PM73122 / PM73123 / PM73124

AAL1GATOR-32/-8/-4

DEVICE DRIVER ERRATA

PRELIMINARY

ISSUE 2: JUNE 2001

PUBLIC REVISION HISTORY

Issue No.	Issue Date	Details of Change
1	December 2000	Document created.
2	June 2001	Modifications made to the errata: al3addq – Erroneous Parameters. Errata: “al3_dpr.c and al3_dpr.h – New Files Required” changed to indicate that these files are highly application dependent and it is recommended that these files be written by the customer. The examples files included with the beta-1.0 software driver release contain errors and can only be used as a guide until the next release of the software driver. New errata: “al3ActivateChannel Returns a Queue Handle Usable Only in the Receive Direction”

CONTENTS

1 ISSUE 1 DEVICE DRIVER ERRATA

This document is the errata notice for the beta-1.0 release of the AAL1GATOR-32/-8/-4 (PM73122/PM73123/PM73124) device driver. Note that the beta-1.0 release of the AAL1GATOR-32/-8/-4 device driver supersedes all prior versions.

1.1 Driver Identification

The information in this document applies to the beta-1.0 release of the PM73122/PM73123/PM73124 AAL1GATOR-32/-8/-4 device driver.

The complete beta-1.0 release of the AAL1GATOR-32/-8/-4 driver includes the following files:

Table 1. AAL1GATOR-32/-8/-4 Device Driver Files.

Directory	Filename
driver/src	al3_api.c
	al3_api2.c
	al3_chan.c
	al3_diag.c
	al3_dpr.c
	al3_hw.c
	al3_init.c
	al3_isr.c
	al3_rtos.c
	al3_stat.c
driver/inc	al3_api.h
	al3_chan.h
	al3_dev.h
	al3_diag.h
	al3_dpr.h
	al3_isr.h
	al3_stat.h
	al3_mdb.h
	al3_rtos.h
	al3_init.h
	al3_hw.h

Directory	Filename
	types.h
examples/src	app.c[1]
	profiles.c[2]
examples/inc	app.h[1]
	hw_pci.h[1]
	rtos_vxw.h[1]
makefile	Makefile

Notes:

1. The app.c, app.h, rtos_vxw.h, and hw_pci.h files contain example code.

2. profiles.c contains sample initialization profiles for T1 UDF, T1 SDF-FR, T1 SDF-MF, E1 UDF, E1 SDF-FR, E1 SDF-MF and DS3.

2 AAL1GATOR-32/-8/-4 DEVICE DRIVER FUNCTIONAL DEFICIENCIES OVERVIEW

This section outlines the known functional deficiencies of the beta-1.0 release of the AAL1GATOR-32/-8/-4 (PM73122/PM73123/PM73124) device driver.

The errata in Table 2 are explained in more detail in section 3 of this document.

Table 2 - FUNCTIONAL DEFICIENCIES SUMMARY LIST

#	Discrepancy	Workaround
3.1	Register Access Limitation Above 0x82302	Yes – code edit
3.2	Example Code profiles.c – Syntax Error and Correction	Yes – code edit
3.3	SBI INSBI INS_FIFO_Underrun_INT: Incorrect Interrupt Bit Polled	Yes – code edit
3.4	SBI INSBI INS_FIFO_Overrun_INT: Incorrect Interrupt Bit Polled	Yes – code edit
3.5	EXSBI: Incorrect Code for Limiting Access to the Currently Active Page	Yes – code edit
3.6	Replace “UINT” with “UINT2”	Yes – code edit
3.7	al3_dpr.c and al3_dpr.h Contain Errors	No – fixed in next release of software driver
3.8	al3addq – Erroneous Parameters	Yes – code edit
3.9	al3setlinestrmode – Missing Register Write	Yes – code edit
3.10	Example Profile #3 – T_CHAN_UNSTRUCT Erroneously Set	Yes – code edit
3.11	al3ActivateChannel Returns a Queue Handle Usable Only in the Receive Direction	Yes – code edit

3 AAL1GATOR-32/-8/-4 DEVICE DRIVER FUNCTIONAL DEFICIENCY DETAILS

Note that all deficiencies listed below will be fixed in the production release of the AAL1GATOR-32/-8/-4 (PM73122/PM73123/PM73124) device driver.

3.1 Register Access Limitation Above 0x82302

Description

Register access has been erroneously limited to below address 0x82302. This affects commands such as al3Read, al3Write, al3ReadBlock, and al3WriteBlock.

