CREDITNET SWITCH USER GUIDE

Version 1.0

Gary Depelteau [depeltea@bnr.ca]

Thomas NG [ngt@bnr.ca]

Bell Northern Research

This document

This document is the user guide of the CreditNet ATM switch co-developed by Harvard University and Bell-Northern Research.

1.0 Introduction

This is the User Guide for Creditnet ATM Switch (also known as ARPA Switch). This guide describes the installation procedure of the switch software, the configuration of the GPU, the start-up script file, GPU memory upgrade. It also includes a printed version of the on-line manual pages of the switch software.

2.0 Release Notes

The latest version of Creditnet Switch SW was released on August 24 95. This note serves to highlight some of the new features of this release. It also serves to explain the partitioning of the S/W developed by BNR, how to find functions that you may be looking for and how to use the man pages.

The main features of this new release of Creditnet Switch S/W are:

1. A more complete set of SCU Drivers and diagnostics.
2. A new module which performs complete SCU initialization to default initial conditions.
3. A sysInitCtl() function which now performs all the CM and PM initialization and diagnostics for which BNR is responsible.
4. A set of Interrupt Service Routines and some monitoring capabilities which report on the status of switch interrupts received by the GPU from the SCU. The only interrupts are the SCU CDM Chalkmark search stale CADR interrupt, SCU CDM Chalkmark search done interrupt and the SCU CTC interval timing complete interrupt.
5. A set of PCC routines were added. That includes fillPcc, testChecksums, several pcc data bus tests and a repeat mode on PCC.
6. A small routine for non-flow control VC setup (nfvc.c) and a function for setting up PM (ij.c)were added as examples.
7. On-line manpages for all S/W that have been written. They are associated with the modules listed below.

2.1 Partition of S/W

The partition of CreditNet Switch S/W is based upon individual hardware block, i.e. GPU, SCM and PM.

2.2 New Release Contents

This new release can be found in:YOUR_BASE_DIR/Release and it includes the following major object files:

gpu.o: composed of scmUtil, flashCtl, gpuUIinterface, xmemTest, gpuDiag, alias, cell.

scm.o: composed of scuDrv, scuInit, scuIntLib, intMonLib, scuSCoreTest, scuSMDiag, scuSDiag, scuHVDiag, sysInitCtl.

gpuVW.o: linked to VxWorks so there is no need to download this. Its composed of sccLib, gpuWdLib, gpuNmiLib.

gpuVW_res.o: linked to VxWorks bootroms so there is no need to download this. It is replaced when the actual VxWorks image including gpuVW.o is downloaded. It is composed of fDramTest, scmUtil_res, gpuDiag_res, sccLib, gpuWdLib, gpuNmiLib.

nfvc.o: composed of nfvc, a function for setting up non-flow control VC.

pmAll.o: composed of pccUtil, pmCtrl, vxServer, pmVCsetup, pccDiag.

pmConf.o: composed of PM configuration functions.

2.3 WARNING

Mixing different versions of the Creditnet Switch SW probably will not work since there are some very painful work-around needed to avoid dependencies.

3.0 Installation procedures

A file called Released.tar.zip (or Released.tar.gz) that contains some essential SW for the switch to run including the VxWorks image is shipped along with the switch. The installation procedure of the SW is as below:

- In case of a zipped file, i.e. with filename Release.tar.zip, do the following:

  % unzip Released.tar.zip

  % tar xvf Released.tar
- In case of a gzip'ed file, i.e. with the name Release.tar.gz, do the following:

  % gunzip Released.tar.gz

  % tar xvf Released.tar

It should then produce a directory tree such as the one below:

```
./          DOCUMENTATION/   PM.rom/   TAGS     UI/          man/              startup.scr*    vcset.scr
../         GPU/             SCH/      TLG/     include/     nfvc.scr          switchConf.c
COMIF/      Makefile*        SCM/      TOOLS/   loadup.scr   scuHVEncDecTest   switchConf.o

COMIF:
./          Makefile       comClient.o    comServ.img*   comServ.out*   comServ.x1*
../         comClient.c*   comServ.c*     comServ.o      comServ.x

DOCUMENTATION:
./      ../                HW_DOCS/                SW_DOCS/

DOCUMENTATION/HW_DOCS:
./      ../

DOCUMENTATION/SW_DOCS:
./      VxWorks/                 scuDiagnostics/              userGuide/
../     gpuDiagnostics/          systemInitialization/

DOCUMENTATION/SW_DOCS/VxWorks:
./      ../      README*

DOCUMENTATION/SW_DOCS/gpuDiagnostics:
./      GpuDiagnostics.fbbook*   GpuDiagnosticsDD.doc*    GpuDiagnosticsTI.doc*
../     GpuDiagnostics.ps*       GpuDiagnosticsFN.doc*    GpuDiagnosticsTOC.doc*

DOCUMENTATION/SW_DOCS/scuDiagnostics:
./      ScuDiagnostics.fbbook*   ScuDiagnosticsDD.doc*    ScuDiagnosticsTI.doc*
../     ScuDiagnostics.ps*       ScuDiagnosticsFN.doc*    ScuDiagnosticsTOC.doc*

DOCUMENTATION/SW_DOCS/systemInitialization:
./      ../      SysInit.fbbook*   SysInit.ps*   SysInitDD.doc*   SysInitFN.doc*   SysInitTI.doc*   SysInitTOC.doc*

DOCUMENTATION/SW_DOCS/userGuide:
./      Darpa.userguide.TOC.doc   Darpa.userguide.ps         Darpa.userguide.complete.ps      Darpa.userguide.manpage.pcc.doc
../     Darpa.userguide.book      Darpa.userguide.intro.doc  Darpa.userguide.manpage.scu.doc

GPU:
./          flashes/       gpuVW.o*       ij.o       intAll.o*      nfvc.o      pmAll.o*     vxWorks*
```

```
../             gpu.o*          gpuVW_res.o*     int1.o          lc.o            nfvc.o.bak     pmConf.c*     vxWorks.sym*

GPU/flashes:
./              README              br0.bin*            br1.bin*            cardid_gpu0.bin*
../             bootrom_res.bin*    br0.hex*            br1.hex*            cardid_gpu0.hex*

PM.rom:
./              ../             bootDpram.img*    bootRom.rom*    bootSCH.img     checksum.out*

SCH:
./                 bootSCH.x            i_handle_SCH.s*    schAlgo1.s     schAlgo8.o
../                bootSCH.x1*          i_table_SCH.o      schAlgo2.c*    schAlgo8.s
Makefile*          boot_dmn_SCH.o       i_table_SCH.s*     schAlgo2.o     schWW.o
Makefile~*         boot_dmn_SCH.s*      out                schAlgo2.s     schWW.s
SCH.c*             checksum.sun*        schAlgo0.o*        schAlgo3.o     schWatchdog.o
SCH.ld*            dmn_ftbl_SCH.o       schAlgo0.s*        schAlgo3.s     schWatchdog.s
SCH.o              dmn_ftbl_SCH.s*      schAlgo0.s.bak*    schAlgo4.o     startup_SCH.o
SCHdpram.inc*      dmn_ptbl_SCH.o       schAlgo1.c*        schAlgo4.s     startup_SCH.s*
bootSCH.img        dmn_ptbl_SCH.s*      schAlgo1.c~*       schAlgo6.o     x
bootSCH.out*       i_handle_SCH.o       schAlgo1.o         schAlgo6.s     xice.cfg*

SCM:
./                 ../              scm.o*

TLG:
./                 ../              tlg.o*

TOOLS:
./                 ../              checksum.sun*

UI:
./                 ../              combined*            mainprog*              switchDaemon*          uiconfig.f
uihelp.f           vxshell*         wish*

include:
./                 comIF.h*          intMonLib.msg.h*      pmCtrl.h*               scuHVDiag.h              scuSDiag.msg.h*
../                configFLEX.h*     ispUtil.h             pmVCsetup.h*            scuHVDiag.msg.h*         scuSMDiag.h
PCCregions.h*      dctest.h*         isplsi.h              regions.h*sculnit.h*    scuSMDiag.msg.h
SCHError.h*        flashCtl.h*       mcpPcMemoryMap.h*     sccLib.h*sculnit.msg.h  scuWDbmIntegTest.h
SCHregions.h*      generateCell.h*   nfvc.h                schMemoryMap.h*         scuIntLib.h*             simpleCom.h*
bootError.h*       gpuDiag.h*        pcMemoryMap.h*        scmUtil.h               scuIntLib.msg.h*         sysInitCtl.h
bus.h*             gpuNmiLib.h*      pcServices.h*         scuDaisyChainTest.h*    scuPortCardTest.h*       tlgDriver.h
cell.h             gpuWdLib.h*       pccDiag.h*            scuDrv.h                scuSCoreTest.h*          tucServices.h*
comError.h*        intMonLib.h*      pccUtil.h*            scuFifoIntegTest.h      scuSDiag.h               xmemTest.h*

man:
./                 ../               cat1/                 cat2/                   man1/                    man2/
wrs.an*

man/cat1:
./                 generateCell.1*   ispUtil.1*            scuDrv.1*               scuPortCardTest.1*       switchConf.1*
../                gpuDiag.1*        pccUtil.1             scuFifoIntegTest.1*     scuSCoreTest.1*          sysInitCtl.1*
alias.1*           gpuNmiLib.1*      sccLib.1*             scuHVDiag.1*            cuSDiag.1*
fDramTest.1*       gpuWdLib.1*       scmUtil.1*            sculnit.1*              scuSMDiag.1*
flashCtl.1*        intMonLib.1*      scuDaisyChainTest.1*  scuIntLib.1*            scuWDbmIntegTest.1*

man/cat2:
./                          sccOutputChar.2*                  scuCtcStopRegRead.2*        scuSCdmCmSrchTest.2*
../                         sccPrintHexWord.2*                scuCurChalkRead.2*          scuSCdmCmUpdTest.2*
Copyright.2*                sccPrintString.2*                 scuCurChalkSet.2*           scuSCdmMonFATest.2*
cellCrc8.2*                 sccWait.2*                        scuDaisyChainTest.2*        scuSChalkmarkCheck.2*
checkPortCardExist.2        scmChkDramParity.2*               scuDbmCapture.2*            scuSCifCrc8Test.2*
crc16.2*                    scmChkFlashSig.2*                 scuDbmInit.2*               scuSCifDataPathTest.2*
displayChecksums.2          scmMarch6Test.2*                  scuDbmSet.2*                scuSCifFifoTest.2*
downloadToPcc.2*            scmMicrosecondWait.2*             scuDbmSetup.2*              scuSCoreTest.2*
dp.2*                       scmRegTest.2*                     scuDefDSInit.2*             scuSCtclt1Test.2*
dw.2*                       scmVerify.2*                      scuFanAlarmISR.2*           scuSCtclt2Test.2*
fDramTest.2*                scmWalkA1Test.2*                  scuFifoIntegTest.2*         scuSDiagCtl.2*
fakeFLMGetNextCADR.2*       scmWalkD0Test.2*                  scuFifoSeqTest.2*           scuSFifoStatCheck.2*
fakeFLMInit.2*              scmWalkD1Test.2*                  scuFlmCmbfCountRead.2*      scuSFlmBasicTest.2*
fillPcc.2                   scu64BCellDecode.2*               scuFlmCntlRegRead.2*        scuSFlmIntegTest.2*
flashAddRamSize.2*          scuACAlarmISR.2*                  scuFlmCntlRegWrite.2*       scuSMCdmConCapTest.2*
flashBootParamsSet.2*       scuCMCompareCell.2*               scuFlmEmptyISR.2*           scuSMCdmRegTest.2*
flashCardIdUsage.2*         scuCMTestEntireCore.2*            scuFlmFLArrayPush.2*        scuSMCifRegTest.2*
flashCheckErase.2*          scuCMTestGeneratePatterns.2*      scuFlmFLCInit.2*            scuSMCtcRegTest.2*
flashCopyFromRam.2*         scuCMTestLostAddresses.2*         scuFlmFLPush.2*             scuSMDbmRamTest.2*
flashDLoad.2*               scuCMTestOneAddress.2*            scuFlmFailISR.2*            scuSMFlmRamStatTest.2*
flashDataPoll.2*            scuCMWalkingTest.2*               scuFlmFreeListInit.2*       scuSMFlmRegTest.2*
flashErase.2*               scuCbtErrISR.2*                   scuFlmFullISR.2*            scuSMPbusRegTest.2*
flashProgAltBank.2*         scuCdmCntlRegRead.2*              scuFlmHalt.2*               scuSMRivcLutTest.2*
flashSetRamSize.2*          scuCdmCntlRegWrite.2*             scuFlmHaltClear.2*          scuSMSharBuffTest.2*
generateCell.2*             scuCdmInit.2*                     scuFlmInit.2*               scuSMWDbmRamTest.2*
gpuCardIdTest.2*            scuCdmStaleCadrRegRead.2*         scuFlmPrime.2*              scuSMWivcLutTest.2*
gpuCntrlRegTest.2*          scuCdmStatRegRead.2*              scuFlmReset.2*              scuSRegCheck.2*
gpuDiagCtl.2*               scuCellCpyRamToSb.2*              scuFlmStatRegRead.2*        scuSSBByteCheck.2*
```

gpuDipSwTest.2*	scuCellCpySbToRam.2*	scuFlmWrapISR.2*	scuShortPortCardTest.2*
gpuEDramTest.2*	scuCellXferFromCore.2*	scuFullPortCardTest.2*	scuStaleCadrISR.2*
gpuEepromTest.2*	scuCellXferToCore.2*	scuGetCellFromCore.2*	scuVerCorePat.2*
gpuErrRegTest.2*	scuCellXferToFromCore.2*	scuGetCellsFromCore.2*	scuVerify.2*
gpuEtherTest.2*	scuChalkSearchDisable.2*	scuHVCdmPETest.2*	scuWDbmCapture.2*
gpuExtBusTest.2*	scuChalkSearchEnable.2*	scuHVCifPETest.2*	scuWDbmIntegTest.2*
gpuFlSwapTest.2*	scuChalkSearchReset.2*	scuHVCrmBFStatTest.2*	scuWDbmMultiCastTest.2*
gpuNmiInit.2*	scuCifCntlRegRead.2*	scuHVCrmPETest.2*	scuWDbmWalkingOneTest.2*
gpuNmiInitIsr.2*	scuCifCntlRegWrite.2*	scuHVCtcItPETest.2*	scuWDbmWalkingZeroTest.2*
gpuNmiMon.2*	scuCifHPWFifoStatRead.2*	scuHVDS1TogCheck.2*	scuWFifoHPAlFullISR.2*
gpuNmiMonStart.2*	scuCifInit.2*	scuHVEncDecTest.2*	scuWFifoHPMidISR.2*
gpuNmiMonStop.2*	scuCifLPWFifoStatRead.2*	scuHVFlmCmbfPETest.2*	scuWFifoLPAlFullISR.2*
gpuNmiMonTask.2*	scuCifRFifoStatRead.2*	scuHVIdLogTest.2*	scuWFifoLPMidISR.2*
gpuSccTest.2*	scuCifRcvCrc8ErrISR.2*	scuHVIdleTest.2*	scuWivcLutInit.2*
gpuTimIntTest.2*	scuCifRdPoll.2*	scuHVPMIntTest.2*	scuWivcLutSet.2*
intCntrMon.2*	scuCifRdWrPoll.2*	scuHVPMResetTest.2*	scuWivcLutSetup.2*
intMonInit.2*	scuCifRivcFifoRead.2*	scuHVWCadrPETest.2*	scuWriteXCells.2*
intMonPerSet.2*	scuCifStatRegRead.2*	scuHwInit.2*	setOwnershipTable.2*
ispEnValCalc.2*	scuCifWivcFifoRead.2*	scuInitCtl.2*	setPM.2*
ispProgram.2*	scuCifWrPoll.2*	scuInitPortCardTest.2*	setPort.2*
ispRead.2*	scuCoreCellRead.2*	scuIntInit.2*	swapCell.2*
ispVerify.2*	scuCoreCellWrite.2*	scuIntMon.2*	sysInitCtl.2*
pccAddressTest.2	scuCoreClkAlarmISR.2*	scuItTimCompISR.2*	testChecksums.2
pccMarch6Test.2	scuCoreMemoryCompareCell.2*	scuIvcInit.2*	ur.2*
pccWalkD0Test.2	scuCoreMemoryTestEntireCore.2*	scuIvcSet.2*	wdFeeder.2*
pccWalkD1Test.2	scuCoreMemoryTestGeneratePatterns.2*	scuIvcSetup.2*	wdFeederHookAdd.2*
printCellStruct.2*	scuCoreMemoryTestLostAddresses.2*	scuMap8To7.2*	wdFeederHookDelete.2*
rFifoAlFullFlag.2*	scuCoreMemoryTestOneAddress.2*	scuPSAlarmISR.2*	wdFeederHookShow.2*
ramDebug.2*	scuCoreMemoryWalkingTest.2*	scuParCrcInit.2*	wdFeederInit.2*
repeatModeOff.2	scuCoreParErrISR.2*	scuPifInit.2*	wdFeederTask.2*
repeatModeOn.2	scuCrmInit.2*	scuPutCellInCore.2*	wdRestartFeeder.2*
resetScheduler.2*	scuCtcInit.2*	scuRFifoAlFullISR.2*	wdStopFeeder.2*
rp.2*	scuCtcIntTimProg.2*	scuRFifoMidISR.2*	wp.2*
rs.2*	scuCtcIntTimRead.2*	scuReadXCells.2*	ws.2*
runPccMemoryTest.2*	scuCtcRealTimProg.2*	scuRivcLutInit.2*	
runPccService.2	scuCtcStartRegRead.2*	scuRivcLutSet.2*	
sccInit.2*	scuCtcStatRegRead.2*	scuRivcLutSetup.2*	

man/man1:

./	generateCell.1*	ispUtil.1*	scuDrv.1*	scuPortCardTest.1*	switchConf.1*
../	gpuDiag.1*	pccUtil.1*	scuFifoIntegTest.1*	scuSCoreTest.1*	sysInitCtl.1*
alias.1*	gpuNmiLib.1*	sccLib.1*	scuHVDiag.1*	scuSDiag.1*	
fDramTest.1*	gpuWdLib.1*	scmUtil.1*	scuInit.1*	scuSMDiag.1*	
flashCtl.1*	intMonLib.1*	scuDaisyChainTest.1*	scuIntLib.1*	scuWDbmIntegTest.1*	

man/man2:

./	sccInit.2*	scuCtcStopRegRead.2*	scuSCdmCmSrchTest.2*
../	sccOutputChar.2*	scuCurChalkRead.2*	scuSCdmCmUpdTest.2*
Copyright.2*	sccPrintHexWord.2*	scuCurChalkSet.2*	scuSCdmMonFATest.2*
cellCrc8.2*	sccPrintString.2*	scuDaisyChainTest.2*	scuSChalkmarkCheck.2*
checkPortCardExist.2	sccWait.2*	scuDbmCapture.2*	scuSCifCrc8Test.2*
crc16.2*	scmChkDramParity.2*	scuDbmInit.2*	scuSCifDataPathTest.2*
displayChecksums.2	scmChkFlashSig.2*	scuDbmSet.2*	scuSCifFifoTest.2*
downloadToPcc.2*	scmMarch6Test.2*	scuDbmSetup.2*	scuSCoreTest.2*
dp.2*	scmMicrosecondWait.2*	scuDefDSInit.2*	scuSCtcIt1Test.2*
dw.2*	scmRegTest.2*	scuFanAlarmISR.2*	scuSCtcIt2Test.2*
fDramTest.2*	scmWalkA1Test.2*	scuFifoIntegTest.2*	scuSDiagCtl.2*
fakeFLMGetNextCADR.2*	scmWalkD0Test.2*	scuFifoSeqTest.2*	scuSFifoStatCheck.2*
fakeFLMInit.2*	scmWalkD1Test.2*	scuFlmCmbfCountRead.2*	scuSFlmBasicTest.2*
fillPcc.2	scu64BCellDecode.2*	scuFlmCntlRegRead.2*	scuSFlmIntegTest.2*
flashAddRamSize.2*	scuACAlarmISR.2*	scuFlmCntlRegWrite.2*	scuSMCdmConCapTest.2*
flashBootParamsSet.2*	scuCMCompareCell.2*	scuFlmEmptyISR.2*	scuSMCdmRegTest.2*
flashCardIdUsage.2*	scuCMTestEntireCore.2*	scuFlmFLArrayPush.2*	scuSMCifRegTest.2*
flashCheckErase.2*	scuCMTestGeneratePatterns.2*	scuFlmFLCInit.2*	scuSMCtcRegTest.2*
flashCopyFromRam.	scuCMTestLostAddresses.2*	scuFlmFLPush.2*	scuSMDbmRamTest.2*
flashDLoad.2*	scuCMTestOneAddress.2*	scuFlmFailISR.2*	scuSMFlmRamStatTest.2*
flashDataPoll.2*	scuCMWalkingTest.2*	scuFlmFreeListInit.2*	scuSMFlmRegTest.2*
flashErase.2*	scuCbtErrISR.2*	scuFlmFullISR.2*	scuSMPbusRegTest.2*
flashProgAltBank.2*	scuCdmCntlRegRead.2*	scuFlmHalt.2*	scuSMRivcLutTest.2*
flashSetRamSize.2*	scuCdmCntlRegWrite.2*	scuFlmHaltClear.2*	scuSMSharBuffTest.2*
generateCell.2*	scuCdmInit.2*	scuFlmInit.2*	scuSMWDbmRamTest.2*
gpuCardIdTest.2*	scuCdmStaleCadrRegRead.2*	scuFlmPrime.2*	scuSMWivcLutTest.2*
gpuCntrlRegTest.2*	scuCdmStatRegRead.2*	scuFlmReset.2*	scuSRegCheck.2*
gpuDiagCtl.2*	scuCellCpyRamToSb.2*	scuFlmStatRegRead.2*	scuSSBByteCheck.2*
gpuDipSwTest.2*	scuCellCpySbToRam.2*	scuFlmWrapISR.2*	scuShortPortCardTest.2*
gpuEDramTest.2*	scuCellXferFromCore.2*	scuFullPortCardTest.2*	scuStaleCadrISR.2*
gpuEepromTest.2*	scuCellXferToCore.2*	scuGetCellFromCore.2*	scuVerCorePat.2*
gpuErrRegTest.2*	scuCellXferToFromCore.2*	scuGetCellsFromCore.2*	scuVerify.2*
gpuEtherTest.2*	scuChalkSearchDisable.2*	scuHVCdmPETest.2*	scuWDbmCapture.2*
gpuExtBusTest.2*	scuChalkSearchEnable.2*	scuHVCifPETest.2*	scuWDbmIntegTest.2*
gpuFlSwapTest.2*	scuChalkSearchReset.2*	scuHVCrmBFStatTest.2*	scuWDbmMultiCastTest.2*
gpuNmiInit.2*	scuCifCntlRegRead.2*	scuHVCrmPETest.2*	scuWDbmWalkingOneTest.2*
gpuNmiInitIsr.2*	scuCifCntlRegWrite.2*	scuHVCtcItPETest.2*	scuWDbmWalkingZeroTest.2*
gpuNmiMon.2*	scuCifHPWFifoStatRead.2*	scuHVDS1TogCheck.2*	scuWFifoHPAlFullISR.2*
gpuNmiMonStart.2*	scuCifInit.2*	scuHVEncDecTest.2*	scuWFifoHPMidISR.2*

gpuNmiMonStop.2*
gpuNmiMonTask.2*
gpuSccTest.2*
gpuTimIntTest.2*
intCntrMon.2*
intMonInit.2*
intMonPerSet.2*
ispEnValCalc.2*
ispProgram.2*
ispRead.2*
ispVerify.2*
pccAddressTest.2
pccMarch6Test.2
pccWalkD0Test.2
pccWalkD1Test.2
printCellStruct.2*
rFifoAlFullFlag.2*
ramDebug.2*
repeatModeOff.2
repeatModeOn.2
resetScheduler.2*
rp.2*
rs.2*
runPccMemoryTest.2*
runPccService.2
scuCifLPWFifoStatRead.2*
scuCifRFifoStatRead.2*
scuCifRcvCrc8ErrISR.2*
scuCifRdPoll.2*
scuCifRdWrPoll.2*
scuCifRivcFifoRead.2*
scuCifStatRegRead.2*
scuCifWivcFifoRead.2*
scuCifWrPoll.2*
scuCoreCellRead.2*
scuCoreCellWrite.2*
scuCoreClkAlarmISR.2*
scuCoreMemoryCompareCell.2*
scuCoreMemoryTestEntireCore.2*
scuCoreMemoryTestGeneratePatterns.2*
scuCoreMemoryTestLostAddresses.2*
scuCoreMemoryTestOneAddress.2*
scuCoreMemoryWalkingTest.2*
scuCoreParErrISR.2*
scuCtcInit.2*
scuCtcIntTimProg.2*
scuCtcIntTimRead.2*
scuCtcRealTimProg.2*
scuCtcStartRegRead.2*
scuCtcStatRegRead.2*
scuHVFlmCmbfPETest.2*
scuHVIdLogTest.2*
scuHVIdleTest.2*
scuHVPMIntTest.2*
scuHVPMResetTest.2*
scuHVWCadrPETest.2*
scuHwInit.2*
scuInitCtl.2*
scuInitPortCardTest.2*
scuIntInit.2*
scuIntMon.2*
scuItTimComplISR.2*
scuIvcInit.2*
scuIvcSet.2*
scuIvcSetup.2*
scuMap8To7.2*
scuPSAlarmISR.2*
scuPifInit.2*
scuPutCellInCore.2*
scuRFifoAlFullISR.2*
scuRFifoMidISR.2*
scuReadXCells.2*
scuRivcLutInit.2*
scuRivcLutSet.2*
scuRivcLutSetup.2*
scuWFifoLPAlFullISR.2*
scuWFifoLPMidISR.2*
scuWivcLutInit.2*
scuWivcLutSet.2*
scuWivcLutSetup.2*
scuWriteXCells.2*
setOwnershipTable.2*
setPM.2*
setPort.2*
swapCell.2*
sysInitCtl.2*
testChecksums.2
ur.2*
wdFeeder.2*
wdFeederHookAdd.2*
wdFeederHookDelete.2*
wdFeederHookShow.2*
wdFeederInit.2*
wdFeederTask.2*
wdRestartFeeder.2*
wdStopFeeder.2*
wp.2*
ws.2*

3.1 Configuration procedures of GPU

This section describes the process of booting the CreditNet Switch up to the point where Harvard University Application SW may be downloaded. It assumes that the BNR portion of the Switch SW has been installed in a directory visible on the Switch Workstation. This section aims to bring out the effort required to get the switch to boot but does not give an over all or detailed view of complete switch initialization. For this type of information please refer to "CreditNet Switch System Initialization".

The first step is to get the GPU on the switch to download the VxWorks image from the workstation via the ethernet port of the GPU. It requires communicating with the GPU via its RS-232 serial port. The following is a step by step guide for initializing the board's booting parameters. After it is done, the GPU will instead download VxWorks from the workstation using the provided IP address and target node name through the ethernet port. Note that anything preceded by // is a comment.

The next thing to do is to run a startup script called *startup.scr* which downloads BNR developed SW. The script file will execute switch diagnostics and some initialization on the portcards. After the diagnostics and initialization are finished, run a function called *flashBootParamsSet()* to 'permanently'[1] change the GPU boot parameters in its Flash EEPROMS. Here's how it should go:

```
*****************************************************************
                    Booting GPU960
                From Primary Bootrom Bank

Testing current bootrom bank............Passed!
Testing Card ID EEPROM..................Passed!
Testing dynamic RAM (fast test).........Passed!
Testing real-time system clock..........Passed!
Testing ethernet controller.............Passed!
```

1. The boot parameters are stored in flash memory, i.e. the parameters are retained in the memory when the switch is powered off. The flash memory can be reprogrammed using the *flashBootParamsSet()* function as many times as required.

```
Testing serial controller...............Passed!
Testing NMI error source register.......Failed!
Testing control register................Passed!
Testing flash bank swapping.............Passed!

Round-robin scheduling started - time-slices are 400 milliseconds

GPU sanity watchdog feeding task started every 500 milliseconds.

NMI ISR installed and NMIs enabled.


                    VxWorks System Boot


Copyright 1984-1993  Wind River Systems, Inc.



CPU: GPU960
Version: 5.1.1
BSP version: 1.0 Beta-4
Creation date: Tue Mar 7 15:26:32 EST 1995


// YOU MUST HIT A KEY TO STOP THE BOOT PROCESS AND CHANGE THE
// BOOT PARAMETERS BEFORE THE FOLLOWING COUNT DOWN EXPIRES.
// THE BEST WAY IS TO START HITTING THE SPACE BAR AS SOON AS
// THE ABOVE DIAGNOSTICS FINISH.

Press any key to stop auto-boot...
 1
[VxWorks Boot]: c    // THIS COMMAND STARTS BOOT PARAMETERS CHANGE DIALOG
                       // YOU SHOULD TYPE IN YOUR OWN STUFF WHEREVER YOU
                      // SEE STUFF IN CAPITAL LETTERS.

'.' = clear field;  '-' = go to previous field;  ^D = quit

boot device          : ei
processor number     : 0
host name            : bcars782b                                    // PLACE YOUR HOST NAME HERE
file name            : /bnr/3rdparty/glan_sw/WindRiver/vw/config/gpu2960/vxWorks   //  BASE DIR/Release/GPU/vxWorks
inet on ethernet (e) : 47.53.32.8:ffff0000                          // THE IP ADDR YOU SUPPLY FOR GPU
inet on backplane (b):
host inet (h)        : 47.53.32.1                                   // THE IP ADDR OF THE HOST WORKSTATION
gateway inet (g)     :
user (u)             : nobody                                       // THE USER ID YOU USE TO DOWNLOAD SW
ftp password (pw) (blank = use rsh): nobody                        // PASSWORD CORRESPONDING TO ABOVE
flags (f)            : 0x8
target name (tn)     : gpu5                                         // THE HOST NAME OF YOU GPU
startup script (s)   :
other (o)            :


[VxWorks Boot]: @                    // INSTRUCT THE GPU TO CONTINUE BOOTING WITH NEW PARAMS

// IF EVERYTHING IS OK YOU SHOULD SEE THE FOLLOWING:


boot device          : ei
processor number     : 0
host name            : YOUR HOST NAME
file name            : YOUR BASE DIR/Release/GPU/vxWorks
inet on ethernet (e) : THE IP ADDRESS YOU SUPPLY FOR GPU
host inet (h)        : THE IP ADDRESS OF YOUR HOST WORKSTATION
user (u)             : THE USER ID YOU USE TO DOWNLOAD SW
ftp password (pw)    : PASSWORD CORRESPONDING TO ABOVE
flags (f)            : 0x8
target name (tn)     : THE HOST NAME OF YOU GPU

Attaching network interface ei0... done.
Attaching network interface lo0... done.
Loading... 492216 + 34104 + 53568                  // THIS MEANS YOU'VE DOWNLOADED VXWORKS
```

```
Starting at 0x10001000...


Round-robin scheduling started - time-slices are 400 milliseconds

GPU sanity watchdog feeding task started every 500 milliseconds.

NMI ISR installed and NMIs enabled.

Attaching network interface ei0... done.
Attaching network interface lo0... done.
Mounting NFS file systems from host bcars782b for target gpu5:
...done
Loading symbol table from bcars782b:/bnr/3rdparty/glan_sw/WindRiver/vw/config/gpu2960/vxWorks.sym ...done


 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
      ]]]]]]]]]]]  ]]]]     ]]]]]]]]]]       ]]              ]]]]         (R)
 ]     ]]]]]]]]]  ]]]]]]     ]]]]]]]]       ]]              ]]]]
 ]]     ]]]]]]]  ]]]]]]]]     ]]]]]] ]     ]]                ]]]]
 ]]]     ]]]]] ]    ]]]  ]     ]]]] ]]]   ]]]]]]]]]  ]]]] ]] ]]]]  ]]   ]]]]]
 ]]]]      ]]]  ]]    ] ]]]     ]] ]]]]] ]]]]]]   ]] ]]]]]]] ]]]] ]]   ]]]]
 ]]]]]     ]  ]]]]     ]]]]]      ]]]]]]]] ]]]]   ]] ]]]]    ]]]]]]]    ]]]]
 ]]]]]]      ]]]]]     ]]]]]]    ]  ]]]]]  ]]]]   ]] ]]]]    ]]]]]]]]    ]]]]
 ]]]]]]]    ]]]]]  ]    ]]]]]]  ]    ]]]   ]]]]   ]] ]]]]    ]]]] ]]]]    ]]]]
 ]]]]]]]]  ]]]]]  ]]]    ]]]]]]]      ]    ]]]]]]]  ]]]]    ]]]]  ]]]] ]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]       Development System
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]]
 ]]]]]]]]]]]]]]]]]]]]]]]]]]]       VxWorks version 5.1.1
 ]]]]]]]]]]]]]]]]]]]]]]]]]]       KERNEL: WIND version 2.4
 ]]]]]]]]]]]]]]]]]]]]]]]]]       Copyright Wind River Systems, Inc., 1984-1993

                           Board: GPU960.  Processor #0.
                          Memory Size: 0x400000.  BSP version 1.0 Beta-4.

->



// THE ABOVE IS THE VXWORKS PROMPT. IF YOU TYPE <help> IT WILL
// GIVE YOU SOME HELP. OK, IF YOU GET THIS FAR YOUR DOING PRETTY
// GOOD BUT THERE'S MORE.

// NOW RUN OUR STARTUP SCRIPT TO LOAD BNR SW AND RUN THE
// BNR INITIALIZATION AND DIAGNOSTICS SW BY TYPING THE
// FOLLOWING AT THE PROMPT.



-><YOUR BASE DIR/Release/startup.scr

#
# Master startup script
#       For release only.  Not for development use.
#
cd "YOUR BASE DIR/Release"
value = 0 = 0x0

# GPU related modules
#       Gary, (except xmemTest and gpuUIInterface by Thomas)
ld < GPU/gpu.o
value = 272625504 = 0x103fef60

# Port Card related modules
#       Ihab
ld < GPU/pmConf.o
value = 272384816 = 0x103c4330
ld < GPU/pmAll.o
value = 272383344 = 0x103c3d70 = _test_count + 0x8c

# SCM related modules and integration tests
```

```
#       Gary, (except generateCell, scuSCoreTest, scuPortCardTest,
#         scuDaisyChainTest by Mark)
ld < SCM/scm.o
value = 272613520 = 0x103fc090


sysInitCtl()



              System Initialization Controller

send my name gpu5 to server
My name echoed is gpu5
Connects to bcars782b thru local port 5131

GPU Diagnostics failed.

// YOU SHOULD BE ABLE TO SEE SOME INFO ON PORT CARD INITIALIZATION
// AND DIAGNOSTICS AROUND HERE AS WELL. IT DOES NOT APPEAR HERE
// BECAUSE THERE WAS NO PORT CARD ON THE WHEN THIS IS GENERATED

SCU interrupt initialization completed.

Switch interrupt monitoring started.

Performing PM Diagnostics/Initialization.

 Error while Access Slot Number : 1
PM Diagnostics/Initialization failed.

SCU Initialization Complete.

value = 0 = 0x0

->  // NOW LETS CHANGE THE BOOT PARAMETERS PERMANENTLY IN THE GPU's FLASH CARD ID.
-> flashBootParamsSet

'.' = clear field;  '-' = go to previous field;  ^D = quit

boot device          : ei
processor number     : 0
host name            : bcars782b                                   // YOUR HOST NAME
file name            : /bnr/3rdparty/glan_sw/WindRiver/vw/config/gpu2960/vxWorks  // BASE DIR/Release/GPU/vxWorks[1]
inet on ethernet (e) : 47.53.32.8:ffff0000                         // THE IP ADDRESS YOU SUPPLY FOR GPU
inet on backplane (b):
host inet (h)        : 47.53.32.1                                  // THE IP ADDRESS OF HOST WORKSTATION
gateway inet (g)     :
user (u)             : nobody                                      // THE USER ID YOU USE TO DOWNLOAD SW
ftp password (pw) (blank = use rsh): nobody                        // PASSWORD CORRESPONDING TO ABOVE
flags (f)            : 0x8
target name (tn)     : gpu5                                        // THE HOST NAME OF YOUR GPU
startup script (s)   :
other (o)            :


Card ID EEPROM updated with the following new boot parameters.

boot device          : ei
processor number     : 0
host name            : YOUR HOST NAME
file name            : YOUR BASE DIR/Rease/GPU/vxWorks
inet on ethernet (e) : THE IP ADDRESS YOU SUPPLY FOR GPU
host inet (h)        : THE IP ADDRESS OF YOUR HOST WORKSTATION
user (u)             : THE USER ID YOU USE TO DOWNLOAD SW
ftp password (pw)    : PASSWORD CORRESPONDING TO ABOVE
flags (f)            : 0x8
target name (tn)     : THE HOST NAME OF YOUR GPU

value = 0 = 0x0
->
```

1. The absolute path of the vxWorks image file.

*****************WE'RE DONE!!!***

For more information regarding the standard VxWorks booting process, boot ROM commands, boot parameters or configuring a host workstation to support a VxWorks target please see the "VxWorks Programmer's Guide", Chapter 2.

4.0 Startup Scripts

The same scripts still exist but have been modified for this release.The script is named 'startup.scr'. It is intended to be the script that performs all of BNR's SW downloading and initialization.The other script which is in the release directory, 'loadup.scr' performs the same duties as 'startup.scr' plus some additional initialization duties for switch debugging or demoing. People using this should be warned that correctness of this script is only guaranteed up to and including the call to sysInitCtl(). We are not too familiar with anything beyond that point and are unsure that it is still working correctly. It does not have any problems finding or loading SW but there could still be some differences that we do not understand.

WARNING: Depending on when you received the BNR portion of the SW, the script 'loadup.scr' may or may not download the BNR SW in the correct order. If it does not load these modules successfully, it should be edited to load the BNR modules in the same order as that seen in 'startup.scr'.

Here is the startup script that is shipped along with the Release SW, note that the following path has to be changed:

```
#
# Master startup script
#	For release only.  Not for development use.
#
# the following line contains the base directory where the switch software are installed,
# change it to the absolute path of your base directory *********************************
cd "/bnr/3rdparty/glan_sw/src/Release"

# GPU related modules
#	Gary, (except xmemTest and gpuUIInterface by Thomas)
ld < GPU/gpu.o

# Switch Configuration
# Ihab and Gary
ld < switchConf.o

# Port Card related modules
#	Ihab
ld < GPU/pmAll.o

# SCM related modules and integration tests
#	Gary, (except generateCell, scuSCoreTest, scuPortCardTest,
#	  scuDaisyChainTest by Mark)
ld < SCM/scm.o


sysInitCtl()
```

Additional object files and function calls for initialization can be added freely to the startup script file, providing the above object files are loaded first. The switch will not function properly without these object files.

5.0 Man Pages Usage

The man pages for BNR developed SW are very similar to those found on-line for VxWorks and in their reference manual. They were generated automatically using a couple of scripts from WindRiver using only the comments in the corresponding source code. On-line man pages for the BNR SW exist for the modules in gpu.o, scm.o, gpuVW.o, pmAll.o and gpuVW_res.o. For a list of these modules, see Section 2.0 on page 4, “New Release Contents”.

The man pages can be found in: YOUR_BASE_DIR/Release/man

It is recommend to either add the above path to your man path or add the following alias to your .cshrc (or wherever):

alias cnsman   man -M YOUR_BASE_DIR/Release/man

If you want to view the man pages for any of the modules just enter:

cnsman [1] moduleName   eg. cnsman scuDrv

This will list all the functions in the module (which would be useful if you can’t remember a function name or the module it is in) as well as how to use them and a brief overview of the entire module.

If you want to view the man pages for a particular function enter:

cnsman [2] functionName   eg. cnsman scuCellXferToCore

6.0 Upgrade

6.1 S/W Upgrade

Future S/W releases will be in the form as gzip’ed tarred files. Small patches may be available as well. Note that any minor upgrade will require a temporary shutdown of the switch due to the fact that vxWorks dynamically links objects and upgrading object modules during operation will very likely crash the system.

