AR7 DSL Device Driver User Guide

Revision: 0.1

Revision Date: 16th June, 2006

Texas Instruments Inc.

Dallas, TX

Foreword

The AR7 Linux DSL Device Driver User Guide provides documention for building and interfaces of TI's LinuxDSL driver.

Revision History

Rev.	Date	Description	Author
0.1	0/1/2006	Original document	Arvind Vasudev

Contents

1 Introduction

This document describes implementation details of AR7 Linux DSL driver. Details of CPSAR HAL and DSL HAL are excluded from this documentation. For references on them, please refer to “Ax7 DSL HAL Programmer's Guide” and “Communication Processor Hardware Abstraction Layer (CPHAL) API Reference Manual”.

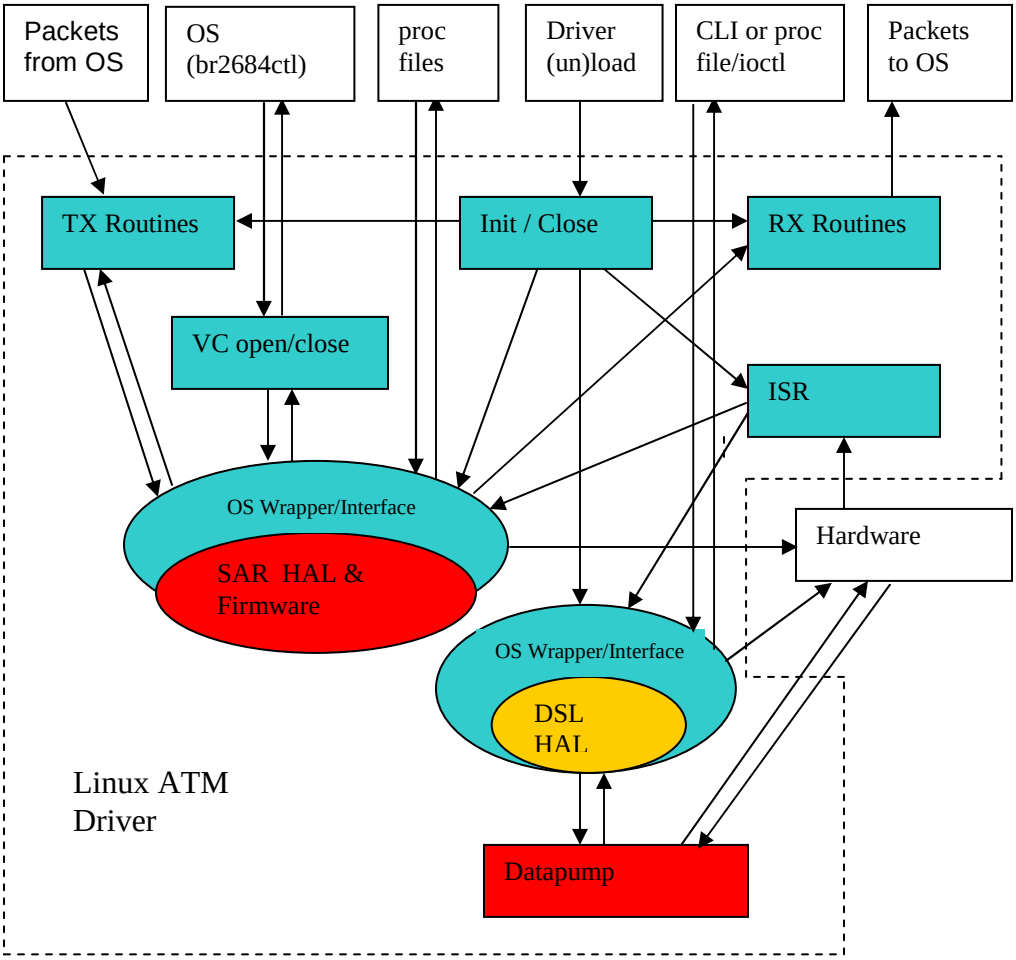
2 References

[1] Texas Instruments Inc., Ax7 DSL HAL Programmer's Guide.

[2] Texas Instruments Inc., Communication Processor Hardware Abstraction Layer (CPHAL) API Reference Manual.

3 AR7 Driver Architecture

AR7 Linux ATM driver is built as dynamically loadable modules. Figure 1 below shows the functional interfaces and software delineation of AR7 Linux ATM driver.



OS: Linux operating system.
CLI: Command line interface or any user space applications.
Proc file: Application and kernel (driver) interface for Linux.
Ioctl: I/O interface between application and the driver.