Workaround

Change constant AL3_REG_MAX_OFFSET from 0x82302 to 0x84003 in file al3_dev.h.

Performance With Workaround

Registers above 0x82302 are accessible.

Performance Without Workaround

Access above address 0x82302 is not possible. Issuing the al3Read or al3ReadBlock command with an address range above 0x82302 will fail and return the value 0x0000.

Write operations above address 0x82302 using the al3Write command are also not possible without the workaround.

Note: This errata has no known negative effects on the operation of the device.

3.2 Example Code profiles.c – Syntax Error and Correction

Description

There are erroneous instances of semicolons that result in syntax errors when profiles.c is compiled.

Workaround

Remove the instances of erroneous semicolons which all appear after ‘srtsCDVT’ or ‘srtsEnable’ in profiles.c.

Performance With Workaround

profiles.c compiles and works as expected.

Performance Without Workaround

Syntactical errors result when profiles.c is compiled.

3.3 SBI INSBI INS_FIFO_Underrun_INT: Incorrect Interrupt Bit Polled

Description

In file al3_api2.c, the wrong bit (an unused bit) is polled in the INS_MSTR_INT register.

Workaround

Bit 3, the INS_FIFO_UDR_INT bit, should be polled instead of the current implementation which polls an unused bit. Update the function initINSBI() in al3_api2.c as below:

Incorrect Code

```
while (sysAl3ReadReg(pDDB->baseAddr, AL3_INS_MSTR_INT) & 0x10)

        (void)sysAl3ReadReg(pDDB->baseAddr, AL3_INS_FI_URI);
```

Correct Code

```
while (sysAl3ReadReg(pDDB->baseAddr, AL3_INS_MSTR_INT) &
AL3_INS_MSTR_INT_FIFO_UDR)

        (void) sysAl3ReadReg(pDDB->baseAddr, AL3_INS_FI_URI);
```

Performance With Workaround

Functions normally.

Performance Without Workaround

Incorrect interrupt detection.

3.4 SBI INSBI INS_FIFO_Overrun_INT: Incorrect Interrupt Bit Polled

Description

In file al3_api2.c, the wrong bit is polled in the INS_MSTR_INT register.

Workaround

Bit 2, the INS_FIFO_OVR_INT bit, should be polled instead of the current implementation which polls the INS_FIFO_UDR_INT bit. Update the function initINSBI() in al3_api2.c as below:

Incorrect Code

```
while (sysAl3ReadReg(pDDB->baseAddr, AL3_INS_MSTR_INT) & 0x08)

        (void) sysAl3ReadReg(pDDB->baseAddr, AL3_INS_FI_ORI);
```

Correct Code

```
while (sysAl3ReadReg(pDDB->baseAddr, AL3_INS_MSTR_INT) &
AL3_INS_MSTR_INT_FIFO_OVR)

        (void) sysAl3ReadReg(pDDB->baseAddr, AL3_INS_FI_ORI);
```

Performance With Workaround

Functions normally.

Performance Without Workaround

Incorrect interrupt detection.

3.5 EXSBI: Incorrect Code for Limiting Access to the Currently Active Page

Description

There is incorrect code in the functions setEXSBI, getEXSBI, setINSBI and getINSBI in file al3_api2.c.

Workaround

For the functions setEXSBI, getEXSBI, setINSBI and getINSBI in file al3_api2.c, remove the line below:

```
if (pageNum != ((sysAl3ReadReg(base, AL3_EXSBI_CFG) & 0x0080) >> 7))

        return aalErrUTIL(AL3_ERR_CFG);
```

Performance With Workaround

Functions normally.

Performance Without Workaround

The code above checks the APAGE bit. This is unnecessary code as it is simpler and safer to write the active page.

3.6 Replace "UINT" with "UINT2"

Description

In file al3_init.c instances of "UINT" need to be replaced with "UINT2".

Workaround

In file al3_init.c instances of "UINT" need to be replaced with "UINT2".

Performance With Workaround

al3_init.c compiles properly.

Performance Without Workaround

al3_init.c may not compile properly.

3.7 al3_dpr.c, al3_dpr.h, al3_isr.c and al3_isr.h Files Contain Errors

Description

Because the files al3_dpr.c, al3_dpr.h, al3_isr.c and al3_isr.h are highly application dependent, it is recommended that these files be written by the

customer. Note that the example files al3_dpr.c, al3_dpr.h, al3_isr.c and al3_isr.h in the beta-1.0 release of the AAL1GATOR-32/-8/-4 (PM73122/PM73123/PM73124) device driver contain inefficiencies and errors. It is possible that the system could hang due to CPU starvation of system threads during execution of the ISR. New revisions of these files will be available in the next release of the software driver.