6.2 GPU RAM - Upgrading/Changing Size

It may be desirable to change the amount of RAM on the GPU. This section describes how to determine the amount of GPU RAM installed, change the amount of GPU RAM and re-configure VxWorks to make use of a different amount of RAM.

The GPU is designed to use either 4 1M DRAM SIMMs or 4 4M DRAM SIMMs for a total of 4 and 16M respectively. The SIMMs should be 30-pin with 9 data bits.

6.2.1 Determining Amount of GPU RAM Installed

The amount of GPU RAM installed can be determined by examining the part numbers on the SIMMs. The 1M parts have part numbers ending in ‘1000BJ’ while the 4M parts have part numbers ending in ‘4000BJ’. If it is necessary to acquire a set of 4M parts, one choice would be Samsung’s KM41C4000BJ-7 or KM41C4000BJ-6 (70ns or 60ns parts will do). Other suppliers will also do.

Please note that although the GPU has 8 DRAM SIMM slots, it has only been debugged to use the first 4 slots.

6.2.2 Changing DRAM SIMMs

In order to remove an existing DRAM SIMM, two tabs, visible through circular holes at either end of the SIMM, must be depressed before the SIMM can slid out of its slot.

6.2.3 Reconfiguring VxWorks for Different RAM Sizes

VxWorks learns of the amount of RAM available to it on a GPU by reading a location in the GPU’s Card ID Flash which indicates the amount of RAM available to VxWorks. In order to change this amount, function *flashSetRamSize*() should be used. For more information on *flashSetRamSize*(), please see the man page either on-line or later in this document. For example, in order to change the RAM size to 8M, at the VxWorks prompt, simply type flashSetRamSize(0x800000) and wait until the VxWorks prompt returns (it may take several seconds) then reboot the GPU so that VxWorks can make use of the additional RAM.

WARNING: BNR SW only supports use of up to 8M of GPU RAM. This is in order to make use of a faster i960 assembly call instruction which uses relative addressing which reaches only 8M. Do not configure a GPU to use more than 8M of RAM, even if 16M is installed, unless otherwise notified. At the current time neither Harvard University nor BNR can foresee the need to use more than 8M of RAM on the GPUs. Unfortunately, 4 2M SIMMs can not be purchased since the 2M SIMM parts do no exist.

7.0 Support

Please contact Ihab Mahna[ihab@bnr.ca] (for PM related stuff), Thomas NG [ngt@bnr.ca] (UI, TLG) or Gary Depelteau [delpeltea@bnr.ca] (SCU, GPU) if you have any questions or problems.

8.0 List of Functions

The following is a list of functions in the Creditnet Switch S/W (not in alphabetical order):

8.1 GPU

These functions can be found in gpu.o and gpuVW.o

8.1.1 Flash Memory Functions

fDramTest()	-	Perform a Fast memory test
flashAddRamSize()	-	Flash Add RAM Size.
flashBootParamsSet()	-	Flash Boot Parameter Set.
flashCardIdUsage()	-	Flash Card ID Usage.
flashCheckErase()	-	Check flash bank was erased.
flashCopyFromRam()	-	Copy nbytes from RAM to a flash bank.

flashDataPoll()	-	Polls a flash location.
flashDLoad()	-	Download a file from a workstation.
flashErase()	-	Erase a flash bank.
flashProgAltBank()	-	Program alternate flash bank.
flashSetRamSize()	-	Flash Set RAM Size.

8.1.2 GPU Testing Functions

gpuCardIdTest()	-	GPU Card ID Test
gpuCntrlRegTest()	-	GPU Control Register Test
gpuDiagCtl()	-	GPU Diagnostics Controller
gpuDipSwTest()	-	GPU DIP Switch Test
gpuEDramTest()	-	GPU Extended DRAM Test
gpuEepromTest()	-	GPU EEPROM Test
gpuErrRegTest()	-	GPU NMI Error Source Register Test
gpuEtherTest()	-	Test the i82596ca ethernet controller
gpuExtBusTest()	-	GPU External Bus Test
gpuFlSwapTest()	-	GPU Flash Swap Test
gpuSccTest()	-	Test the 85C30 SCC on the GPU
gpuTimIntTest()	-	GPU Real Time Clock Test

8.1.3 NMI and General Interrupts Functions

gpuNmiInit()	-	GPU NMI Initialization
gpuNmiInitIsr()	-	GPU NMI Initialization ISR
gpuNmiMon()	-	GPU NMI Monitor
gpuNmiMonStop()	-	Stop NMI counter monitoring
gpuNmiMonStart()	-	Start NMI counter monitoring
gpuNmiMonTask()	-	The GPU NMI Monitoring Task
intCntrMon()	-	Interrupt Counters Monitor
intMonInit()	-	Initialize Interrupt monitoring
intMonPerSet()	-	Set the Interrupt Monitor Period (seconds)

8.1.4 Lattice Chip Programming Functions

ispEnValCalc()	-	Calculate the ISP Programming Register Value
ispProgram()	-	Programs an ISP Lattice Device
ispRead()	-	Reads an ISP Lattice Device
ispVerify()	-	Verifies an ISP Lattice Device

8.1.5 Serial Port Functions

sccInit()	-	SCC initialization.
sccOutputChar()	-	Output a single character via the SCC (Serial Communication).
sccPrintHexWord()	-	Output a 32-bit hexadecimal number.
sccPrintString()	-	Output a string via the SCC.
sccWait()	-	Poll until SCC is no longer transmitting a char.

8.1.6 Watchdog Functions

wdFeeder()	-	Watchdog feeder function
wdFeederHookAdd()	-	Add hook routine to WD feeder task
wdFeederHookDelete()	-	Delete WD feeder task hook routine
wdFeederHookShow()	-	Displays WD hook routines
wdFeederInit()	-	Initialize watchdog feeder library
wdFeederTask()	-	Watchdog feeder task
wdRestartFeeder()	-	Re-enable the feeding of the watchdog
wdStopFeeder()	-	Force the disabling of watchdog feeds

8.2 SCM

These functions can be found in scm.o:

8.2.1 Miscellaneous functions

scmMicrosecondWait()	-	Wait n microseconds
scuACAlarmISR()	-	SCU AC Alarm ISR
scuCbtErrISR()	-	SCU Cell Bus-tap Error ISR
scuCoreClkAlarmISR()	-	SCU Core Memory Clock Alarm ISR
scuCoreParErrISR()	-	SCU Core Memory Parity Error ISR
scuFanAlarmISR()	-	SCU Fan Alarm ISR
scuIntInit()	-	SCU Interrupt Initialization
scuIntMon()	-	SCU Interrupt Monitor
scuItTimCompISR()	-	SCU Interval Timer Timing Complete ISR
scuPSAlarmISR()	-	SCU Power Supply Alarm ISR
scuSDiagCtl()	-	SCU Start-up Diagnostics Controller
scuSRegCheck()	-	SCU Register Bit Check
scuStaleCadrISR()	-	SCU Stale CADR ISR
scuVerify()	-	SCU Verify

8.2.2 Cell Manipulation and Share buffer functions

crc16()	-	CRC16 Calculation
cellCrc8()	-	8 bit CRC calculation
generateCell()	-	Generate Cell
printCellStruct()	-	Print Cell Structure
scu64BCellDecode()	-	SCU 64-Byte Cell Decode to 56-Bytes
scuCellCpyRamToSb()	-	Copy Cell GPU RAM->Shared Buffer
scuCellCpySbToRam()	-	Copy Cell Shared Buffer->GPU RAM
scuCellXferFromCore()	-	SCU Cell Transfer from Core Memory
scuCellXferToCore()	-	SCU Cell Transfer to Core Memory
scuCellXferToFromCore()	-	SCU Cell Transfer to/from Core
scuCoreCellRead()	-	SCU Core Cell Read
scuCoreCellWrite()	-	SCU Core Cell Write
scuGetCellsFromCore()	-	SCU Get Cells From Core
scuGetCellFromCore()	-	SCU Get Cell From Core
scuMap8To7()	-	SCU Map 8 TO 7 (byte)
scuPutCellInCore()	-	SCU Put Cell In Core
scuReadXCells()	-	Read X Cells from the Core Memory
scuSMSharBuffTest()	-	SCU SM Diag. Shared Buffer Test
scuSSBByteCheck()	-	SCU Shared Buffer Byte Check
scuVerCorePat()	-	SCU Verify Core Pattern
scuWriteXCells()	-	Write X Cells into the Core Memory
swapCell()	-	Swap Cell

8.2.3 CIF Functions

scuCifCntlRegRead()	-	SCU Read from CIF Control Register
scuCifCntlRegWrite()	-	SCU Write to CIF Control Register
scuCifInit()	-	SCU Cell Interface Initialization
scuCifHPWFifoStatRead()	-	SCU CIF HP W_IVC FIFO Status Read
scuCifLPWFifoStatRead()	-	SCU CIF LP W_IVC FIFO Status Read
scuCifRdPoll()	-	SCU CIF Read Cycle Poll
scuCifRdWrPoll()	-	SCU CIF Read/Write Cycle Poll
scuCifRFifoStatRead()	-	SCU CIF R_IVC FIFO Status Read
scuCifRivcFifoRead()	-	SCU CIF R_IVC FIFO Read
scuCifStatRegRead()	-	SCU Read from CIF Status Register
scuCifWrPoll()	-	SCU CIF Write Cycle Poll
scuCifWivcFifoRead()	-	SCU CIF W_IVC FIFO Read
scuSCifDataPathTest()	-	SCU CIF Data Path Test
scuSCifFifoTest()	-	SCU CIF FIFO Test

scuSFifoStatCheck()	-	SCU CIF FIFO Status Check
scuCifRcvCrc8ErrISR()	-	SCU CIF Receive CRC8 Error ISR
scuSCifCrc8Test()	-	SCU CIF CRC8 Test
scuSMCifRegTest()	-	SCU SM CIF Register Test

8.2.3.1 IVC Monitor functions

scuRFifoAlFullISR()	-	SCU R_IVC FIFO Almost Full ISR
scuRFifoMidISR()	-	SCU R_IVC FIFO Mid-range ISR
scuRivcLutInit()	-	SCU R_IVC Lookup Table Initialization
scuRivcLutSet()	-	SCU R_IVC Lookup Table Entry Set
scuRivcLutSetup()	-	SCU R_IVC Lookup Table Setup
scuSMWivcLutTest()	-	SCU SM Diag. W_IVC Lookup Table Test
scuSMRivcLutTest()	-	SCU SM Diag. R_IVC Lookup Table Test
scuWFifoHPAlFullISR()	-	SCU HP W_IVC FIFO Almost Full ISR
scuWFifoHPMidISR()	-	SCU HP W_IVC FIFO Mid-range ISR
scuWFifoLPAlFullISR()	-	SCU LP W_IVC FIFO Almost Full ISR
scuWFifoLPMidISR()	-	SCU LP W_IVC FIFO Mid-range ISR
scuWivcLutInit()	-	SCU W_IVC Lookup Table Initialization
scuWivcLutSet()	-	SCU W_IVC Lookup Table Entry Set
scuWivcLutSetup()	-	SCU W_IVC Lookup Table Setup

8.2.4 CDM functions

scuCdmCntlRegRead()	-	SCU Read from CDM Control Register
scuCdmCntlRegWrite()	-	SCU Write to CDM Control Register
scuCdmStaleCadrRegRead()	-	SCU CDM Stale CADR Reg. Read
scuCdmStatRegRead()	-	SCU Read from CDM Status Register
scuChalkSearchDisable()	-	SCU CDM Chalkmark Search Disable
scuChalkSearchEnable()	-	SCU CDM Chalkmark Search Enable
scuChalkSearchReset()	-	SCU CDM Chalkmark Search Reset
scuCurChalkRead()	-	SCU Current Chalkmark Read
scuCurChalkSet()	-	SCU Current Chalkmarks Set
scuIvcSetup()	-	SCU IVC Setup
scuIvcSet()	-	SCU Single IVC Setup
scuIvcInit()	-	SCU IVC Initialization
scuSCdmCmSrchTest()	-	SCU CDM Chalkmark Search Test
scuSCdmCmUpdTest()	-	SCU CDM Chalkmark Updating Test
scuSCdmMonFATest()	-	SCU CDM Monitoring and FA Test
scuSChalkmarkCheck()	-	SCU Chalkmark Check
scuSMCdmConCapTest()	-	SCU SM CDM Configuration/Capture Test
scuSMCdmRegTest()	-	SCU SM CDM Register Test

8.2.5 CTC functions

scuCtcInit()	-	SCU Cell Time Counter Initialization
scuCtcIntTimProg()	-	SCU CTC Interval Timer Program
scuCtcIntTimRead()	-	SCU CTC Interval Timer Register Read
scuCtcRealTimProg()	-	SCU CTC Real-time Timer Program
scuCtcStartRegRead()	-	SCU CTC Start Register Read
scuCtcStatRegRead()	-	SCU CTC Status Register Read
scuCtcStopRegRead()	-	SCU CTC Stop Register Read
scuSCtcIt1Test()	-	SCU CTC Interval Timer Test 1
scuSCtcIt2Test()	-	SCU CTC Interval Timer Test 2
scuSMCtcRegTest()	-	SCU SM CTC Register Test

8.2.6 DBM functions

scuDbmCapture()	-	SCU Distribution Bitmap Capture
scuDbmInit()	-	SCU Distribution Bitmap Initialization
scuDbmSet()	-	SCU Distribution Bitmap Entry Set
scuDbmSetup()	-	SCU Distribution Bitmap Setup

scuSMDbmRamTest()	-	SCU SM DBM RAM Test
scuSMWDbmRamTest()	-	SCU SM Working DBM RAM Test
scuWDbmCapture()	-	SCU Working Distribution Bitmap Capture
scuWDbmIntegTest()	-	SCU wDBM Integration Test
scuWDbmMultiCastTest()	-	SCU wDBM Multi-Cast Test
scuWDbmWalkingOneTest()	-	SCU wDBM Walking One Test
scuWDbmWalkingZeroTest()	-	SCU wDBM Walking Zero Test

8.2.7 FLM Functions

fakeFLMInit()	-	Initialize FLM emulator
fakeFLMGetNextCADR()	-	get the next address out of the FLM emulator
scuFlmCmbfCountRead()	-	SCU FLM CMBF Counter Read
scuFlmCntlRegRead()	-	SCU Read from FLM Control Register
scuFlmCntlRegWrite()	-	SCU Write to FLM Control Register
scuFlmEmptyISR()	-	SCU FLM Free List Empty ISR
scuFlmFailISR()	-	SCU FLM Free List Fail ISR
scuFlmFLArrayPush()	-	SCU FLM Free List Array Push
scuFlmFLCInit()	-	SCU FLM Free List Complete Initialization
scuFlmFLPush()	-	SCU FLM Free List Push
scuFlmFreeListInit()	-	SCU FLM Free List Initialization
scuFlmFullISR()	-	SCU FLM Full ISR
scuFlmHalt()	-	SCU FLM Halt
scuFlmHaltClear()	-	SCU FLM Halt Clear
scuFlmInit()	-	SCU FLM Initialization
scuFlmPrime()	-	SCU FLM Prime
scuFlmReset()	-	SCU FLM Reset
scuFlmStatRegRead()	-	SCU Read from FLM Status Register
scuFlmWrapISR()	-	SCU FLM Free List Wrap ISR
scuSFlmBasicTest()	-	SCU Flm Basic Functionality Test
scuSMFlmRamStatTest()	-	SCU SM Diag. FLM RAM and Status Test
scuSMFlmRegTest()	-	SCU SM FLM Register Test
scuSFlmIntegTest()	-	SCU FLM Integrity Test

8.2.8 Miscellaneous Initialization and Configuration

scuCdmInit()	-	SCU Cell Distribution Manager Initialization
scuDefDSInit()	-	SCU Default Data Structure Initialization
scuHwInit()	-	SCU Hw Initialization Initialization
scuInitCtl()	-	SCU Initialization Control
scuPifInit()	-	SCU Processor Interface Initialization
sysInitCtl()	-	System Initialization Controller

8.2.9 Miscellaneous SCM/SCU Tests

scmMarch6Test()	-	SCM March 6 Memory Test
scmRegTest()	-	SCM Register Test
scmWalkA1Test()	-	SCM Address-bit Walking One Test
scmWalkD0Test()	-	SCM Data-bit Walking Zero Test
scmWalkD1Test()	-	SCM Data-bit Walking One Test
scmChkDramParity()	-	SCM Check for GPU DRAM Parity Error
scmChkFlashSig()	-	SCM Check Flash Signature
scuCMCompareCell()	-	SCU CM Compare Cell
scuCMTestEntireCore()	-	SCU CM Test Entire Core
scuCMTestGeneratePatterns()	-	SCU CM Test Generate Patterns
scuCMTestLostAddresses()	-	SCU CM Test Lost Addresses
scuCMTestOneAddress()	-	SCU CM Test One Address
scuCMWalkingTest()	-	SCU CM Walking Test
scuDaisyChainTest()	-	SCU Daisy Chain Test
scuFifoIntegTest()	-	SCU FIFO Integration Test
scuFifoSeqTest()	-	SCU FIFO Sequential Test

scuHVCdmPETest()	-	SCU HV CDM PE Test (N.T.)
scuHVCifPETest()	-	SCU HV CIF PE Test (N.T.)
scuHVCrmBFStatTest()	-	SCU HV CRM BF Status Test (N.T.)
scuHVCrmPETest()	-	SCU HV CRM PE Test (N.T.)
scuHVCtcItPETest()	-	SCU HV CTC IT PE Test (N.T.)
scuHVDS1TogCheck()	-	SCU HV DIP Switch Toggle Check
scuHVEncDecTest()	-	SCU HV Encoded/Decoder Test
scuHVFlmCmbfPETest()	-	SCU HV FLM CMBF Count PE Test (N.T.)
scuHVIdleTest()	-	SCU HV Idle Test (N.T.)
scuHVIdLogTest()	-	SCU HV ID/LOG Test
scuHVPMIntTest()	-	SCU HV PM Interrupt Test (N.T.)
scuHVPMResetTest()	-	SCU HV PM Reset Test
scuHVWCadrPETest()	-	SCU HV W_CADR PE Test (N.T.)
scuInitPortCardTest()	-	SCU Initialization Port Card Test
scuSCoreTest()	-	SCU Start-up CM Test
scuSMPbusRegTest()	-	SCU SM P-Bus Register Test

8.3 PM

These functions can be found in pmAll.o:

8.3.1 General PCC functions

checkPortCardExist()	-	Checks if a port card is present in the specified slot
displayChecksums()	-	display portcard checksums
repeatModeOff()	-	turn PCC repeat mode off
repeatModeOn()	-	turn PCC repeat mode on
runPccService()	-	Run a PCC service
setPort()	-	Sets the current port - Required for many PCC functions
ur()	-	Update ROM

8.3.2 Memory Access

downloadToPcc()	-	Download to PCC
dp()	-	Dump PCC memory to the screen with specified address, width, and length
fillPcc()	-	Fill pcc memory.
rp()	-	Read from a PCC memory location
wp()	-	Write to a PCC memory location

8.3.3 PCC test

pccAddressTest()	-	PCC Address Test
pccMarch6Test()	-	PCC March 6 Memory Test
pccWalkD1Test()	-	PCC Data-bit Walking One Test
pccWalkD0Test()	-	PCC Data-bit Walking Zero Test
runPccMemoryTest()	-	Run a PCC memory test
testChecksums()	-	test portcard checksums

8.3.4 Scheduler specific

rs()	-	Read Scheduler
ws()	-	Write Scheduler
resetScheduler()	-	Reset Scheduler

9.0 Memory maps

Individual memory maps for SCM and PM can be found in the following files:

SCM - scu.h, scu.h.ramAdrs

PCM - PCCregions.h, SCHregions.h, mcpPcMemoryMap.h, pcMemoryMap.h and schMemoryMap.h.