Figure 1. DSL driver architecture and module interactions.

4 ATM Device Driver Files

The TNETD73xx chipset provides hardware supports for both AAL5 packet segmentation and reassemble and ADSL physical layer. The Linux driver provides APIs to all interfaces as well to IP stack. The files for AR7 DSL device driver are described in following table.

Table 1. ATM device driver files

Files	Description
Makefile	Primary Makefile for the DSL driver.
dda_init_close.c	Primary interface for opening and closing the Linux ATM interface.
dda_txrx.c	Linux Tx and Rx functions are in this file.
dda_common.c	Contains all the miscellaneous features supported by the driver.
ddc_input.c	Contains the environment variable definitions and their accessor functions.
ddc_init_close.c	Primary interface for opening and closing the low level core of the device driver. This is OS independent
ddc_txrx.c	The OS independent Tx and Rx functions are in this file.
ddc_common.c	Contains all the miscellaneous features supported by the device including the functions for the accessing of the statistics.
ddc_clear_eoc.c	This file contains the OS independent Clear EOC functions.
dsl_hal_api.c	OS independent functions of DSL functions
dsl_hal_api.h	include file for OS independent functions of DSL functions
dsl_hal_support.c	Support functions of OS independent functions of DSL functions
dsl_hal_support.c	Include file for dsl_hal_support.c.
dsl_hal_register.h	DSL related register defines.
dev_host_interface.h, dev_host_verdef.h, env_def_defines.h, env_def_typedefs.h	DSL related include files.
tnetd7300_sar_firm.h	PDSP firmware for Tnetd73xx.
ar0700mp.bin	DSL firmware for Annex A.
ar0700db.bin	DSL firmware for Annex B.
ar0700dc.bin	DSL firmware for Annex C.
aal5sar.c, cpcommon_cpaal5.c, cpcommon_cpsar.c, cppi_cpaal5.c, cpremap_cpaal5.c, cpremap_cpsar.c, cpsar.c, aal5sar.h, cp_sar_reg.h, cpcommon_cpaal5.h, cpcommon_cpsar.h, cpsar.h, cpsar_cpaal5.h, cpswhal_cpaal5.h,	SAR HAL related files.

cpswhal_cpsar.h, ec_errors_cpaal5.h, ec_errors_cpsar.h	

4 Driver Functionality Description

4.1 Driver initialization

The AR7 ATM driver is initialized through the **tn7atm_init** function. The **tn7atm_init** function allocates memory for the private data structure used by ATM drive and calls the initialization routines of SAR HAL and DSL HAL to initialize both SAR and DSL subsystems . Upon completion of **tn7atm_init** function the driver is ready to accept socket calls from ATM applications at user level. Interrupt registrations are also handled in **tn7atm_init .** The **tn7atm_open** function is called when a user applications such as PPPoE or BR2684ctl makes a socket call to the driver to open a new connection to the SAR interface.

The **tn7atm_open** function initializes the VCC structure with the appropriate ATM VPI/VCI and QoS options. The ATM device operation structure is shown in Figure 12.

```
static const struct atmdev_ops tn7atm_ops = {
        open:           tn7atm_open,
        close:          tn7atm_close,
        ioctl:          tn7atm_ioctl,
        send:           tn7atm_send,
        change_qos:     tn7atm_change_qos,
};
```

4.2 Packet transmission

Once a proper VPI/VCI pair is initialized through user application, data can be transmitted through the **tn7atm_send** function. The **tn7atm_send** function checks the status of DSL interface and AR7 modem to ensure that a valid connection is available to send data on. The type of data is also checked and a determination is made to whether to queue the packet to high priority queue or low priority queue for further data transmission. The driver checks whether there is packets to send in every **tn7_send** call. If there is any packet to be sent, the driver will locate channel information from vcc structure and call SAR HAL send routine to send out packet. The packets are de-queued from priority queue first. Upon completion, the SAR interrupt is handled through the **tn7atm_sar_irq** function. The **tn7atm_sar_irq** calls the interrupt handle routine of SAR HAL which in turn calls the **tn7atm_send_complete** to free the packets.

4.3 Packet Reception

The ATM driver processes the reception of incoming packets (i.e., from DSL interface) through the registered SAR interrupt (**tn7atm_sar_irq)**. **tn7atm_sar_irq** calls SAR HAL interrupt handling routine which in turn calls the **tn7atm_receive** to pass on received packets. If no errors exist within the PDU, skb is passed to the IP stack

through the **atm_charge** and **vcc->push** routines. The SAR HAL will call **tn7atm_allocate_rx_skb** to replenish the used skb.