Workaround

Application specific code should be used in place of the example files provided with the software driver. Note the files al3_dpr.c, al3_dpr.h, al3_isr.c and al3_isr.h in the beta-1.0 software driver release currently contain inefficiencies and errors.

Performance With Workaround

Application specific code should be used in place of the example files provided with the software driver.

Performance Without Workaround

Application specific code should be used in place of the example files provided with the software driver. Note the files al3_dpr.c, al3_dpr.h, al3_isr.c and al3_isr.h in the beta-1.0 software driver release currently contain inefficiencies and errors.

3.8 al3addq – Erroneous Parameters

Description

While testing mixed signaling and non-signaling queues on a DS1 SDF-MF line, the al3ActivateChannel(), al3ActivateChannelStr(), al3ActivateChannelUnstr(), al3EnhancedActivateChannel(), al3EnhancedActivateChannelStr(), and al3EnhancedActivateUnstr() functions calculate some values incorrectly. The following fields in the transmit and receive queues are incorrect:

1. T_CHAN_NO_SIG set to '0'.

2. R_TOT_SIZE set to size calculated with the SDF-MF formula.

Workaround

1) In al3_chan.c, in the function aalUpdateRxTable(), change the calculation for R_TOT_SIZE(9:0) to:

```
**************
if (pRxQDB->modeFR ¦¦ pRxQDB->suppressSignaling)
{
temp = numChans - 1;
}
else if (pRxQDB->modeT1 ¦¦ pRxQDB->t1Sig)
{
temp = (((numChans + 1) / 2) - 1) + (numChans * 24);
}
else
{
temp = (((numChans + 1) / 2) - 1) + (numChans * 16);
}
*********
```

2) In the file al3_chan.c, in the function aalUpdateRxTable(), change the calculation for R_CHAN_NO_SIG(14) to:

```
*********
if (!(pRxQDB->modeFR) && pRxQDB->suppressSignaling) data += (1 << 14);
*********
```

3) In the file al3_chan.c, in the function aalUpdateTxTable(), change the calculation for T_CHAN_NO_SIG(7) to:

```
********
if (!(pTxQDB->modeFR) && pTxQDB->suppressSignaling) data += (1 << 7);
********
```

4) In the file al3_mdb.h, both structures, al3_queue_db_rx and al3_queue_db_tx, need an additional element added (can be added anywhere within the structure):

BOOLEAN t1Sig; /* Is T1 signalling used? */

5) This modification is only required if using T1 signaling over E1. In the file al3_chan.c, in the function aalActivateQueueStr(), the following lines need to be added - an existing line is included that shows the setup of "modeT1" as a guide for where the additional lines go:

existing line:

pRxQDB->modeT1 = (pRxLine->t1Mode ? TRUE : FALSE);

add this line:

pRxQDB->t1Sig = (pRxLine->sigType ? TRUE : FALSE);

existing line:

pTxQDB->modeT1 = (pTxLine->t1Mode ? TRUE : FALSE);

add this line:

pTxQDB->t1Sig = (pTxLine->sigType ? TRUE : FALSE);

Performance With Workaround

Mixed signaling and non-signaling queues operate normally.

Performance Without Workaround

Queues do not function as expected - cells get dropped and underrun conditions result.

3.9 al3setlinestrmode – Missing Register Write

Description

When using the driver API function “al3SetLineMode“ the driver does not set the “Call Attention Bit”, which is required for this command to take effect.

Workaround

In the file al3_init.c, in the function aalInitRegsLINE(), add the following to the very end of the function:

****
while (sysAl3ReadReg(pDDB->baseAddr, (UINT4)AL3_A0_CMD + aspNum) & AL3_AX_CMD_ATTN);

```
sysAl3WriteReg(pDDB->baseAddr, (UINT4 )AL3_A0_CMD + aspNum,
AL3_AX_CMD_ATTN);
********
```

Performance With Workaround

al3SetLineMode operates normally.

Performance Without Workaround

al3SetLineMode may not operate as expected due to the fact that the “Call Attention Bit”, which is required in order for the command to take effect, may not get set .