10.0 Glossary

ATM	:Asynchronous Transfer Mode
BNR	:Bell Northern Research
BTL	:Bus Transceiver Logic
CCB	:Cell Control Bus
CDM	:Cell Departure Monitor
CIF	:Cell Interface
CM	:Core Memory or Credit Manager on PCC
CRM	:Core Memory Resource Manager
DM	:Data Manager
FIFO	:First-In First-Out
FLM	:Free List Manager
GPU	:General Processor Unit
IVC	:Internal Virtual Connection
P-BUS	:Processor Bus
PCC	:port Card Controller
PM	:Port Module
SCC	:Serial Communication Controller
SCH	:Scheduler on PCM
SCM	:Switch Control Module (consists of a GPU and an SCU)
SCU	:Switch Control Unit
TBD	:To Be Determined
TC	:Terminator Card
TLG	:Test Load Generator

NAME

displayChecksums - display portcard checksums.

SYNOPSIS

STATUS displayChecksums ( )

DESCRIPTION

This function scans all existing portcards for their PM rom checksums and displays them to the VxWorks shell. This function is called from within the function 'testChecksums()' if the checksums don't all agree, but it can also be called on its own from the VxWorks shell by itself.

PARAMETERS

WARNINGS

SEE ALSO

testChecksums(), extractTextChecksum().

NAME

downloadToPcc() - Download to PCC

SYNOPSIS

```
STATUS downloadToPcc
        (
        UINT32 port_num,
        UCHAR  *fname,
        UINT32 addr,
        UINT32 width
        )
```

DESCRIPTION

This function downloads a block of memory in the GPU to the PCC memory.

PARAMETERS

UINT32 port_num - PM whose PCC to which GPU memory is downloading.
UCHAR *fname   - pointer to beginning of memory block in PCC.
UINT32 addr   - address in PCC where GPU memory is downloaded.
UINT32 width   - width to access--WORD(4), SHORT(2), BYTE(1).

SEE ALSO

dp(2) **CREDITNET SWITCH SW MANUAL** dp(2)

NAME

dp() - Dump Memory

SYNOPSIS

STATUS dp (int add, int width, int length)

DESCRIPTION

This function dumps a block of memory from the PCC. The size of the block is determined by the parameter length. The dumped memory is organized according to width specified by the parameter width--long word (32 bits), short word (16 bits) and byte (8 bits). The PM is assumed to be specified using setPort() prior to calling this function.

PARAMETERS

int add - PCC address from which value is written.
int width - width to access--WORD(4), SHORT(2), BYTE(1).
int length - length of block to dump.

SEE ALSO

fillPcc(2) CREDITNET SWITCH SW MANUAL fillPcc(2)

NAME

fillPcc - Fill pcc memory.

SYNOPSIS

STATUS fillPcc (int start, int pattern, int width, int count)

DESCRIPTION

This function fills a range of the PCC memory with a specified pattern of a specified width. The range is defined by the starting address and the number of times to repeat the pattern. The main loop is executed entirely within the PCC of the portcard using pccServices in the function pmFill(). The PM is assumed to be specified using setPort() prior to calling this function..

PARAMETERS

int start - starting address for fill
int pattern - pattern to fill with
int width - width of pattern in bytes (4 or 2 or 1)
int count - number of times pattern is repeated

WARNINGS

SEE ALSO

NAME

pccAddressTest - PCC Address Test.

SYNOPSIS

STATUS pccAddressTest (void *start, UINT32 length, UINT32 width, UINT32 mask)

DESCRIPTION

This function tests memory by writing the lower bits of the address of a memory location into itself. The memory width may be set to 1, 2, or 4 bytes, a mask may be specified to avoid testing all the bits. The method used is similar to one used int scuWIVCLutTest().

This function sets up and calls the function pmAddressTest() which is executed on the PCC. The test results are then passed back to the GPU where this function relays the output

RETURNS

OK, or ERROR if an fault is found. Further info is passed back through the dual port memory from the PCC and printed to the shell.

PARAMETERS

void *start - test start position
UINT32 length - how many UINTXX's to test
UINT32 width - Width of memory in bytes
UINT32 mask -bit mask for which bits to test

WARNINGS

This test obliterates the contents of the memory or register it tests and caution should be exercised to ensure that the memory region under test does not contain important or needed data.

SEE ALSO

scuWIVCLutTest().

NAME

pccMarch6Test - PCC March 6 Memory Test.

SYNOPSIS

```
STATUS pccMarch6Test
        ( UINT32 start, UINT32 size, UINT32 width, UINT32 foreground, BOOL chkDramPE )
```

DESCRIPTION

This function is a conversion of the SCM function of the same name. This function performs the March 6 algorithm on a section of memory. This test checks for stuck address and data bits, leakages and decoding problems. If being performed during GPU startup diagnostics, this test must feed the watchdog timer at intervals less than 2 seconds or else a watchdog NMI will occur. This is a fast test which uses fixed pattern and is suitable for large memories. It will catch a high percentage of memory faults but it does not provide as much coverage as qMemTest() which performs multiple passes with random patterns.

This function sets up and calls the function pmMarch6Test() which is executed on the PCC. The test results are then passed back to the GPU where this function relays the output.

RETURNS

OK, or ERROR if an fault is found.

PARAMETERS

UINT32 start - Start address of memory to test
UINT32 size - Size of memory in bytes
UINT32 width - Width of memory in bytes
UINT32 foreground - Foreground pattern
BOOL chkDramPE - 1=Check DRAM par. err., 0=don’t (un-used)

WARNINGS

This test obliterates the contents of the memory or register it tests and caution should be exercised to ensure that the memory region under test does not contain important or needed data.

SEE ALSO

scmMarch6Test().

NAME

pccWalkD1Test - PCC Data-bit Walking Zero Test.

SYNOPSIS

STATUS pccWalkD0Test
(void *testLoc, UCHAR lsb, UCHAR msb, UINT32 mask, UCHAR memWidth, UCHAR flag)

DESCRIPTION

This function is a conversion of the SCM function of the same name. This function walks a zero across a register or memory location. It is highly configurable in that the user can specify the range of bits covered during the walk, a mask which causes certain bits to be skipped and the width of the memory region in which the register or location exists in order to control the type of read and write cycle that gets used. A flag is also passed to this function to instruct it to perform the test once with error checking or repeatedly without error checking.

This function sets up and calls the function pmWalkD0Test() which is executed on the PCC. The test results are then passed back to the GPU where this function relays the output.

RETURNS

OK, or ERROR if any pattern read from the register is not the same as that written there. It does not return or perform error checking if flag is set to CONTINUOUS (0).

PARAMETERS

void *testLoc - Location to test
UCHAR lsb - Least significant bit to start
UCHAR msb - Most significant bit to stop
UCHAR memWidth - Width of testLoc region (bytes)
UCHAR flag - 0 = CONTINUOUS, 1 = ONCE

WARNINGS

This test obliterates the contents of the memory or register it tests and caution should be exercised to ensure that the memory region under test does not contain important or needed data.

SEE ALSO

scmWalkD1Test().

NAME

pccWalkD1Test - PCC Data-bit Walking One Test.

SYNOPSIS

```
STATUS pccWalkD1Test
      ( void *testLoc, UCHAR lsb, UCHAR msb, UINT32 mask, UCHAR memWidth, UCHAR flag )
```

DESCRIPTION

This function is a conversion of the SCM function of the same name. This function walks a one across a register or memory location. It is highly configurable in that the user can specify the range of bits covered during the walk, a mask which causes certain bits to be skipped and the width of the memory region in which the register or location exists in order to control the type of read and write cycle that gets used. A flag is also passed to this function to instruct it to perform the test once with error checking or repeatedly without error checking.

This function sets up and calls the function pmWalkD1Test() which is executed on the PCC. The test results are then passed back to the GPU where this function relays the output.

RETURNS

OK, or ERROR if any pattern read from the register is not the same as that written there. It does not return or perform error checking if flag is set to CONTINUOUS (0).

PARAMETERS

void *testLoc - Location to test
UCHAR lsb - Least significant bit to start
UCHAR msb - Most significant bit to stop
UCHAR memWidth - Width of testLoc region (bytes)
UCHAR flag - 0 = CONTINUOUS, 1 = ONCE

WARNINGS

This test obliterates the contents of the memory or register it tests and caution should be exercised to ensure that the memory region under test does not contain important or needed data.

SEE ALSO

scmWalkD1Test().

NAME

repeatModeOff - turn PCC repeat mode off.

SYNOPSIS

STATUS repeatModeOff ( )

DESCRIPTION

This function sets the REPEAT_MODE flag in the dual port memory to FALSE. The PCC uses this flag to decide if it should return from executing a function, or if it should repeat that function again. To enter repeat mode, type 'repeatModeOn' and the next command you type will be repeated (but only the PCC part of it). Functions wich may be repeated are PC services (see pcServices.h) and simple reads and writes. To cancel the repeatModeOn() command, you may use repeatModeOff, but this will only work if it is the VERY NEXT COMMAND entered after repeatModeOn(). The PM is assumed to be specified using setPort() prior to calling this function.

RETURNS

OK, or ERROR if the portcard isn't set properly.

PARAMETERS

WARNINGS

SEE ALSO

repeatModeOn().

NAME

repeatModeOn - turn PCC repeat mode on.

SYNOPSIS

STATUS repeatModeOn ( )

DESCRIPTION

This function sets the REPEAT_MODE flag in the dual port memory to TRUE. The PCC uses this flag to decide if it should return from executing a function, or if it should repeat that function again. To enter repeat mode, type 'repeatModeOn' and the next command you type will be repeated (but only the PCC part of it). Functions wich may be repeated are PC services (see pcServices.h) and simple reads and writes. To cancel the repeatModeOn() command, you may use repeatModeOff, but this will only work if it is the VERY NEXT COMMAND entered after repeatModeOn(). The PM is assumed to be specified using setPort() prior to calling this function.

RETURNS

OK, or ERROR if the portcard doesn't exist.

PARAMETERS

WARNINGS

SEE ALSO

repeatModeOff().

NAME

resetPcc() - Reset PCC

SYNOPSIS

STATUS resetPcc (int port_num)

DESCRIPTION

This function resets the PCC of the PM specified by the parameter port_num. It performs PM diagnostics and initialization, which include testing the GPU-PCC dual-port memory by the GPU, PCC static, PCC-GPU dual-port, PCC-SCH dual-port, HWIO using loopback, all PM memories and all PM registers; configuring the H/W FLEX; and initalizing PM memories and registers.

PARAMETER

int port_num - PM whose PCC is to be reset.

WARNINGS

INCLUDE FILES

SEE ALSO

NAME

resetScheduler() - Reset Scheduler

SYNOPSIS

STATUS resetScheduler (UINT32)

DESCRIPTION

This function resets the Scheduler of the PM specified by the parameter port_num. It downloads the scheduler software and reboots the scheduler.

PARAMETER

UINT32 port_num - PM whose Scheduler is to be reset.

SEE ALSO

resetPcc()

rp(2) **CREDITNET SWITCH SW MANUAL** **rp(2)**

NAME

rp() - Read PCC

SYNOPSIS

STATUS rp (int add, int width)

DESCRIPTION

This function reads the value in the address of the PCC specified by the parameter add. The value read is of width specified by the parameter width--long word (32 bits), short word (16 bits) and byte (8 bits). If the macro _USE_PRINTF is defined, the value will be printed. The PM is assumed to be specified using setPort() prior to calling this function.

PARAMETERS

int add - PCC address from which value is written.
int width - width to access--WORD(4), SHORT(2), BYTE(1).

WARNINGS

SEE ALSO

readPcc(), wp(), writePcc()

NAME

rs() - Read Scheduler

SYNOPSIS

STATUS rs (int add, int width)

DESCRIPTION

This function reads from Scheduler. The value read will be of width specified by the parameter width--long word (32 bits), short word (16 bits) and byte (8 bits). If the macro _USE_PRINTF is defined, the value will be printed. The PM is assumed to be specified using setPort() prior to calling this function.

PARAMETERS

int add - Scheduler address from which value is read.
int width - width to access--WORD(4), SHORT(2), BYTE(1).

WARNINGS

SEE ALSO

readPcc(), rp(), ws(), resetScheduler()

NAME

runPccService() - Run PCC Services.

SYNOPSIS

STATUS runPccService (UINT32 port_num, UINT32 service)

DESCRIPTION

This function calls a PCC service, which starts the execution of a routine on the PCC. The service functions are stored in the array ‘serviceArray’ (in pcServices.h).

PARAMETERS

UINT32 port_num - port number.
UINT32 service - service index.

WARNINGS

SEE ALSO

pcServices.h

NAME

setPort() - Set Port

SYNOPSIS

STATUS setPort (UINT32 port_num)

DESCRIPTION

This function sets the current slot to the PM specified by the parameter port_num. It verifies that the PM exists before setting the current slot to the new PM. If PM does not exist, ERROR will be returned.

PARAMETER

UINT32 port_num - PM which is to be set to the current slot.

WARNINGS

SEE ALSO

NAME

testChecksums - test portcard checksums.

SYNOPSIS

STATUS testChecksums ( )

DESCRIPTION

This function scans all existing portcards for their PM rom checksums to ensure that they all match the Release directory checksum. If they do not, it calls ‘displayChecksums()’ to output them to the VxWorks shell.

PARAMETERS

WARNINGS

SEE ALSO

displayChecksums(), extractTextChecksum().

ur(2) **CREDITNET SWITCH SW MANUAL** ur(2)

NAME

ur() - Update ROM

SYNOPSIS

STATUS ur (UINT32 port_num)

DESCRIPTION

This function updates the PCC ROM of the specified PM.

PARAMETER

UINT32 port_num - the PM whose ROM is to be updated.

WARNINGS

Error will be generated if vxWorks is not in a directory with bootRom.rom not in PM.rom subdirectory.

SEE ALSO

downloadToPcc()

wp(2) **CREDITNET SWITCH SW MANUAL** wp(2)

NAME

wp() - Write PCC

SYNOPSIS

STATUS wp (int add, int width)

DESCRIPTION

This function writes a value, specified by the parameter val, to address, specified by the parameter add, in the PCC. The width being written is specified by the parameter width--long word (32 bits), short word (16 bits), and byte (8 bits). The PM is assumed to be specified using setPort() prior to calling this function.

PARAMETERS

int add - PCC address to which value is written.
int val - value to be written to PCC memory.
int width - width to access--WORD(4), SHORT(2), BYTE(1).

WARNINGS

SEE ALSO

readPcc(), rp(), writePcc()

ws(2) **CREDITNET SWITCH SW MANUAL** ws(2)

NAME

ws() - Write Scheduler

SYNOPSIS

STATUS ws (int add, int val, int width)

DESCRIPTION

This function writes a value, specified by the parameter val, to address, specified by the parameter add, in the Scheduler. The width to be written is specified by the parameter width-- long word (32 bits), short word (16 bits) and byte (8 bits). The PM is assumed to be specified using setPort() prior to calling this function.

PARAMETERS

int add - Scheduler address to which value is written.
int val - value to be written to Scheduler memory.
int width - width to access--WORD(4), SHORT(2), BYTE(1).

WARNINGS

SEE ALSO

readPcc(), rp(), rs(), resetScheduler()

alias(1) **CREDITNET SWITCH SW MANUAL** **alias(1)**

NAME

alias - Aliases functions

SYNOPSIS

dw() - Dump Word.
ramDebug() - RAM Debug

```
void dw
   (void * adrs)
STATUS ramDebug
   (void * start, UINT32 size, UINT32 mask, UCHAR memWidth,
   BOOL forever)
```

DESCRIPTION

This module contains functions that alias or alter the default output and functionality of VxWorks and CreditNe Switch functions.

INCLUDE FILES

usrLib.h, xmemTest.h

SEE ALSO

NAME

fDramTest - Perform a fast DRAM tests

SYNOPSIS

fDramTest() - Perform a Fast memory test

```
fDramTest
   (void * start, UINT32 size)
```

DESCRIPTION

This module provides a function to perform a fast test on DRAM. It ensures that only i960 registers are used. It only performs a few test sequences since it is called at boot and it is imperative to keep execution time down. The test is performed with parity checking turned off since there is a known problem with the GPU parity error bits (which does not affect the DRAM sanity).

WARNINGS

This module can only use registers. The calling function should also deactivate the i960's data cache if the test function is being performed on a processor with a data cache.

INCLUDE FILES

gpu960.h

SEE ALSO

romInit(), "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics"

flashCtl(1) **CREDITNET SWITCH SW MANUAL** **flashCtl(1)**

NAME

flashCtl - Flash Control Library

SYNOPSIS

flashCopyFromRam() - Copy nbytes from RAM to a flash bank.
flashCheckErase() - Check flash bank was erased.
flashErase() - Erase a flash bank.
flashDataPoll() - Polls a flash location.
flashDLoad() - Download a file from a workstation.
flashProgAltBank() - Program alternate flash bank.
flashBootParamsSet() - Flash Boot Parameter Set.
flashCardIdUsage() - Flash Card ID Usage.
flashAddRamSize() - Flash Add RAM Size.
flashSetRamSize() - Flash Set RAM Size.

```
STATUS flashCopyFromRam
   (UCHAR * src, UCHAR * dest, UINT32 nbytes)
STATUS flashCheckErase
   (UCHAR * devStart)
STATUS flashErase
   (UCHAR * dest)
STATUS flashDataPoll
   (UCHAR * dest, UINT32 value)
UINT32 flashDLoad
   (char * file, void * buf)
STATUS flashProgAltBank
   (char * fileName)
STATUS flashBootParamsSet
   (void)
UINT32 flashCardIdUsage
   (void)
STATUS flashAddRamSize
   (UINT32 ramSize)
STATUS flashSetRamSize
   (UINT32 ramSize)
```

DESCRIPTION

This module contains the functions required to alter infor-mation in the GPU's bootrom or Card ID flash devices under software control. It assumes that the flash parts used on the GPUs are AMD or Intel 29F040s. The main functions to use are flashProgAltBank() to program a bootrom bank given a binary bootrom image on the workstation and flashBoot- ParamsSet() which prompts the user for changes to be made to the GPU's boot parameters stored in the Card ID flash.

WARNINGS

Functions in this module can severly corrupt the contents of the GPU's bootrom or Card ID flashes and should be used carefully.

INCLUDE FILES

gpu960.h, flashCtl.h, bootLib.h, scmUtil.h

SEE ALSO

flashProgAltBank(), flashBootParamsSet(), "CreditNet GPU Specification" generateCell(1) CREDITNET SWITCH SW MANUAL generateCell(1)

NAME

generateCell - Generate Cell

SYNOPSIS

generateCell() - Generate Cell

CELL_STRUCT * generateCell
(UINT16 ivc, char gfc, char vpi, UINT16 vci, char plt,
UINT32 pattern)

DESCRIPTION

This module contains one function generateCell(), which generates a cell based on the parameters passed to it.

INCLUDE FILES

scuDrv.h, generateCell.h

SEE ALSO

generateCell()

gpuDiag(1) **CREDITNET SWITCH SW MANUAL** gpuDiag(1)

NAME

gpuDiag - GPU Diagnostics Library

SYNOPSIS

gpuDiagCtl() - GPU Diagnostics Controller
gpuEepromTest() - GPU EEPROM Test
gpuCardIdTest() - GPU Card ID Test
gpuEDramTest() - GPU Extended DRAM Test
gpuTimIntTest() - GPU Real Time Clock Test
gpuEtherTest() - Test the i82596ca ethernet controller
gpuSccTest() - Test the 85C30 SCC on the GPU
gpuErrRegTest() - GPU NMI Error Source Register Test
gpuCntrlRegTest() - GPU Control Register Test
gpuFlSwapTest() - GPU Flash Swap Test
gpuExtBusTest() - GPU External Bus Test
gpuDipSwTest() - GPU DIP Switch Test

```
STATUS gpuDiagCtl
   (UINT32 testMask)
STATUS gpuEepromTest
   (void)
STATUS gpuCardIdTest
   (void)
M6STATUS gpuEDramTest
   (UINT32 start, UINT32 size, UINT32 width)
STATUS gpuTimIntTest
   (void)
STATUS gpuEtherTest
   (void)
STATUS gpuSccTest
   (void)
STATUS gpuErrRegTest
   (void)
STATUS gpuCntrlRegTest
   (void)
STATUS gpuFlSwapTest
   (void)
STATUS gpuExtBusTest
   (void)
STATUS gpuDipSwTest
   (void)
```

DESCRIPTION

This module contains diagnostics functions that test GPU HW. They are invoked during System Initialization when the GPU (and CreditNet Switch) boot. There general purpose is to ensure that the GPU is sane enough to download a VxWorks image from the switch workstation and begin running the switch application software. These tests can also be invoked from the VxWorks shell for debugging purposes. gpuDiagCtl() is the control SW for GPU Diagnostics and can be invoked passing it a test mask to invoke any subset of GPU tests excluding gpuEDramTest().

