Enum
89	89	hw_config.UIM[0].UIM_CONTROLLER_IRQ_NUM	16	UINT16
0	BADGER_MSS_UIM...	hw_config.UIM[0].UIM_UART	8	Signed Enum
87	87	hw_config.UIM[0].UIM_UART_IRQ_NUM	16	UINT16
0	PMIC_NPA_RESOU...	hw_config.UIM[0].UIM_VCC	8	Signed Enum

UIM1 is enabled

Hotswap disabled by default. Set to TRUE to enable it

NV Settings (cont.)

- 70210 – UIM HW Config (cont.)
  - This image provides details on how to disable Slot 2. You can set it to False in order to enable Slot 2.

1	1	hw_config.UIM[0].UIM_CARD_DETECT_GPIO_FUNC_SEL...	8	UINT8	UIM2 is disabled
0	2MA	hw_config.UIM[0].UIM_CARD_DETECT_GPIO_DRV_STRE...	8	Signed Enum	
0	NO_PULL	hw_config.UIM[0].UIM_CARD_DETECT_PULL_SETTING	8	Signed Enum	
1	TRUE	hw_config.UIM[1].DISABLE_UIM	8	Signed Enum	
0	FALSE	hw_config.UIM[1].ENABLE_UIM_HOTSWAP	8	Signed Enum	

NV Settings (cont.)

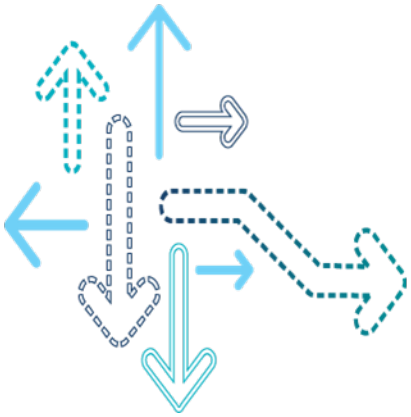
- NV 73535 – Extended Recovery
  - This is used to configure the number of extended_powerups, delay_between_powerup,s and whether to use incremental_delay
  - This NV is applicable for PL's : BOLT 2.5.1 onwards

  Note: extended_powerups means the fresh power-up attempts after all the recovery attempts are exhausted.

- NV 73600 – UIM GPIO based on Hotswap
  - Used for configuring which GPIO for each detect line
  - This NV is applicable for MDM 9x45 and MSM8952/56 (PL's 1.0 onwards)
  - Each UIM detect can be configured as controller-based GPIO or common GPIO

```
typedef enum {
UIM_CNTRL_BASED_HOTSWAP = 0
HOTSWAP_UIM1_PRESENT_GPIO
HOTSWAP_UIM2_PRESENT_GPIO
HOTSWAP_UIM3_PRESENT_GPIO
HOTSWAP_UIM4_PRESENT_GPIO
}
The default configuration would be as shown below -
UIM1 UIM_CNTRL_BASED_HOTSWAP
UIM2 UIM_CNTRL_BASED_HOTSWAP
UIM3 UIM_CNTRL_BASED_HOTSWAP
```

Code Customizations



Code Customizations

- Increase voltage
  - Enable FEATURE_UIM_DRIVE_MAX_PMIC_VOLTAGE
- Increase guard time

```
File: uimdrv.c
Function: uim_update_op_params
change
UIM_UART_CNFG( UART_SIM_CFG__STOP_BIT_LEN_MASK,
op_params->guardtime_bits<<UART_SIM_CFG__STOP_BIT_LEN_SHFT);
to
UIM_UART_CNFG( UART_SIM_CFG__STOP_BIT_LEN_MASK,
op_params->guardtime_bits+10<<UART_SIM_CFG__STOP_BIT_LEN_SHFT);
```

Warning: These code customizations are only for reference. Any of these code customizations should not be integrated by the customer without consulting Qualcomm Technologies Inc. (QTI).

Code Customizations (cont.)

- Increase WWT

File: uimgen.c

Function: uim_generic_command

Change From:

```
/* Send the operational parameters */
uim_update_op_params( uim_op_params_ptr, uim_drv_slot );

/* Compute a new work waiting time */
uim_work_waiting_time[uim_drv_slot] = ((960 * uim_WI[uim_drv_slot]) *
crcf_values[uim_op_params_ptr->FI]) /
(uim_clk_freq[uim_drv_slot] / 1000) + UIM_CLK_MS_PER_TICK;
```

To:

```
/* Send the operational parameters */
uim_update_op_params( uim_op_params_ptr, uim_drv_slot );

/* Compute a new work waiting time */
uim_work_waiting_time[uim_drv_slot] = ((960 * uim_WI[uim_drv_slot]*10) *
crcf_values[uim_op_params_ptr->FI]) /
(uim_clk_freq[uim_drv_slot] / 1000) + UIM_CLK_MS_PER_TICK;
```

Code Customizations (cont.)

- Increase driver strength
  - Example – MSM7x25 chipset
  - Location – AMSS\products\76XX\dal\drivers\tlmm\inc\TLMMGpio7625.h

```
UIM1_CLK = TLMM_GPIO_CFG(47, 2, GPIO_OUTPUT, GPIO_PULL_DOWN, GPIO_2MA), --------> change
   to GPIO_4MA


UIM1_DATA_OUT = TLMM_GPIO_CFG(50, 2, GPIO_OUTPUT, GPIO_PULL_DOWN, GPIO_2MA), -------->
   change to GPIO_4MA
```


Warning: Do not try changing the driver strength at all unless consulted with QTI.

References

Title	Number
Standards	
ISO 7816 Part 3: Electronic Signals and Transmission Protocols	ISO 7816 Part 3
Universal Mobile Telecommunications System (UMTS); USIM Conformance Test Specification	3GPP TS 31.122
Resources	
Presentation: FR1150 – EFS Item Configuration	80-NE596-1
UIM Driver Configurable Items	80-NE596-2

Acronyms	
Term	**Definition**
APDU	Application Protocol Data Unit
DSDS	Dual Sim Dual Standby
PDU	Protocol Data Unit
TPDU	Transfer Protocol Data Unit
UIM	User Identity Module

Questions?

https://createpoint.qti.qualcomm.com