3.10 Example Profile #3 – T_CHAN_UNSTRUCT Erroneously Set

Description

In example profile # 3 which sets up the AAL1gator-32 device in T1-SDF-MF mode with CAS, the commands al3ActivateChannel(), al3ActivateChannelStr(), al3ActivateChannelUnstr(), al3EnhancedActivateChannel(), al3EnhancedActivateChannelStr() and al3EnhancedActivateUnstr(),are found to have the T_CHAN_UNSTRUCT bit in the QUEUE_CONFIG register set. This is an error – this bit should only be set for single DS0 without pointer in SDF-FR mode.

Workaround

In the file al3_chan.c, in the function aalUpdateRxTable(), change the calculation of R_CHAN_UNSTRUCT(15) to:

```
********
if ((numChans == 1) && (pRxQDB->modeFR ¦¦ pRxQDB->supressSignaling))
data += (1 << 15);
********
```

In the file al3_chan.c, in the function aalUpdateTxTable(), change the calculation of T_CHAN_UNSTRUCT(6) to:

```
********
if ((numChans == 1) && (pTxQDB->modeFR ¦¦ pTxQDB->supressSignaling))
data += (1 << 6);
```

****

Performance With Workaround

Example profile #3 operates as expected.

Performance Without Workaround

Example profile #3 which sets up the AAL1gator-32 device in T1-SDF-MF mode with CAS will not operate as expected due to the fact that the T_CHAN_UNSTRUCT bit is erroneously set.

3.11 al3ActivateChannel Returns a Queue Handle Usable Only in the Receive Direction

Description

The command al3ActivateChannel returns a queue handle which is usable only in the receive direction – not the transmit direction.

Workaround

Add the following function aalExtractQueId to al3_api2.c and use this function before calling any Transmit API commands. This function obtains the TX Queue Id from the default queue ID returned by the Activate channel API functions.

```
/*-----------------------------------------------------------------------**
**
**  aalExtractQueId
**
_____________________________________________________________________
_____________
**
**  DESCRIPTION:   Extracts Rx and Tx Queue ID (pointers)
**
**  PARAMETERS:    pDDB - device Handle (from al3Add())
**                 queId - original queue ID
**                 pRxQDB - (pointer to) extracted Rx Queue ID variable
**                 pTxQDB - (pointer to) extracted Tx Queue ID variable
**
**  OUTPUTS:       pRxQDB - extracted Rx Queue ID
**                 pTxQDB - extracted Tx Queue ID
```

```
**
**  RETURNS:       True if parameter 'queId' is invalid
**
**  VALID STATES:  (states not checked)
**
**  SIDE EFFECTS:  A Rx Queue MAY be linked to a Tx Queue
**                 A Tx Queue is NOT linked to a Rx Queue
**
**  PSEUDOCODE:    Begin
**                 End
**
**-----------------------------------------------------------------------------*/
BOOLEAN aalExtractQueId (sAL3_DDB *pDDB, sAL3_QID queId,
sAL3_QDB_RX **pRxQDB, sAL3_QDB_TX **pTxQDB)
 {
   if (queId == NULL) return TRUE;

   if ((UINT4 )queId & 0x80000000)
     {
       if (pTxQDB != NULL) *pTxQDB = (sAL3_QDB_TX *)((UINT4 )queId &
~0x80000000);
       if (pRxQDB != NULL) *pRxQDB = NULL;
     }
   else
     {
       if (pTxQDB != NULL)
         {
           UINT2 txQueue = ((sAL3_QDB_RX *)queId)->txQueue;

           if (txQueue >= pDDB->numQUE) *pTxQDB = NULL;
           else *pTxQDB = &pDDB->a1sp[((txQueue >> 8) &
AL3_A1SP_MASK)].line[((txQueue >> 5) & AL3_LINE_MASK)].txQue[(txQueue
& AL3_QUE_MASK)];
         }

       if (pRxQDB != NULL) *pRxQDB = (sAL3_QDB_RX *)queId;
     }

   return FALSE;
 }                                        /* aalExtractQueId() */
```

Performance With Workaround

A valid TX Queue Id is obtained by calling aalExtractQueId. With this TX Queue Id Transmit API commands operate normally.

Performance Without Workaround

Without the workaround, Transmit API commands may not operate normally due to an invalid TX Queue Id.

CONTACTING PMC-SIERRA, INC.

PMC-Sierra, Inc.
105-8555 Baxter Place Burnaby, BC
Canada V5A 4V7

Tel: (604) 415-6000

Fax: (604) 415-6200

Document Information: document@pmc-sierra.com
Corporate Information: info@pmc-sierra.com
Application Information: apps@pmc-sierra.com
(604) 415-4533
Web Site: http://www.pmc-sierra.com

PMC- 2001614 (P2) Issue date: June 2001