Some of the functions in this module compile differently depending on whether ROM_RESIDENT is defined. Usually this is done to disable output via 'printf' while the switch is booting. You will compile this module with ROM_RESIDENT defined if you make gpuDiag_res.o instead of gpuDiag.o. When you make this module as gpuDiag.o ROM_RESIDENT will not be defined and all output via 'printf' will be available.

WARNINGS

Users should be careful when invoking these tests from the VxWorks Shell because they could place the GPU in an unusable state. Tests that are particularily dangerous for this are gpuEDramTest() and gpuEtherTest().

INCLUDE FILES

vxWorks.h, gpu960.h, glanErrno.h, scmUtil.h, sysLib.h, gpuDiag.h, stdio.h, taskLib.h, intLib.h, tyLib.h, vxLib.h, logLib.h

SEE ALSO

romInit(), sysLib, bootConfig, usrRoot(), "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics"

gpuNmiLib(1) CREDITNET SWITCH SW MANUAL gpuNmiLib(1)

NAME

gpuNmiLib - GPU Startup Non-maskable Interrupt Library

SYNOPSIS

gpuNmiInit() - GPU NMI Initialization
gpuNmiMon() - GPU NMI Monitor
gpuNmiMonTask() - The GPU NMI Monitoring Task
gpuNmiMonStop() - Stop NMI counter monitoring
gpuNmiMonStart() - Start NMI counter monitoring
gpuNmiInitIsr() - GPU NMI Initialization ISR

```
STATUS gpuNmiInit
   (VOIDFUNCPTR nmiIsr)
void gpuNmiMon
   (void)
void gpuNmiMonTask
   (void)
STATUS gpuNmiMonStop
   (void)
STATUS gpuNmiMonStart
   (void)
void gpuNmiInitIsr
   (void)
```

DESCRIPTION

This module contains functions pertaining to the handling of NMIs during System Initialization. The approach to handling each NMI is that it will be categorically counted (there are counters for parity error, watchdog timer expiration and external bus timeout NMIs) and the counters will be printed out periodically (probably every 10 seconds) if they are non-zero. When the counters are non-zero, they will be cleared to zero when printed. The monitorring task which prints out the counters when they are non-zero is separate from the actual NMI Service Routine since this sort of IO should not be performed at interrupt level. It is also possible for debugging reasons to turn NMI monitorring off in which case an individual message per NMI is printed. This is not normally desirable.

WARNINGS

Please refer to the list of VxWorks routines callable from interrupt code in section 3.4 of the VxWorks Programmer's Guide 5.1. Generally, routines that could cause blocking or perform IO should not be called.

INCLUDE FILES

vxWorks.h, intLib.c, gpu960.h, glanErrno.h, sysLib.h,
stdio.h, taskLib.h

SEE ALSO

sysLib, “CreditNet GPU Specification”

NAME

gpuWdLib - GPU Sanity Watchdog Library

SYNOPSIS

wdFeeder() - Watchdog feeder function
wdFeederTask() - Watchdog feeder task
wdFeederInit() - Initialize watchdog feeder library
wdFeederHookAdd() - Add hook routine to WD feeder task
wdFeederHookDelete() - Delete WD feeder task hook routine
wdStopFeeder() - Force the disabling of watchdog feeds
wdRestartFeeder() - Re-enable the feeding of the watchdog
wdFeederHookShow() - Displays WD hook routines

```
void wdFeeder
   (void)
void wdFeederTask
   (void)
STATUS wdFeederInit
   (void)
STATUS wdFeederHookAdd
   (FUNCPTR hookFunc, UINT32 hookArg)
STATUS wdFeederHookDelete
   (FUNCPTR hookFunc, UINT32 hookArg)
void wdStopFeeder
   (void)
void wdRestartFeeder
   (void)
void wdFeederHookShow
   (void)
```

DESCRIPTION

This module contains functions to install and maintain a watchdog feeding task to feed the GPU's processor sanitor watchdog. These functions also provide the ability to attach and detach hook functions which can be executed every time the watch dog is fed. These hook functions could check the sanity of the system every time the watchdog is fed.

This module is conditionally compiled based on the macro ROM_RESIDENT. If ROM_RESIDENT is defined the watchdog hook functions are not compiled. This is done so that they are not included in the GPU bootroms in order to save space. If ROM_RESIDENT is not defined then the watchdog hook functions are compiled which is the case when this module is linked with the downloaded VxWorks image.

WARNINGS

The wdFeeder task should execute at the highest vxWorks task priority. Not to be confused with the vxWorks watchdog timer library.

INCLUDE FILES

vxWorks.h, gpu960.h, glanErrno.h, sysLib.h, stdio.h, taskLib.h

SEE ALSO

sysLib

NAME

intMonLib - Interrupt Monitoring Library

SYNOPSIS

intCntrMon() - Interrupt Counters Monitor
intMonInit() - Initialize Interrupt monitoring
intMonPerSet() - Set the Interrupt Monitor Period (seconds)

```
void intCntrMon
   (void)
STATUS intMonInit
   (void)
void intMonPerSet
   (UINT32 seconds)
```

DESCRIPTION

This module contains functions to initialize and maintain CreditNet Switch interrupt monitoring. It currently does not cover NMI interrupt monitoring.

INCLUDE FILES

intMonLib.h, scuIntLib.h

SEE ALSO

scuIntLib, gpuNmiLib

NAME

sccLib - Low Level serial communications library

SYNOPSIS

sccOutputChar() - Output a single character via the SCC.
sccPrintString() - Output a string via the SCC.
sccPrintHexWord() - Output a 32-bit hexadecimal number.
sccWait() - Poll until SCC is no longer transmitting a char.
sccInit() - SCC initialization.

```
void sccOutputChar
   (outputChar) UINT8outputChar;
void sccPrintString
   (string) UINT8*string;
void sccPrintHexWord
   (value) UINT32value;
void sccWait
   (void)
void sccInit
   (baudRate, pSccCr,
   sccPclkRate) UINT32 baudRate; volatile UINT8 *pSccCr; UINT32sccPclkRate;
```

DESCRIPTION

This library contains a set of functions to initialize and manage “console” output. The vxWorks console is the GPU SCC port A. These routines strive not to use any memory so that they can be called from the bowels of the system code, possibly before any memory tests have been performed. The “console” SCC control register and data register pointers are stored in i960 internal RAM. These drivers also strive not to use interrupts and perform polled serial output. They are very simple and very fast.

Drivers in this module are used for output bootstrap info during early GPU startup diagnostics before the VxWorks serial driver is initialized and even before the GPU DRAM is tested.

NAME

scmUtil - SCM Utilities Library

SYNOPSIS

crc16() - CRC16 Calculation
scmMarch6Test() - SCM March 6 Memory Test
scmChkDramParity() - SCM Check for GPU DRAM Parity Error
scmChkFlashSig() - SCM Check Flash Signature
scmMicrosecondWait() - Wait n microseconds
scmWalkD1Test() - SCM Data-bit Walking One Test
scmWalkD0Test() - SCM Data-bit Walking Zero Test
scmWalkA1Test() - SCM Address-bit Walking One Test
scmRegTest() - SCM Register Test

```
short crc16
   (dataptr, imagelen) void *dataptr; unsigned intimagelen;
M6STATUS scmMarch6Test
   (UINT32 start, UINT32 size, UINT32 width,
   UINT32 foreground, BOOL chkDramPE)
STATUS scmChkDramParity
   (void)
STATUS scmChkFlashSig
   (void)
void scmMicrosecondWait
   (UINT32 microSeconds)
STATUS scmWalkD1Test
   (void * testLoc, UCHAR lsb, UCHAR msb, UINT32 mask,
   UCHAR memWidth, UCHAR flag)
STATUS scmWalkD0Test
   (void * testLoc, UCHAR lsb, UCHAR msb, UINT32 mask,
   UCHAR memWidth, UCHAR flag)
STATUS scmWalkA1Test
   (UINT32 startAdrs, UINT32 endAdrs, UINT32 mask,
   UCHAR memWidth, UCHAR flag)
STATUS scmRegTest
   (UINT32 regAdrs, UINT32 mask, UCHAR memWidth)
```

DESCRIPTION

This module contains utilities functions that can be used by various libraries written for the CreditNet Switch.

WARNINGS

The memory and register testing functions are capable of corrupting the state of the GPU, SCU or perhaps even the entire switch depending on the addresses which they are asked to test. They should only be invoked for debugging purposes or during startup diagnostics.

INCLUDE FILES

scmUtil.h, flashCtl.h, sccLib.h

SEE ALSO

gpuDiag, flashCtl, sccLib "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics"

sysInitCtl(1) CREDITNET SWITCH SW MANUAL sysInitCtl(1)

NAME

sysInitCtl - GLAN Switch System Initialization Controller Module

SYNOPSIS

sysInitCtl() - System Initialization Controller

```
void sysInitCtl
  (void)
```

DESCRIPTION

This module contains only function sysInitCtl() which is the control software for the entire System Initialization process while the switch is in the NIS_INIT_DIAG state.The man page for sysInitCtl() can be viewed by typing 'man 2 sysInitCtl'.

INCLUDE FILES

gpuDiag.h, scuSDiag.h ,scuSMDiag.h, sysInitCtl.h

SEE ALSO

"CreditNet Switch - System Initialization", sysInitCtl(2)

Copyright(2) CREDITNET SWITCH SW MANUAL Copyright(2)

NAME

Copyright() Copyright (c) 1990,1991,1992 Intel Corpora-tion

SYNOPSIS

Intel hereby grants you permission to copy, modify, and distribute this software and its documentation. Intel grant this permission provided that the above copyright notice appears in all copies and that both the copyright notice and this permission notice appear in supporting documentation. In addition, Intel grants this permission provided that you prominently mark as not part of the original any modification made to this software or documentation, and that the name of Intel Corporation not be used in advertising or publicity pertaining to distribution of the software or the documentation without specific, written prior permission.

SEE ALSO

scmUtil

NAME

crc16() - CRC16 Calculation

SYNOPSIS

```
short crc16 (dataptr, imagelen)
   void          *dataptr;
   unsigned int  imagelen;
```

DESCRIPTION

This function calculates the CRC16 checksum of a region of memory once given a pointer to the beginning of the region and the size. The CRC16 is calculated based on:

CRC16 polynomial: x**0 + x**2 + x**15 + x**16 (0xA001)
(CCITT polynomial: x**0 + x**5 + x**12 + x**16 (0x8408))
Initial condition: 0

The calculation is done using a mapping table rather than an actual formula in order to save time.

RETURNS

CRC16 Checksum

SEE ALSO

scmUtil,

NAME

dw() - Dump Word.

SYNOPSIS

```
void dw
   (
   void *  adrs
   )
```

DESCRIPTION

This function displays the contents of adrs formatted as a32-bit word. It calls d(adrs, 4, 4) to do this.

RETURNS

Nothing.

SEE ALSO

alias, d()

NAME

fDramTest() - Perform a Fast memory test

SYNOPSIS

```
fDramTest
  (
  void *  start,  /* Start address (must be in g0). */
  UINT32  size    /* Size in bytes (must be in g1). */
  )
```

DESCRIPTION

This routine can be called to perform a set of memory test that will detect a high proportion of memory system faults without consuming vast amounts of time. The test strives to detect stuck memory cell bits (in both main memory and parity memory for DRAM), address line shorts, and data bus shorts.

This function was cloned from the BNR Falcon project's memory test will be undertaken when and if time permits.

Note that address bus to data bus shorts are not detected here.

WARNINGS

It must be ensured that the execution time of each step of these tests remain under 2 seconds (watchdog reset interval). The test is performed with parity checking turned off since there is a known problem with the GPU parity error bits (which does not affect the DRAM sanity).

RETURNS

OK, or ERROR if any part of the test fails.

SEE ALSO

fDramTest, romInit(), "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics"

NAME

flashAddRamSize() - Flash Add RAM Size.

SYNOPSIS

```
STATUS flashAddRamSize
  (
  UINT32  ramSize  /* GPU RAM size in bytes. */
  )
```

DESCRIPTION

This routine adds the size of the GPU's DRAM into the second block of a Card ID. This RAM size block is inserted after the MAC ethernet block and before the boot parameters block. Once a Card ID has been upgraded by this function, the GPU's VxWorks image and bootrom must be upgraded so that the VxWorks macro 'LOCAL_MEM_SIZE' points to 0xf0000010 and the boot parameter line starts at 0xf000001c.

WARNINGS

This function should only be run once per Card ID. When it is run more than once it will add multiple copies of the RAM size block to the Card ID and throw off the starting addresses of the boot parameter and following blocks.

RETURNS

OK, or ERROR if a problem was encountered adding the RAM size.

SEE ALSO

flashCtl, flashSetRamSize()

flashBootParamsSet(2) CREDITNET SWITCH SW MANUAL flashBootParamsSet(2)

NAME

flashBootParamsSet() - Flash Boot Parameter Set.

SYNOPSIS

STATUS flashBootParamsSet (void)

DESCRIPTION

This routine allows a user to change a GPU's boot parameters which are stored in the Card ID Flash. Since the the boot parameters are in an EEPROM, the whole Card ID is copied to RAM, the boot parameters are changed (a RAM copy), then the complete Card is erased and re-programmed with the new boot parameters in place. Each block in the Card ID has its own CRC16 so this function must also re-calculate the CRC16 for the new boot parameter block and append this at the end o the block. It reuses the user interface presented by the VxWorks function bootParamsPrompt() which is identical to that presented when the booting seqence is halted (at the countdown) and the 'c' boot command is entered.

WARNINGS

Setting boot parameters incorrectly could result in the GPU (and therefore the switch) being unable to boot.

RETURNS

OK, or ERROR if unsuccessful.

SEE ALSO

flashCtl

flashCardIdUsage(2) CREDITNET SWITCH SW MANUAL flashCardIdUsage(2)

NAME

flashCardIdUsage() - Flash Card ID Usage.

SYNOPSIS

UINT32 flashCardIdUsage (void)

DESCRIPTION

This routine returns the number of bytes that are currently being occupied by valid data in the Card ID Flash.

RETURNS

Number of bytes used up in the Card ID.

SEE ALSO

flashCtl

NAME

flashCheckErase() - Check flash bank was erased.

SYNOPSIS

```
STATUS flashCheckErase
   (
   UCHAR *  devStart  /* Start address of device.            */
   )
```

DESCRIPTION

This function checks to ensure that the desired flash bank is erased.

RETURNS

OK if the flash bank was successfully erased and ERROR if it was not.

SEE ALSO

flashCtl, flashErase(), "AMD - Flash Memory Products - 1992/1993 Data Book/Handbook"

NAME

flashCopyFromRam() - Copy nbytes from RAM to a flash bank.

SYNOPSIS

```
STATUS flashCopyFromRam
   (
   UCHAR *  src,    /* Pointer to source data.            */
   UCHAR *  dest,   /* Pointer to destination in bootrom  */
   UINT32   nbytes  /* Number of bytes to copy to flash.  */
   )
```

DESCRIPTION

This function copies nbytes from RAM to a flash bank. It has the same programmer interface as bcopy().

RETURNS

OK, or ERROR if copying the desired bytes from RAM can not be accomplished.

WARNINGS

Flash programming will not be successful unless flashErase() is called first.

SEE ALSO

flashCtl, flashErase(), “AMD - Flash Memory Products - 1992/1993 Data Book/Handbook”

NAME

flashDLoad() - Download a file from a workstation.

SYNOPSIS

```
UINT32 flashDLoad
   (
   char *  file,  /* File to download.             */
   void *  buf    /* Buffer in which to place file. */
   )
```

DESCRIPTION

This function downloads a binary bootrom file from a workstation and places it in a predefined buffer. The file should be specified as “device:pathname/filename.”

RETURNS

Number of bytes downloaded from file.

WARNINGS

Flash programming will not be successful unless flashErase() and flashProgEnable() are called first.

SEE ALSO

flashCtl, flashProgAltBank()

NAME

flashDataPoll() - Polls a flash location.

SYNOPSIS

```
STATUS flashDataPoll
   (
   UCHAR *  dest,  /* Pointer to byte to check.    */
   UINT32   value  /* Expected value.              */
   )
```

DESCRIPTION

This function polls a location in flash EEPROM to verify that the byte has been programmed. It follows the algorithm outlined on page 2-178 of AMD's "Flash Memory Products - 1992/1993 Data Book/ Handbook." Note that bit 5 on the flash device is set to indicate that the device has timed out programming of the specified byte.

RETURNS

OK if bit 7 is the same as the expected value, or ERROR if bit 5 is set and bit 7 is not the same as the expected value.

SEE ALSO

flashCtl, "AMD - Flash Memory Products - 1992/1993 Data Book/Handbook"

NAME

flashErase() - Erase a flash bank.

SYNOPSIS

```
STATUS flashErase
    (
    UCHAR *  dest  /* Ptr to base adrs of bank  erased */
    )
```

DESCRIPTION

This function erases a flash bank. It must be passed a pointer to the starting address of the bank to be erased.

RETURNS

OK, or ERROR if the bank can not be erased or it was not passed the starting address of one of the flash banks.

SEE ALSO

flashCtl, flashCheckErase(), “AMD - Flash Memory Products - 1992/1993 Data Book/Handbook”

flashProgAltBank(2) **CREDITNET SWITCH SW MANUAL** **flashProgAltBank(2)**

NAME

flashProgAltBank() - Program alternate flash bank.

SYNOPSIS

```
STATUS flashProgAltBank
   (
   char *  fileName  /* Source bootrom file to program.      */
   )
```

DESCRIPTION

This function programs a bootrom image specified as a file on the switch worstation into the current alternate flash bank. It checks the CRC16 for the image after it is down- loaded from the workstation and placed in GPU RAM as well as after it has been programmed into the flash device. This function is responsible for determing and writing the bank signature into the last two bytes of the bootrom flash.

RETURNS

OK, or ERROR if programming the alternate flash bank can not be accomplished.

WARNINGS

Both bootrom flash devices should be programmed at the same time to ensure that they are in sync. Failure to do this could result in GPU Diagnostcs failing or even hanging when the GPU boots.

SEE ALSO

flashCtl, “AMD - Flash Memory Products - 1992/1993 Data Book/Handbook” flashSetRamSize(2) CREDITNET SWITCH SW MANUAL flashSetRamSize(2)

NAME

flashSetRamSize() - Flash Set RAM Size.

SYNOPSIS

```
STATUS flashSetRamSize
  (
  UINT32  ramSize  /* GPU RAM size in bytes. */
  )
```

DESCRIPTION

This routine sets the size of the GPU's DRAM which lies in the second block of a Card ID. This RAM size is pointed to by the VxWorks macro LOCAL_MEM_SIZE (which points to 0xf0000010).

WARNINGS

Setting the RAM size incorrectly can result in DRAM testing failing during GPU startup diagnostics or even in the GPU (and therefore switch) being unable to boot.

RETURNS

OK, or ERROR if a problem was encountered setting the RAM size.

SEE ALSO

flashCtl, flashAddRamSize()

NAME

generateCell() - Generate Cell

SYNOPSIS

```
CELL_STRUCT * generateCell
   (
   UINT16  ivc,     /* 14-bit IVC value                    */
   char    gfc,     /* 4 bits of GFC                       */
   char    vpi,     /* 8 bits of VPI information           */
   UINT16  vci,     /* 16 bits of VCi information          */
   char    plt,     /* 4 bits of payload type              */
   UINT32  pattern  /* 16-bit pattern used to fill the payload */
   )
```

DESCRIPTION

This function generates a cell based on the parameters passed into the function. Any parameter except for the IVC can be NULL. NULLs will be replaced with default values for GFC, VPI, VCI, PLT and pattern. The GFC, VPI, VCI, and PLT are arranged in proper bit positions within the header. CRC-8 is not currently calculated or included in the header.