4.4 Command interface

The ATM driver provides a single command interface that an user application can call to pass command or information to the driver. To access this interface, user application needs to write a string to the proc file **/proc/sys/dev/dslmod** with following,

echo command > /proc/sys/dev/dslmod

The **dslmod_sysctl** in tn7dsl.c will parse the string and take appropriate action. The following commands are currently supported. They must be in the *string* format. Table 2 shows their name and descriptions.

Table 2. ioctl commands and description

Command	Description
T1413	Set DSL training mode to T1.413.
GDMT	Set DSL training mode to G.dmt.
GLITE	Set DSL training mode to G.Lite.
MMOD	Set DSL training mode to multi mode.
NMOD	Set no training mode.
AD2MOD	Set DSL training mode to ADSL2
AD2DEL	Set DSL training mode to ADSL2 DELT
A2PMOD	Set DSL training mode to ADSL2+ mode
A2PDEL	Set DSL training mode to ADSL2+ DELT mode
tmode nn	Set DSL training mode based on bit cmd, where nn is the two-digit hex number, which specifies the training mode.
exxxpyyyyc	Send end to end F5 cells with VPI=xxx and VCI=yyyy. xxx and yyyy are in decimal.
sxxxpyyyyc	Send seg to seg F5 cells with VPI=xxx and VCI=yyyy. xxx and yyyy are in decimal.
exxxp0c	Send end to end F4 cells with VPI=xxx.
sxxxp0c	Send segment F4 cells with VPI=xxx.
exxxpyyyycdzzzt	Send end to end F5 cells with VPI=xxx and VCI=yyyy. xxx and yyyy are in decimal. zzz is time out value in millisecond
sxxxpyyyycdzzzt	Send seg to seg F5 cells with VPI=xxx and VCI=yyyy. xxx and yyyy are in decimal. zzz is time out value in millisecond.
trellison	Set trellis on
trellisoff	Set trellis off
bitswapon	Set bitswap ON
bitswapoff	Set bitswap OFF

4.5 Proc files

AR7 ATM driver provides several proc files for user application to retrieve information from the driver. They are listed in following table.

Table 3 Proc file list.

File name and path	Description
/proc/avalanche/avsar_modem_stat	Statistics for SAR and DSL.
/proc/avalanche/avsar_modem_training	First line: modem training state.
/proc/avalanche/avsar_ver	Version information for the Driver, PDSP, DSP, and etc.
/proc/avalanche/avsar_oam_ping	Oam ping result.
/proc/avalanche/avsar_pvc_table	Received PVC table information.
/proc/avalanche/avsar_sarhal_stats	Unformatted SAR HAL statistics. (debug only)
/proc/avalanche/avsar_channels	Channel information. (debug only)

4.5.1 Important Defines

The table below lists the important defines and their corresponding values used to interpret the meaning of the data in the modem stat file.

Table 4 Trellis and Bitswap Status

Name	Value
DISABLE	0
ENABLE	1

Table 5 Modem Training Mode

Name	Value	Comments
NO_MODE	0	
MULTI_MODE	1	Default modem modulation mode
T1413_MODE	2	
GDMT_MODE	3	
GLITE_MODE	4	
ADSL2_MODE	8	
ADSL2_DELT	9	
ADSL2_PLUS	16	
ADSL2_PLUS_DELT	17	

READSL2_MODE	32	
READSL2_DELT	33	

4.6 Environment variables

The following environment variables are supported for various DSL operational needs.

Name	Value type	Meaning
modulation mmm	String	Same as command, where mmm is one of the first 9 string commands from Table 2.
modulation nnn	String	Same as command, where nnn is a decimal number specifying the corresponding command code (0 – 255) of one of the first 9 string commands in Table 2.
trellis	Integer	0, trellis off; 1 trellis on.
eoc_vendor_id	Hex	8 byte in ascii
eoc_vendor_revision	Integer	Revision In decimal.
eoc_vendor_serialnum	String	Null terminated string.
fine_gain_control	integer	0, control off; 1 control on.
fine_gain_value	integer	Gain value in hex.
dsp_freq	integer	250 -> Run DSP at 250 Mhz Any other value -> Run DSP at 200 Mhz

4.7 Build Environment

The following vobs need to be mounted in order to build in clearcase:

- ADSL
- ADSLdrivers
- bcg_cpsw
- dsps_dsldk
- psp
- psp_linux
- psp_boot
- psp_imports

Then on the linux build server go to the dsps_dsldk vob and type “make”. This will make the squashfs images for the kernel and the filesystem in the build_op directory.