RETURNS

Pointer to CELL_STRUCT (a cell).

SEE ALSO

generateCell,

NAME

gpuCardIdTest() - GPU Card ID Test

SYNOPSIS

STATUS gpuCardIdTest (void)

DESCRIPTION

This routine tests the GPU's CARD ID EEPROM by performing a CRC16 test on each of its data blocks. The Card ID is composed of various blocks, each of which starts with a 32-bit size and ends with a 16-bit CRC16 calculated over size - 2 bytes of the block starting with the first byte of the size. For the purposes of this test, the format of the data between the size and CRC16 is unimportant. The very first thing that should be found in the Card ID is therefore the size of the first block. This test uses the block size to work its way through the Card ID's blocks, calculate the blocks' CRC16 and compare this to the blocks' CRC16 values stored at the end of the blocks.

RETURNS

OK, or ERROR if the CRC16 test fails on any block. It also returns ERROR if the first block in the Card ID has a block size of 0x0 or 0xffffffff which would indicate that the Card ID is empty or erased.

SEE ALSO

gpuDiag, romInit(), gpuDiagCtl(), "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics"

NAME

gpuCntrlRegTest() - GPU Control Register Test

SYNOPSIS

STATUS gpuCntrlRegTest (void)

DESCRIPTION

This function tests the GPU's control status register. This register is composed of:

BIT 0-3 Unused.
BIT 4 FLASH_SWAP_JMP_ENABLED
BIT 5 NMI_DISABLED
BIT 6 FLASH_PROG_ENABLED
BIT 7 FLASH_SWAP_SW_ENABLED

All but the FLASH_SWAP_JMP_ENABLED bit are tested by this function. Since it involves physically changing the position of this jumper in order to varify its operation. Of the bit under test, they will be tested to varify that they can change from 0 to 1 and from 1 to 0. If bits other than the bit under test fail this routine will signal an error but will not report whether the bit under test failed or another bit was altered. Only the scenario under which the error was encountered will be reported. Error messages will only be reported if this routine is not running out of bootrom (ie. ROM_RESIDENT is not defined).

WARNINGS

This function should be run with the instruction cache turned on. It also runs with with tasks locked so that it can starve any other tasks that are running.

RETURNS

OK, or ERROR if testing any one of the bits fails.

SEE ALSO

gpuDiag, gpuDiagCtl(), "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics"

NAME

gpuDiagCtl() - GPU Diagnostics Controller

SYNOPSIS

```
STATUS gpuDiagCtl
   (
   UINT32  testMask  /* Mask indicating tests to perform. */
   )
```

DESCRIPTION

This function controls execution of GPU diagnostics tests based on the value of the testMask passed to it. It invokes a particular test for each valid bit that is set in the test mask (see table). When the test mask is 0x0, it determines the results of the GPU diagnostics that were performed while the GPU was booting and reports them to the switch workstation.

Bit	Test to perform
0	gpuEepromTest
1	gpuCardIdTest
2	gpuTimIntTest
3	gpuEtherTest
4	gpuSccTest
5	gpuErrRegTest

This test also sends messages to the VxWorks Shell and switch UI regarding the pass or fail status of each test it executes.

WARNINGS

Users should be careful when invoking these tests from the VxWorks Shell because they could place the GPU in an unusable state. A test particularily dangerous for this is gpuEtherTest().

RETURNS

OK, or ERROR if any of the GPU diagnostics tests invoked by this function return ERROR.

SEE ALSO

gpuDiag, usrRoot(), “CreditNet GPU Specification”, “CreditNet Switch - GPU Diagnostics”

NAME

gpuDipSwTest() - GPU DIP Switch Test

SYNOPSIS

STATUS gpuDipSwTest (void)

DESCRIPTION

This function tests the GPU's 8 DIP switches. It requires someone to be present near the GPU in order to toggle the DIP switches when prompted.

RETURNS

OK, or ERROR if the test fails.

WARNINGS

SEE ALSO

gpuDiag, "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics" gpuEDramTest(2) CREDITNET SWITCH SW MANUAL gpuEDramTest(2)

NAME

gpuEDramTest() - GPU Extended DRAM Test

SYNOPSIS

```
M6STATUS gpuEDramTest
   (
   UINT32  start,  /* Start address of memory.  */
   UINT32  size,   /* Size of memory.           */
   UINT32  width   /* Width of memory in bytes. */
   )
```

DESCRIPTION

This function performs an extended memory test on the GPU's DRAM. It tests for stuck address, data, and parity bits as well as address decoding and leakages. It invokes scmMarch6Test() in module scmUtil.c to perform most of the testing.

RETURNS

OK, or the first address which contains a fault.

WARNINGS

This test obliterates the contents of the memory it tests and caution should be exercised to ensure that the memory region being tested does not contain important or needed data.

SEE ALSO

gpuDiag

NAME

gpuEepromTest() - GPU EEPROM Test

SYNOPSIS

STATUS gpuEepromTest (void)

DESCRIPTION

This function performs a CRC16 test on the entire primaryEEPROM bank. It then compares the calculated CRC16 value to the value found at the end of the bank under test and returns STATUS based on the comparison. It also checks the bank signature at the end of the current primary bank.

WARNINGS

Software or users calling this routine should probably swap EEPROM banks and reboot the GPU if this test fails since the current bank is not sane.

RETURNS

OK, or ERROR if the calculated CRC16 does not match the one stored at the end of the EEPROM or the bank signature is incorrect for the bank.

SEE ALSO

gpuDiag, gpuDiagCtl(), romInit(), flashCtl, flashProgAlt-Bank(), “CreditNet GPU Specification”, “Credit Net Switch - GPU Diagnostics”

NAME

gpuErrRegTest() - GPU NMI Error Source Register Test

SYNOPSIS

STATUS gpuErrRegTest (void)

DESCRIPTION

This routine tests both the GPU's NMI interrupt and the NMI Error Source register. It tests the bus timeout, parity error and watchdog timer interrupt bits in the NMI Error Source register.

WARNINGS

This function should be run with the instruction cache turned on. It also runs with with tasks locked so that it can starve any other tasks that are running. There are some know problems with some GPUs when this test is run.

RETURNS

OK, or ERROR if the test fails.

SEE ALSO

gpuDiag, gpuDiagCtl(), "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics"

NAME

gpuEtherTest() - Test the i82596ca ethernet controller

SYNOPSIS

STATUS gpuEtherTest (void)

DESCRIPTION

This function tests the LAN co-processor device (Intel 82596CA). It performs the port command self-test and analyzes the results. This tests the devices ROM contents, parallel registers, bus throttle timer and serial subsystem.

WARNINGS

This test often fails on many GPUs and is a know problem on the GPUs. Most GPU always are able to boot and operate correctly even if this test fails.

RETURNS

OK, or ERROR if the test fails for any reason.

SEE ALSO

gpuDiag, gpuDiagCtl(), “CreditNet GPU Specification”, “Credit Net Switch - GPU Diagnostics”

NAME

gpuExtBusTest() - GPU External Bus Test

SYNOPSIS

STATUS gpuExtBusTest (void)

DESCRIPTION

This function performs a loopback test of the external bus. In order for this test to execute, a special test jig must also be connected to J2 and J3 which loops the A31-2 to D31-2 and BE1-BE0 to D1-D0. This test does not actually return but instead runs forever so a user can check the external bus bits with a scope or view the bit values printed to the Shell.

RETURNS

Never.

SEE ALSO

gpuDiag, “CreditNet GPU Specification”, “Credit Net Switch - GPU Diagnostics”

NAME

gpuFlSwapTest() - GPU Flash Swap Test

SYNOPSIS

STATUS gpuFlSwapTest (void)

DESCRIPTION

This function performs a flash bank swapping test to ensure that the banks can actually be swapped by use of the software bank swapping register on the GPU. It does this by checking the bank signatures found at the end of each bank and ensuring that they are different.

WARNINGS

This function can potentially hang if the alternate bank has been corrupted, is erased or is even out of date with the primary bank. Users should not run this test when the alternate bank is under any of the above conditions and should probably check the sanity of the alternate bank before running this test.

RETURNS

OK, or ERROR if bank swapping is not working correctly.

SEE ALSO

gpuDiag, gpuDiagCtl(), flashCtl, flashProgAltBank(), "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics"

NAME

gpuNmiInit() - GPU NMI Initialization

SYNOPSIS

```
STATUS gpuNmiInit
   (
   VOIDFUNCPTR  nmiIsr
   )
```

DESCRIPTION

This function attaches a function to the NMI vector, clears NMI registers, enables NMIs and starts the NMI monitoring task.

RETURNS

OK, or ERROR if initialization was unsuccessful.

SEE ALSO

gpuNmiLib, gpuNmiMonStart(), gpuNmiMonTask()

NAME

gpuNmiInitIsr() - GPU NMI Initialization ISR

SYNOPSIS

void gpuNmiInitIsr (void)

DESCRIPTION

This is the NMI service routine used during System Initialization. It can be replaced using gpuNmiInit() by another service routine if the application software so desires. Normally, when NMI monitorring is running (a "tNmiMonitor" task will exist) this routine will just increment the appropriate NMI counter and OR together the NMI status bits currently set with a global NMI status bit variable. When NMI monitoring is not enabled, it will print out a short one line message regarding the NMI that occured.

RETURNS

Nothing.

SEE ALSO

gpuNmiLib, gpuNmiMonStart(), gpuNmiMonStop(). gpuNmiMon(), gpuNmiMonTask(), gpuNmiInit().

NAME

gpuNmiMon() - GPU NMI Monitor

SYNOPSIS

void gpuNmiMon (void)

DESCRIPTION

This function monitors GPU Non-maskable Interrupt counters by checking to see if they are non-zero, sending their values to the console if they are non-zero then clearing the counters. It is called by gpuNmiMonTask() which executes as VxWorks task "tNmiMonitor".

RETURNS

Nothing.

SEE ALSO

gpuNmiLib, gpuNmiMonStart(), gpuNmiMonStop(). gpuNmiMon-Task()

gpuNmiMonStart(2) CREDITNET SWITCH SW MANUAL gpuNmiMonStart(2)

NAME

gpuNmiMonStart() - Start NMI counter monitoring

SYNOPSIS

STATUS gpuNmiMonStart (void)

DESCRIPTION

This function spawns the GPU Monitor, gpuNmiMonTask(). It first verifies that the Monitor is not already running then initializes the NMI counters to 0.

RETURNS

ERROR if a GPU Monitor task exists or taskSpawn() returns ERROR. Otherwise it returns OK.

SEE ALSO

gpuNmiLib, gpuNmiMonTask(), gpuNmiMonStop(). gpuNmiMon()

NAME

gpuNmiMonStop() - Stop NMI counter monitoring

SYNOPSIS

STATUS gpuNmiMonStop (void)

DESCRIPTION

This function stops GPU NMI monitoring by killing the GPU NMI Monitor task, “tNmiMonitor”, if one exists.

RETURNS

The status returned from taskDelete() or ERROR if the Monitor task does not exist.

SEE ALSO

gpuNmiLib, gpuNmiMonStart(), gpuNmiMonTask(). gpuNmiMon()

NAME

gpuNmiMonTask() - The GPU NMI Monitoring Task

SYNOPSIS

void gpuNmiMonTask (void)

DESCRIPTION

This function is the main function for GPU NMI monitoring. It simply calls function gpuNmiMon() and sleeps for NMI_MON_PER seconds.

RETURNS

Nothing, absitively, posolutely nothing.

SEE ALSO

gpuNmiLib, gpuNmiMonStart(), gpuNmiMonStop(). gpuNmiMon()

NAME

gpuSccTest() - Test the 85C30 SCC on the GPU

SYNOPSIS

STATUS gpuSccTest (void)

DESCRIPTION

This routine tests the SCC (Serial Communication Controller) device by performing an internal loop-back test. After the test it re-enables SCC interrupts so that the SCC can still be used as an interface to the VxWorks Shell.

WARNINGS

It is assumed that the device is already initialized with TX and RX sections enabled. Also, this test may fail if you hit keys at the serial terminal while the test is running.

RETURNS

OK, or error if the test fails for any reason.

SEE ALSO

gpuDiag, gpuDiagCtl(), "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics"

NAME

gpuTimIntTest() - GPU Real Time Clock Test

SYNOPSIS

STATUS gpuTimIntTest (void)

DESCRIPTION

This routine tests both the GPU's real time clock interrupt and the real time clock period.

WARNINGS

This function should be run with the instruction cache turned on. It also runs with with tasks locked so that it can starve any other tasks that are running.

RETURNS

OK, or ERROR if the test fails.

SEE ALSO

gpuDiag, gpuDiagCtl(), "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics"

NAME

intCntrMon() - Interrupt Counters Monitor

SYNOPSIS

void intCntrMon (void)

DESCRIPTION

This function wakes up every ‘intMonPeriod’ seconds to examine the value of the SCU and PM interrupt counters and send messages about the nonzero counters to the switch UI which it does by calling functions scuIntMon() and pmIntMon().

RETURNS

Nothing, absitively, posolutely nothing.

SEE ALSO

intMonLib, scuIntMon(

intMonInit(2) **CREDITNET SWITCH SW MANUAL** **intMonInit(2)**

NAME

intMonInit() - Initialize Interrupt monitoring

SYNOPSIS

STATUS intMonInit (void)

DESCRIPTION

This function sets the length of the default interrupt monitor period then spawns the interrupt monitor, function intCntrMon().

RETURNS

OK, or ERROR if could not spawn intCntrMon()

SEE ALSO

intMonLib, intCntrMon(), intMonPerSet()

NAME

intMonPerSet() - Set the Interrupt Monitor Period (seconds)

SYNOPSIS

```
void intMonPerSet
   (
   UINT32  seconds
   )
```

DESCRIPTION

This function sets intMonPeriod to seconds if seconds is non-zero or disables interrupt monitorring if seconds is zero.

RETURNS

Nothing, absitively, posolutely nothing.

SEE ALSO

intMonLib, intCntrMon(), intMonInit()

NAME

ramDebug() - RAM Debug

SYNOPSIS

```
STATUS  ramDebug
   (
   void *  start,     /* Start address.                        */
   UINT32  size,      /* Size of memory in bytes.              */
   UINT32  mask,      /* Mask of bits to test.                 */
   UCHAR   memWidth,  /* Width of testLoc region (bytes)       */
   BOOL    forever    /* FALSE(0)=once; TRUE(1)=repeat forever. */
   )
```

DESCRIPTION

This function calls qmemTest() to perform a comprehensive memory test. If qmemTest() returns ERROR it prints an error message indicating the first bad address found by qmemTest() as well as the data written to that address and the incorrect data read from it. If forever is FALSE (0) it only calls qmemTest() once then returns the value returned from qmemTest(). If forever is TRUE (1) it calls qmemTest() over and over again.

RETURNS

OK if forever is FALSE and qMemTest() returns OK; ERROR if forever is FALSE and qmemTest() returns ERROR; nothing if forever is TRUE.

SEE ALSO

alias, qmemTest()

NAME

sccInit() - SCC initialization.

SYNOPSIS

```
void sccInit (baudRate, pSccCr, sccPclkRate)
   UINT32          baudRate;
   volatile UINT8  *pSccCr;
   UINT32          sccPclkRate;
```

DESCRIPTION

Initialize a port of a Zilog Z85C30 SCC device. The desired baud rate is passed through g0. The address of the command register is passed in g1. The input PCLK (in Hz) of the SCC device is passed in g2.

RETURNS

Nothing.

SEE ALSO

sccLib, "AMD - Personal Computer Products - Chapter 2 - Z85C30 Data Sheet"

sccOutputChar(2) **CREDITNET SWITCH SW MANUAL** **sccOutputChar(2)**

NAME

sccOutputChar() - Output a single character via the SCC.

SYNOPSIS

```
void sccOutputChar (outputChar)
   UINT8  outputChar;
```

DESCRIPTION

Output a character to an SCC port A - for debugging.

RETURNS

Nothing.

SEE ALSO

sccLib, sccPrintString(), sccPrintHexWord(), “AMD - Personal Computer Products - Chapter 2 - Z85C30 Data Sheet”

NAME

sccPrintHexWord() - Output a 32-bit hexadecimal number.

SYNOPSIS

```
void sccPrintHexWord (value)
   UINT32  value;
```

DESCRIPTION

Output a 32bit value in hexadecimal.

RETURNS

Nothing.

SEE ALSO

sccLib, sccPrintString(), sccOutputChar(), “AMD - Personal Computer Products - Chapter 2 - Z85C30 Data Sheet”

NAME

sccPrintString() - Output a string via the SCC.

SYNOPSIS

```
void sccPrintString (string)
   UINT8*  string;
```

DESCRIPTION

Output a null-terminated fixed string to SCC port A.

RETURNS

Nothing.

SEE ALSO

sccLib, sccOutputChar(), sccPrintHexWord(), “AMD - Personal Computer Products - Chapter 2 - Z85C30 Data Sheet” sccWait(2) **CREDITNET SWITCH SW MANUAL** sccWait(2)

NAME

sccWait() - Poll until SCC is no longer transmitting a char.

SYNOPSIS

void sccWait (void)

DESCRIPTION

Wait for the currently selected SCC to finish transmitting a char.

RETURNS

Nothing.

SEE ALSO

sccLib, “AMD - Personal Computer Products - Chapter 2 - Z85C30 Data Sheet”

NAME

scmChkDramParity() - SCM Check for GPU DRAM Parity Error

SYNOPSIS

STATUS scmChkDramParity (void)

DESCRIPTION

This function checks the GPU's Error Source register to see if a DRAM parity error has occured due to a CPU bus cycle. If such an error has occured then it clears the NMI error registers and returns ERROR. This function should not be used to find parity errors on memories other than the GPU's DRAM.

RETURNS

OK, or ERROR if a DRAM parity error has occured.

WARNINGS

This test only checks for DRAM parity errors caused by a CPU bus cycle and will not clear the GPU's Error Source register if any other NMI error has occured.

SEE ALSO

scmUtil, scmMarch6Test(), "CreditNet GPU Specification", "Credit Net Switch - GPU Diagnostics"

NAME

scmChkFlashSig() - SCM Check Flash Signature

SYNOPSIS

STATUS scmChkFlashSig (void)

DESCRIPTION

This function checks the flash signature of the logical pri- mary bank (according to the two flash swap control bits in the GPU control status register) and compares them to their expected values.

RETURNS

OK, or ERROR if an incorrect flash signature is found.

SEE ALSO

scmUtil, “CreditNet GPU Specification”, “Credit Net Switch - GPU Diagnostics”

NAME

scmMarch6Test() - SCM March 6 Memory Test

SYNOPSIS

```
M6STATUS scmMarch6Test
   (
   UINT32  start,       /* Start address of memory to test.            */
   UINT32  size,        /* Size of memory in bytes.                    */
   UINT32  width,       /* Width of memory in bytes.                   */
   UINT32  foreground,  /* Foreground pattern.                         */
   BOOL    chkDramPE    /* 1=Check DRAM par. err., 0=don't.            */
   )
```

DESCRIPTION

This function performs the March 6 algorithm on a section of memory. This test checks for stuck address and data bits, leakages and decoding problems. If being performed during GPU startup diagnostics, this test must feed the watchdog timer at intervals less than 2 seconds or else a watchdog NMI will occur. This is a fast test which uses fixed pattern and is suitable for large memories. It will catch a high percentage of memory faults but it does not provide as much coverage as qMemTest() which performs multiple passes with random patterns.

RETURNS

OK, or the first address which contains a fault.

WARNINGS

This test obliterates the contents of the memory it tests and caution should be exercised to ensure that the memory region under test does not contain important or needed data.

SEE ALSO

scmUtil, gpuDiag, qMemTest()

NAME

scmMicrosecondWait() - Wait n microseconds

SYNOPSIS

```
void scmMicrosecondWait
  (
  UINT32  microSeconds  /* Number of seconds to delay. */
  )
```

DESCRIPTION

This function provides a delay approximately microSeconds long provided it is not interrupted for any reason. The calling task should not use this routine for periods greater than the interval between system clock ticks (default 20ms) since this would constitute an inefficient use of the processor. Use the vxWorks task delay facility instead.

```
Note1: Callx --> 1.44us
       rts   --> 0.36us
Note2: If callx is used, the minimum delay is 3.10us due  to
       callx and rts overhead.
```

microSeconds	#us actual
0	3.10
1	3.10
2	3.10
3	3.10
4	4.56
5	5.76
6	6.80
7	7.76
8	8.80
9	9.76
10	10.80
100	100.76
1000	1000.48
10000	10081.44

SEE ALSO

scmUtil, taskDelay()

NAME

scmRegTest() - SCM Register Test

SYNOPSIS

```
STATUS scmRegTest
  (
  UINT32  regAdrs,  /* Ptr to register to test.                 */
  UINT32  mask,     /* Mask of bits to test.                    */
  UCHAR   memWidth  /* Width of testLoc region ( bytes)         */
  )
```

DESCRIPTION

This function performs a simple register test using functions scuSMWalkD1Test() and scuSMWalkD0Test(). Calling functions pass it a register address, memory region width and mask.

RETURNS

OK, or ERROR if scmWalkD1Test() or scmWalkD0Test() return ERROR.

WARNINGS

This test obliterates the contents of the memory or register it tests and caution should be exercised to ensure that the memory region under test does not contain important or needed data.

SEE ALSO

scmUtil, scmWalkD0Test(), scmWalkD1Test()

NAME

scmWalkA1Test() - SCM Address-bit Walking One Test

SYNOPSIS

```
STATUS scmWalkA1Test
   (
   UINT32  startAdrs,  /* Ptr to beginning of test region.            */
   UINT32  endAdrs,    /* Ptr to beginning of test region.            */
   UINT32  mask,       /* Mask of bits to test.                       */
   UCHAR   memWidth,   /* Width of testLoc region ( bytes)            */
   UCHAR   flag        /* 0 = CONTINUOUS, 1 = ONCE.                   */
   )
```

DESCRIPTION

This function walks a one across the address bits for a region of memory. It is highly configurable in that the user can specify the beginning and end of the address range, a mask which causes certain data bits to be skipped and the width of the memory region in which the test is to be per- formed in order to control the type of read and write cycles that get used. A flag is also passed to this function to instruct it to perform the test once with error checking or repeatedly without error checking.

WARNINGS

If testing a 32-bit region, this test skips addresses ending in 0x01 and 0x02 since data written to these locations would overlap with that written address ending in 0x00 and 0x04. Also, if testing a 16-bit region, this test skips addresses ending in 0x01 since data written there would overlap with data written to addresses ending 0x00 and 0x02. Therefore, it may be necessary to repeat the test as if the region tested were only 8-bits wide in order to cover the above exceptions when testing 16 and 32-bit regions.

This test obliterates the contents of the memory it tests and caution should be exercised to ensure that the memory region under test does not contain important or needed data.

RETURNS

OK, or ERROR if any pattern read from the address range is not the same as that written there. It does not return or perform error checking if flag is set to CONTINUOUS (0).

SEE ALSO

scmUtil,

NAME

scmWalkD0Test() - SCM Data-bit Walking Zero Test

SYNOPSIS

```
STATUS scmWalkD0Test
  (
  void *  testLoc,   /* Location to test.                        */
  UCHAR   lsb,       /* Least significant bit to start.          */
  UCHAR   msb,       /* Most significant bit to stop.            */
  UINT32  mask,      /* Mask of bits to test.                    */
  UCHAR   memWidth,  /* Width of testLoc region ( bytes)         */
  UCHAR   flag       /* 0 = CONTINUOUS, 1 = ONCE.                */
  )
```

DESCRIPTION

This function walks a zero across a register or memory location. It is highly configurable in that the user can specify the range of bits covered during the walk, a mask which causes certain bits to be skipped and the width of the memory region in which the register or location exists in order to control the type of read and write cycle that gets used. A flag is also passed to this function to instruct it to perform the test once with error checking or repeatedly without error checking.

RETURNS

OK, or ERROR if any pattern read from the register is not the same as that written there. It does not return or perform error checking if flag is set to CONTINUOUS (0).

WARNINGS

This test obliterates the contents of the memory or register it tests and caution should be exercised to ensure that the memory region under test does not contain important or needed data.

SEE ALSO

scmUtil, scmWaldD1Test(), scmRegTest()

NAME

scmWalkD1Test() - SCM Data-bit Walking One Test

SYNOPSIS

```
STATUS scmWalkD1Test
   (
   void *  testLoc,   /* Location to test.                    */
   UCHAR   lsb,       /* Least significant bit to start.      */
   UCHAR   msb,       /* Most significant bit to stop.        */
   UINT32  mask,      /* Mask of bits to test.                */
   UCHAR   memWidth,  /* Width of testLoc region ( bytes)     */
   UCHAR   flag       /* 0 = CONTINUOUS, 1 = ONCE.            */
   )
```

DESCRIPTION

This function walks a one across a register or memory location. It is highly configurable in that the user can specify the range of bits covered during the walk, a mask which causes certain bits to be skipped and the width of the memory region in which the register or location exists in order to control the type of read and write cycle that gets used. A flag is also passed to this function to instruct it to perform the test once with error checking or repeatedly without error checking.

RETURNS

OK, or ERROR if any pattern read from the register is not the same as that written there. It does not return or perform error checking if flag is set to CONTINUOUS (0).

WARNINGS

This test obliterates the contents of the memory or register it tests and caution should be exercised to ensure that the memory region under test does not contain important or needed data.

SEE ALSO

scmUtil, scmWaldD1Test(), scmRegTest()

NAME

sysInitCtl() - System Initialization Controller

SYNOPSIS

void sysInitCtl (void)

DESCRIPTION

This function controls the process of CreditNet Switch System Initialization. It should only be invoked after VxWorks has been booted on the GPU. It initializes the switch for application SW use. It intiates the iniitialization and diagnostics of switch HW including the SCU, PMs and TLGs as well as the communication between these entities. It also initializes communication between the GPU and Switch UI running on the Switch Workstation and sends the first messages to the UI. This function brings the switch to the end of the NIS_BOOT state, takes it through the NIS_INIT_DIAG state and leave it in the NIS_READY state upon successful completion.

RETURNS

Nothing

WARNINGS

This function will obliterate the current state of the switch and initialize it to know values.

SEE ALSO

sysInitCtl, "CreditNet Switch - System Initialization"

NAME

wdFeeder() - Watchdog feeder function

SYNOPSIS

void wdFeeder (void)

DESCRIPTION

This function is the main function for the watchdog feeder task and is called by wdFeederTask() periodically. It invokes each hook function in turn and if ERROR is returned by any of these hook routines the watchdog is NOT fed. Note that the card should not be reset by this function. Let the watchdog circuitry do it, if the watchdog disable jumper is not installed. If all hook functions return OK then the watchdog is fed by calling sysWdFeed(). Even if watchdog feeding is disabled (by wdStopFeeder()) this function still runs in order to execute the hook functions.

RETURNS

Nothing, absitively, posolutely nothing.

SEE ALSO

gpuWdLib, sysWdFeed(), wdFeederTask(), wdFeederInit(), wdFeederHookAdd, wdFeederHookDelete(), wdStopFeeder, wdRestartFeeder(), wdFeederHookShow().

NAME

wdFeederHookAdd() - Add hook routine to WD feeder task

SYNOPSIS

```
STATUS wdFeederHookAdd
   (
   FUNCPTR  hookFunc,
   UINT32   hookArg
   )
```

DESCRIPTION

This function adds the desired function to the watchdog hook function table. Each function in this table is executed by wdFeeder() before the watchdog is actually fed. These functions should test the sanity of their subsystems and if all is well return OK. If ERROR is returned by any hook the watchdog will not be fed, by wdFeeder() i.e. let that darn dog starve to death!!

The hook routine must have the form:
STATUS feederHook (unsigned int userArg)

RETURNS

OK, or ERROR if hook table is full.

SEE ALSO

gpuWdLib, wdFeeder(), wdFeederHookDelete(), wdFeederHookShow().

NAME

wdFeederHookDelete() - Delete WD feeder task hook routine

SYNOPSIS

```
STATUS wdFeederHookDelete
   (
   FUNCPTR  hookFunc,
   UINT32   hookArg
   )
```

DESCRIPTION

The first instance of the hook routine with the matched argument will be removed from the table.

RETURNS

OK, or ERROR if hook function is not in hook function table.

SEE ALSO

gpuWdLib, wdFeeder(), wdFeederHookAdd(), wdFeederHookShow().

wdFeederHookShow(2) **CREDITNET SWITCH SW MANUAL** wdFeederHookShow(2)

NAME

wdFeederHookShow() - Displays WD hook routines

SYNOPSIS

void wdFeederHookShow (void)

DESCRIPTION

The list of hook routines and the paramters that are passed to them are displayed on the console in symbolic form. If the function symbols are not present in the system symbol table the closest match will be printed.

RETURNS

void

SEE ALSO

gpuWdLib, wdFeederHookAdd(), wdFeederHookDelete(), wdFeeder().

NAME

wdFeederInit() - Initialize watchdog feeder library

SYNOPSIS

STATUS wdFeederInit (void)

DESCRIPTION

This function initializes a boolean wdKeepFeeding to TRUE, initializes the watchdog hook function table to indicate there are no hook functions then spawns wdFeederTask at priority 0.

RETURNS

OK, or ERROR if could not spawn feeder task

SEE ALSO

gpuWdLib, wdFeederTask(), wdFeederHookAdd().

NAME

wdFeederTask() - Watchdog feeder task

SYNOPSIS

void wdFeederTask (void)

DESCRIPTION

This function is the main function for the watchdog feeder task. It simply calls wdFeeder() then then sleeps until the next time it should feed the watchdog.

RETURNS

nothing, absitively, posolutely nothing.

SEE ALSO

gpuWdLib, wdFeeder(), wdFeederRestart(), wdFeederStop(), wdFeederInit().

wdRestartFeeder(2) **CREDITNET SWITCH SW MANUAL** **wdRestartFeeder(2)**

NAME

wdRestartFeeder() - Re-enable the feeding of the watchdog

SYNOPSIS

void wdRestartFeeder (void)

DESCRIPTION

This function re-enables watchdog feeding which occurs when wdFeeder() executes.

RETURNS

void

SEE ALSO

gpuWdLib, wdFeeder(), wdStopFeeder().

wdStopFeeder(2) CREDITNET SWITCH SW MANUAL wdStopFeeder(2)

NAME

wdStopFeeder() - Force the disabling of watchdog feeds

SYNOPSIS

void wdStopFeeder (void)

DESCRIPTION

This function disables watchdog feeding which occurs when wdFeeder() executes.

RETURNS

void

SEE ALSO

gpuWdLib, wdFeeder(), wdRestartFeeder().

ispUtil(1) CREDITNET SWITCH SW MANUAL ispUtil(1)

NAME

ispUtil - ISP Utilities

SYNOPSIS

ispProgram() - Programs an ISP Lattice Device
ispRead() - Reads an ISP Lattice Device
ispVerify() - Verifies an ISP Lattice Device
ispEnValCalc() - Calculate the ISP Programming Register
Value

```
STATUS ispProgram
   (char * fileName, char scuOrCore, char deviceNum)
STATUS ispRead
   (char * fileName, char scuOrCore, char deviceNum)
STATUS ispVerify
   (char * fileName, char scuOrCore, char deviceNum)
INT16 ispEnValCalc
   (char scuOrCore, char deviceNum)
```

DESCRIPTION

This module contains a set of utilities to operate on the In System Programmable (ISP) Lattice devices on the SCU (and Core Memory). All functions require users to specify whether an SCU or Core Memory device is being operated on and its corresponding device number. ispProgram() and ispVerify() also require the user to specify the filename of the Jedec file used for programming or verification.

WARNINGS

Although this SW supports programming of Core Memory devices, the HW and SW to support this have not been tested and debugged yet so it is not operational. The function ispRead also has not been tested (or needed) yet.

INCLUDE FILES

ispUtil.h, isplsi.h, isp_port.h

SEE ALSO

"Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

NAME

scuDrv - SCU Driver Library

SYNOPSIS

scuCellCpyRamToSb() - Copy Cell GPU RAM->Shared Buffer
scuCellCpySbToRam() - Copy Cell Shared Buffer->GPU RAM
scu64BCellDecode() - SCU 64-Byte Cell Decode to 56-Bytes
scuMap8To7() - SCU Map 8 TO 7 (byte)
scuVerCorePat() - SCU Verify Core Pattern
scuCifCntlRegWrite() - SCU Write to CIF Control Register
scuCifStatRegRead() - SCU Read from CIF Status Register
scuCifCntlRegRead() - SCU Read from CIF Control Register
scuCifWrPoll() - SCU CIF Write Cycle Poll
scuCifRdPoll() - SCU CIF Read Cycle Poll
scuCifRdWrPoll() - SCU CIF Read/Write Cycle Poll
scuWivcLutSetup() - SCU W_IVC Lookup Table Setup
scuWivcLutSet() - SCU W_IVC Lookup Table Entry Set
scuWivcLutInit() - SCU W_IVC Lookup Table Initialization
scuRivcLutSetup() - SCU R_IVC Lookup Table Setup
scuRivcLutSet() - SCU R_IVC Lookup Table Entry Set
scuRivcLutInit() - SCU R_IVC Lookup Table Initialization
scuCifWivcFifoRead() - SCU CIF W_IVC FIFO Read
scuCifRivcFifoRead() - SCU CIF R_IVC FIFO Read
scuCifHPWFifoStatRead() - SCU CIF HP W_IVC FIFO Status Read
scuCifLPWFifoStatRead() - SCU CIF LP W_IVC FIFO Status Read
scuCifRFifoStatRead() - SCU CIF R_IVC FIFO Status Read
scuFlmCmbfCountRead() - SCU FLM CMBF Counter Read
scuFlmCntlRegWrite() - SCU Write to FLM Control Register
scuFlmStatRegRead() - SCU Read from FLM Status Register
scuFlmCntlRegRead() - SCU Read from FLM Control Register
scuFlmFreeListInit() - SCU FLM Free List Initialization
scuFlmFLPush() - SCU FLM Free List Push
scuFlmFLArrayPush() - SCU FLM Free List Array Push
scuFlmFLCInit() - SCU FLM Free List Complete Initialization
scuFlmHalt() - SCU FLM Halt
scuFlmHaltClear() - SCU FLM Halt Clear
scuFlmReset() - SCU FLM Reset
scuFlmPrime() - SCU FLM Prime
scuDbmSetup() - SCU Distribution Bitmap Setup
scuDbmSet() - SCU Distribution Bitmap Entry Set
scuDbmInit() - SCU Distribution Bitmap Initialization
scuDbmCapture() - SCU Distribution Bitmap Capture
scuWDbmCapture() - SCU Working Distribution Bitmap Capture
scuChalkSearchEnable() - SCU CDM Chalkmark Search Enable
scuChalkSearchDisable() - SCU CDM Chalkmark Search Disable
scuChalkSearchReset() - SCU CDM Chalkmark Search Reset

scuCurChalkSet() - SCU Current Chalkmarks Set
scuCurChalkRead() - SCU Current Chalkmark Read
scuCdmCntlRegWrite() - SCU Write to CDM Control Register
scuCdmStatRegRead() - SCU Read from CDM Status Register
scuCdmCntlRegRead() - SCU Read from CDM Control Register
scuCdmStaleCadrRegRead() - SCU CDM Stale CADR Reg. Read
scuCtcIntTimProg() - SCU CTC Interval Timer Program
scuCtcRealTimProg() - SCU CTC Real-time Timer Program
scuCtcIntTimRead() - SCU CTC Interval Timer Register Read
scuCtcStartRegRead() - SCU CTC Start Register Read
scuCtcStopRegRead() - SCU CTC Stop Register Read
scuCtcStatRegRead() - SCU CTC Status Register Read
scuCellXferToCore() - SCU Cell Transfer to Core Memory
scuWriteXCells() - Write X Cells into the Core Memory
scuCellXferFromCore() - SCU Cell Transfer from Core Memory
scuReadXCells() - Read X Cells from the Core Memory
scuCellXferToFromCore() - SCU Cell Transfer to/from Core
scuIvcSetup() - SCU IVC Setup
scuIvcSet() - SCU Single IVC Setup
scuIvcInit() - SCU IVC Initialization
scuCoreCellWrite() - SCU Core Cell Write
scuCoreCellRead() - SCU Core Cell Read
scuVerify() - SCU Verify

```
STATUS scuCellCpyRamToSb
   (CELL_STRUCT * ramCell, UINT16 cellIndex)
STATUS scuCellCpySbToRam
   (UINT16 cellIndex, CELL_STRUCT * ramCell)
STATUS scu64BCellDecode
   (UINT16 cellToDec[32], UINT16 pattern,
   CELL_STRUCT * pCell, UCHAR cntlFlag)
STATUS scuMap8To7
   (UCHAR * byte)
STATUS scuVerCorePat
   (UINT16 corePattern)
void scuCifCntlRegWrite
   (UINT16 controlWord)
UINT16 scuCifStatRegRead
   (void)
UINT16 scuCifCntlRegRead
   (void)
STATUS scuCifWrPoll
   (void)
STATUS scuCifRdPoll
   (void)
STATUS scuCifRdWrPoll
   (void)
STATUS scuWivcLutSetup
   (IVC_STRUCT *ivcData, UINT16 numIvcs, UINT16 ivc,
   UCHAR wCsid, UCHAR hpWFifoEn, UCHAR lpWFifoEn)
```

STATUS scuWivcLutSet
(UINT16 ivc, UCHAR wCsid, UCHAR hpWFifoEn,
UCHAR lpWFifoEn)
STATUS scuWivcLutInit
(IVC_STRUCT * ivcData, UINT16 numIvcs)
STATUS scuRivcLutSetup
(IVC_STRUCT * ivcData, UINT16 numIvcs, UINT16 ivc,
UCHAR rCsid, UCHAR rFifoEn)
STATUS scuRivcLutSet
(UINT16 ivc, UCHAR rCsid, UCHAR rFifoEn)
STATUS scuRivcLutInit
(IVC_STRUCT * ivcData, UINT16 numIvcs)
STATUS scuCifWivcFifoRead
(UINT16 * wIvc, UINT16 * wCadr, char fifoSelect)
STATUS scuCifRivcFifoRead
(UINT16 * rIvc)
UINT32 scuCifHPWFifoStatRead
(void)
UINT32 scuCifLPWFifoStatRead
(void)
UINT32 scuCifRFifoStatRead
(void)
UINT16 scuFlmCmbfCountRead
(void)
void scuFlmCntlRegWrite
(UINT16 controlWord)
UINT16 scuFlmStatRegRead
(void)
UINT16 scuFlmCntlRegRead
(void)
STATUS scuFlmFreeListInit
(UINT16 * cadrArray, UINT16 numCadrs, UINT16 singleCadr)
STATUS scuFlmFLPush
(UINT16 cadr)
STATUS scuFlmFLArrayPush
(UINT16 * cadrArray, UINT16 numCadrs)
STATUS scuFlmFLCInit
(void)
void scuFlmHalt
(void)
void scuFlmHaltClear
(void)
void scuFlmReset
(void)
void scuFlmPrime
(void)
STATUS scuDbmSetup
(IVC_STRUCT * ivcData, UINT16 numIvcs, UINT16 ivc,
UINT16 dbmEntry, UCHAR castBit)
STATUS scuDbmSet

```
   (UINT16 ivc, UINT16 dbmEntry, UCHAR castBit)
STATUS scuDbmInit
   (IVC_STRUCT * ivcData, UINT16 numIvcs)
UINT32 scuDbmCapture
   (UINT16 ivc)
UINT32 scuWDbmCapture
   (UINT16 cadr)
void scuChalkSearchEnable
   (void)
void scuChalkSearchDisable
   (void)
void scuChalkSearchReset
   (void)
STATUS scuCurChalkSet
   (UINT16 curChalkmark)
UINT16 scuCurChalkRead
   (void)
void scuCdmCntlRegWrite
   (UINT16 controlWord)
UINT16 scuCdmStatRegRead
   (void)
UINT16 scuCdmCntlRegRead
   (void)
UINT16 scuCdmStaleCadrRegRead
   (void)
STATUS scuCtcIntTimProg
   (UINT16 startIvc, UCHAR startFlag, UINT16 stopIvc,
   UCHAR stopFlag)
STATUS scuCtcRealTimProg
   (void)
UINT32 scuCtcIntTimRead
   (void)
UINT32 scuCtcStartRegRead
   (void)
UINT32 scuCtcStopRegRead
   (void)
UINT32 scuCtcStatRegRead
   (void)
STATUS scuCellXferToCore
   (UINT16 sbCellIndex, UINT16 cifCntlFlag, UCHAR wId,
   UCHAR wLog, UINT16 wIvc)
void scuWriteXCells
   (UINT16 wIvc, UINT32 x)
STATUS scuCellXferFromCore
   (UINT16 sbCellIndex, UINT16 cifCntlFlag, UCHAR rId,
   UCHAR rLog, UINT16 rIvc, UINT16 rCadr)
void scuReadXCells
   (UINT16 rIvc, UINT16 rCadr, UINT32 x)
STATUS scuCellXferToFromCore
   (UINT16 wSbCellIndex, UINT16 rSbCellIndex,
```

```
  UINT16 cifCntlFlag, UCHAR wId, UCHAR wLog, UINT16 wIvc,
  UCHAR rId, UCHAR rLog, UINT16 rIvc, UINT16 rCadr)
STATUS scuIvcSetup
  (IVC_STRUCT * ivcData, UINT16 numIvcs, UINT16 ivc,
  UCHAR wCsid, UCHAR hpWFifoEn, UCHAR lpWFifoEn,
  UCHAR rCsid, UCHAR rFifoEn, UINT16 dbm, UCHAR cast)
STATUS scuIvcSet
  (UINT16 ivc, UCHAR wCsid, UCHAR hpWFifoEn,
  UCHAR lpWFifoEn, UCHAR rCsid, UCHAR rFifoEn, UINT16 dbm,
  UCHAR cast)
STATUS scuIvcInit
  (IVC_STRUCT * ivcData, UINT16 numIvcs)
STATUS scuCoreCellWrite
  (UINT16 sbCellIndex, CELL_STRUCT * ramCell, UCHAR wId,
  UCHAR wLog)
STATUS scuCoreCellRead
  (UINT16 sbCellIndex, CELL_STRUCT * ramCell, UCHAR rId,
  UCHAR rLog, UINT16 rIvc, UINT16 rCadr)
STATUS scuVerify
  (void)
```

DESCRIPTION

This module contains a set of Switch Control Unit (SCU) driver functions that provide a simple interface allowing a user to manipulate and control SCU functions with simple one line function calls. This module provides a low-level HW Interface layer upon which higher layers of SW can build. This module does not perform full-blown SCU initialization but this is instead done by scuInitLib which calls upon the lower level functions in this module.

WARNINGS

Some of these functions can initialize SCU HW or perform operations on the SCU which should not be called during normal switch operation (eg. scuFlmInit). Users invoking any functions in this library while the switch is running should have a comprehensive knowledge of the SCU.

INCLUDE FILES

scu.h, scuDrv.h,

SEE ALSO

scuInitLib, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization” scuHVDiag(1) **CREDITNET SWITCH SW MANUAL** scuHVDiag(1)

NAME

scuHVDiag - SCU HW Verification Diagnostics Library

SYNOPSIS

scuHVDS1TogCheck() - SCU HV DIP Switch Toggle Check
scuHVPMResetTest() - SCU HV PM Reset Test
scuHVIdLogTest() - SCU HV ID/LOG Test
scuHVEncDecTest() - SCU HV Encoded/Decoder Test

```
STATUS scuHVDS1TogCheck
    (void)
STATUS scuHVPMResetTest
    (void)
STATUS scuHVIdLogTest
    (void)
STATUS scuHVEncDecTest
    (void)
```

DESCRIPTION

This module contains a set of tests that perform HW verification diagnostics on the SCU. These tests are not performed during System Initialization when the switch boots. These tests should only be called for debugging purposes from the VxWorks Shell. Most of the tests require assistance from the user to jumper various signals. Some tests only provide certain stimuli to the HW and the user is required to verify correct HW responses using a scope or logic analyzer. Users are normally prompted to perform these activities via messages printed to the VxWorks Shell.

N.T. - Not Tested yet (HW most likely not ready for test).
PE - Parity Error.

WARNINGS

These tests severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

INCLUDE FILES

scu.h, scuDrv.h, scuHVDiag.h, scuSDiag.h, scuInit.h

SEE ALSO

cuDrv, scuSDiag, scuInit, scuDrv, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics"

NAME

scuInit - SCU Initialization Library Library

SYNOPSIS

scuInitCtl() - SCU Initialization Control
scuDefDSInit() - SCU Default Data Structure Initialization
scuHwInit() - SCU Hw Initialization Initialization
scuPifInit() - SCU Processor Interface Initialization
scuCtcInit() - SCU Cell Time Counter Initialization
scuCdmInit() - SCU Cell Distribution Manager Initialization
scuFlmInit() - SCU FLM Initialization
scuCifInit() - SCU Cell Interface Initialization

```
STATUS scuInitCtl
   (void)
void scuDefDSInit
   (void)
STATUS scuHwInit
   (void)
void scuPifInit
   (void)
void scuCtcInit
   (void)
void scuCdmInit
   (void)
STATUS scuFlmInit
   (void)
void scuCifInit
   (void)
```

DESCRIPTION

This module contains a set functions for initializing SCU hardware and data structures. A global SCU configuration data structure named scuConfig is initialized by scuDefDSInit(). This data structure is used by the hardware initialization functions to initialize the individual blocks of the SCU. Function scuHwInit() calls all the individual hardware block initialization functions to do complete SCU hardwire initialization. All of the hardware initialization functions are built on top of the functions in scuDrv which is the low level hardware interface library for the SCU. The highest level function in this module, scuInitCtl(), calls both scuDefDSInit() and scuHwInit() to perform complete SCU hardware and software data structure initialization.

WARNING

The functions in this module should not be invoked during normal switch operation because they can obliterate the current state of the switch.

INCLUDE FILES

scu.h, scuDrv.h scuInit.h

SEE ALSO

scuDrv "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuIntLib - SCU Interrupt Library

SYNOPSIS

scuIntInit() - SCU Interrupt Initialization
scuIntMon() - SCU Interrupt Monitor
scuFanAlarmISR() - SCU Fan Alarm ISR
scuACAlarmISR() - SCU AC Alarm ISR
scuPSAlarmISR() - SCU Power Supply Alarm ISR
scuCoreClkAlarmISR() - SCU Core Memory Clock Alarm ISR
scuFlmEmptyISR() - SCU FLM Free List Empty ISR
scuFlmFailISR() - SCU FLM Free List Fail ISR
scuCoreParErrISR() - SCU Core Memory Parity Error ISR
scuWFifoHPAlFullISR() - SCU HP W_IVC FIFO Almost Full ISR
scuWFifoLPAlFullISR() - SCU LP W_IVC FIFO Almost Full ISR
scuStaleCadrISR() - SCU Stale CADR ISR
scuCifRcvCrc8ErrISR() - SCU CIF Receive CRC8 Error ISR
scuCbtErrISR() - SCU Cell Bus-tap Error ISR
scuRFifoAlFullISR() - SCU R_IVC FIFO Almost Full ISR
scuFlmWrapISR() - SCU FLM Free List Wrap ISR
scuItTimCompISR() - SCU Interval Timer Timing Complete ISR
scuFlmFullISR() - SCU FLM Full ISR
scuWFifoHPMidISR() - SCU HP W_IVC FIFO Mid-range ISR
scuWFifoLPMidISR() - SCU LP W_IVC FIFO Mid-range ISR
scuRFifoMidISR() - SCU R_IVC FIFO Mid-range ISR

```
STATUS scuIntInit
   (void)
void scuIntMon
   (void)
void scuFanAlarmISR
   (void)
void scuACAlarmISR
   (void)
void scuPSAlarmISR
   (void)
void scuCoreClkAlarmISR
   (void)
void scuFlmEmptyISR
   (void)
void scuFlmFailISR
   (void)
void scuCoreParErrISR
   (void)
void scuWFifoHPAlFullISR
   (void)
```

void scuWFifoLPAlFullISR
(void)
void scuStaleCadrISR
(void)
void scuCifRcvCrc8ErrISR
(void)
void scuCbtErrISR
(void)
void scuRFifoAlFullISR
(void)
void scuFlmWrapISR
(void)
void scuItTimCompISR
(void)
void scuFlmFullISR
(void)
void scuWFifoHPMidISR
(void)
void scuWFifoLPMidISR
(void)
void scuRFifoMidISR
(void)

DESCRIPTION

This module contains ISRs (Interrupt Service Routines) to service SCU interrupts as well as functions to initialize SCU interrupts and to report interrupt statistics to the Switch User Interface. ISRs in this module simply increment a counter in an array of SCU interrupt counters. These counters hold at there maximum value if they reach them. These counters are initialized by scuIntInit() and are cleared periodically by scuIntMon() when Switch Interrupt Monitorring is enabled. These ISRs are very simple and are not intended to fullfil the needs of application SW.

The SCU interrupt monitorring function, scuIntMon(), is invoked by the switch interrupt monitorring function, intCntrMon(), to examine the SCU interrupt counters, reset them to 0 and report the non-zero counters to the switch User Interface.

WARNINGS

If application SW replaces these ISR they should still maintain the interrupt counters in the replacement ISRs so that
interrupt reporting to the Switch UI will still be accurate.

INCLUDE FILES

scuIntLib.h, scuDrv.h, scu.h

SEE ALSO

scuDrv, intMonLib, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

NAME

scuSDiag - SCU Start-up Diagnostics Library

SYNOPSIS

scuSDiagCtl() - SCU Start-up Diagnostics Controller
scuSCifFifoTest() - SCU CIF FIFO Test
scuSFifoStatCheck() - SCU CIF FIFO Status Check
scuSCifDataPathTest() - SCU CIF Data Path Test
scuSRegCheck() - SCU Register Bit Check
scuSSBByteCheck() - SCU Shared Buffer Byte Check
scuSCifCrc8Test() - SCU CIF CRC8 Test
scuSCdmMonFATest() - SCU CDM Monitoring and FA Test
scuSFlmBasicTest() - SCU Flm Basic Functionality Test
scuSFlmIntegTest() - SCU FLM Integrity Test
scuSCdmCmUpdTest() - SCU CDM Chalkmark Updating Test
scuSChalkmarkCheck() - SCU Chalkmark Check
scuSCdmCmSrchTest() - SCU CDM Chalkmark Search Test
scuSCtcIt1Test() - SCU CTC Interval Timer Test 1
scuSCtcIt2Test() - SCU CTC Interval Timer Test 2

```
STATUS scuSDiagCtl
   (UINT32 scuTestMask)
STATUS scuSCifFifoTest
   (void)
void scuSFifoStatCheck
   (INT32 fifo, UINT32 value, STATUS * result,
   UINT16 fifoCnt, UINT32 substate)
STATUS scuSCifDataPathTest
   (void)
void scuSRegCheck
   (INT32 reg, UINT16 bit, UINT16 bitState,
   STATUS * result, UINT32 substate)
void scuSSBByteCheck
   (char * sbImage, STATUS * result, UINT32 substate)
STATUS scuSCifCrc8Test
   (void)
STATUS scuSCdmMonFATest
   (void)
STATUS scuSFlmBasicTest
   (void)
STATUS scuSFlmIntegTest
   (void)
STATUS scuSCdmCmUpdTest
   (void)
void scuSChalkmarkCheck
   (UINT16 startCadr, UINT16 stopCadr, UINT16 uniCM,
```

```
    UINT16 multiCM, STATUS * result, UINT32 substate)
STATUS scuSCdmCmSrchTest
    (void)
STATUS scuSCtcIt1Test
    (void)
STATUS scuSCtcIt2Test
    (void)
```

DESCRIPTION

This module contains a set of tests that perform start-up diagnostics on the SCU when the switch boots. It also contains scuSDiagCtl(), the SCU Diagnostics Controller, which can execute any combination of SCU tests in scuSDiag and scuSMDiag based on a test mask passed to it. The individual tests can also be called for debugging purposes from the VxWorks Shell. These tests can test memories and FIFOs in one of two ways depending on whether the macro DEBUG is defined during compilation. If DEBUG is defined then these test will use a small set of fixed patterns for the purpose of debugging ease and fault isolation. If DEBUG is not defined, which should be the case when these tests are run during System Initialization, many sets of random patterns are used in order to give greater test coverage than in the debugging case.

All of the tests in this module are self-contained in that they do not require an additional SCU initialization before they are run since they do all the necessary initialization themselves.

This module informs users of status and results by sending messages to the switch User Interface. It can also print more detailed information out to the VxWorks Shell if DIP Switch 2 on the GPU is turned on. This information printed to the shell can be very useful for HW debugging.

WARNINGS

These tests severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

INCLUDE FILES

scuSDiag.h, scuSMDiag.h, scu.h, scuDrv.h, scuInit.h scuSCoreTest.h

SEE ALSO

scuDrv, scuSMDiag, scuInit, scuSCoreTest, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

NAME

scuSMDiag - SCU Start-up Memory Diagnostics Library

SYNOPSIS

scuSMCifRegTest() - SCU SM CIF Register Test
scuSMSharBuffTest() - SCU SM Diag. Shared Buffer Test
scuSMWivcLutTest() - SCU SM Diag. W_IVC Lookup Table Test
scuSMRivcLutTest() - SCU SM Diag. R_IVC Lookup Table Test
scuSMFlmRegTest() - SCU SM FLM Register Test
scuSMFlmRamStatTest() - SCU SM Diag. FLM RAM and Status Test
scuSMCdmRegTest() - SCU SM CDM Register Test
scuSMCdmConCapTest() - SCU SM CDM Configuration/Capture Test
scuSMDbmRamTest() - SCU SM DBM RAM Test
scuSMWDbmRamTest() - SCU SM Working DBM RAM Test
scuSMCtcRegTest() - SCU SM CTC Register Test
scuSMPbusRegTest() - SCU SM P-Bus Register Test

```
STATUS scuSMCifRegTest
   (void)
STATUS scuSMSharBuffTest
   (void)
STATUS scuSMWivcLutTest
   (void)
STATUS scuSMRivcLutTest
   (void)
STATUS scuSMFlmRegTest
   (void)
STATUS scuSMFlmRamStatTest
   (void)
STATUS scuSMCdmRegTest
   (void)
STATUS scuSMCdmConCapTest
   (void)
STATUS scuSMDbmRamTest
   (void)
STATUS scuSMWDbmRamTest
   (void)
STATUS scuSMCtcRegTest
   (void)
STATUS scuSMPbusRegTest
   (void)
```

DESCRIPTION

This module contains a set of tests that perform start-up diagnostics on SCU memories and registers when the switch boots. Any subset of these tests can be executed by scuSDiagCtl() in scuSDiag by passing it a test mask. The tests can also be called for debugging purposes from the VxWorks Shell. These tests can test memories and FIFOs in one of two ways depending on whether the macro DEBUG is defined during compilation. If DEBUG is defined then these test will use a small set of fixed patterns for the purpose of debugging ease and fault isolation. If DEBUG is not defined, which should be the case when these tests are run during System Initialization, many sets of random patterns are used in order to give greater test coverage than in the debugging case.

All of the tests in this module are self-contained in that they do not require an additional SCU initialization before they are run since they do all the necessary initialization themselves.

This module informs users of status and results by sending messages to the switch User Interface. It can also print more detailed information out to the VxWorks Shell if DIP Switch 2 on the GPU is turned on. This information printed to the shell can be very useful for HW debugging.

WARNINGS

These tests severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

INCLUDE FILES

scuSDiag.h, scuSMDiag.h, scuInit.h, scu.h, scuDrv.h

SEE ALSO

scuDrv, scuSDiag, scuInit, scuSCoreTest, scuSDiagCtl(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

ispEnValCalc(2) **CREDITNET SWITCH SW MANUAL** **ispEnValCalc(2)**

NAME

ispEnValCalc() - Calculate the ISP Programming Register Value

SYNOPSIS

```
INT16 ispEnValCalc
   (
   char  scuOrCore,  /* 0 = SCU, 1 = Core Memory Card */
   char  deviceNum   /* SCU or Core device number.    */
   )
```

DESCRIPTION

This function calculates the value of the curEnVal global variable which is written to the SCU's Program/Test Enable register to enable the desired Lattice for programming.

RETURNS

Value of curEnVal to be written to the ISP Programming Register

SEE ALSO

ispUtil, ispProgram(), ispRead(), ispVerify(), "Design Specification - CreditNet Switch - Switch Control Unit(SCU)"

ispProgram(2) CREDITNET SWITCH SW MANUAL ispProgram(2)

NAME

ispProgram() - Programs an ISP Lattice Device

SYNOPSIS

```
STATUS ispProgram
   (
   char *  fileName,   /* Name of source Jedec file on WS. */
   char    scuOrCore,  /* 0 = SCU, 1 = Core Memory Card    */
   char    deviceNum   /* SCU or Core device number.       */
   )
```

DESCRIPTION

This function programs and verifies an SCU or Core Memory Lattice device with the given Jedec file. Devices are specified by deviceNum and scuOrCore chooses between SCU (0) or Core Memory (1) cards.

RETURNS

OK, or ERROR if the device can not be programmed and verified successfully.

WARNINGS

This function should not be invoked on a switch while it is operating normally because it can put switch out of commision depending on which device is being selected.

SEE ALSO

ispUtil, ispVerify(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)” ispRead(2) **CREDITNET SWITCH SW MANUAL** ispRead(2)

NAME

ispRead() - Reads an ISP Lattice Device

SYNOPSIS

```
STATUS ispRead
   (
   char *  fileName,   /* Name of desination ISP file on WS. */
   char    scuOrCore,  /* 0 = SCU, 1 = Core Memory Card      */
   char    deviceNum   /* SCU or Core device number.         */
   )
```

DESCRIPTION

This function reads the serial ISP data from an SCU or Core Memory Lattice device and places it in the given filename. Devices are specified by deviceNum and scuOrCore chooses between SCU (0) or Core Memory (1) cards.

RETURNS

OK, or ERROR if the device can not be read successfully.

WARNINGS

This function should not be invoked on a switch while it i operating normally because it can put switch out of commission depending on which device is being selected.

SEE ALSO

ispUtil, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

NAME

ispVerify() - Verifies an ISP Lattice Device

SYNOPSIS

```
STATUS ispVerify
  (
  char *  fileName,   /* Name of source Jedec file on WS. */
  char    scuOrCore,  /* 0 = SCU, 1 = Core Memory Card    */
  char    deviceNum   /* SCU or Core device number.       */
  )
```

DESCRIPTION

This function verifies an SCU or Core Memory Lattice device against the given Jedec file. Devices are specified by deviceNum and scuOrCore chooses between SCU (0) or Core Memory(1) cards.

RETURNS

OK, or ERROR if the device can not be verified successfully.

WARNINGS

This function should not be invoked on a switch while it is operating normally because it can put switch out of commision depending on which device is being selected.

SEE ALSO

ispUtil, ispProgram(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

NAME

scu64BCellDecode() - SCU 64-Byte Cell Decode to 56-Bytes

SYNOPSIS

```
STATUS scu64BCellDecode
  (
  UINT16          cellToDec[32],   /* Core ready cell to decode. */
  UINT16          pattern,         /* Source cell pattern.      */
  CELL_STRUCT     *pCell,          /* Decoded cell.             */
  UCHAR           cntlFlag         /* Control flag.             */
   )
```

DESCRIPTION

This function decodes 64-bytes cells as they would appear in the Core Memory into 56-byte cells which include a cell header, ivc and payload. If users set the control flag to GEN_FROM_PATTERN (0x1) then this function first generates a 64-byte encoded cell which repeats ‘pattern’ throughout. If this flag is not set this function decodes the cell pointed to by pCellToDec. This function places the decoded cell in the data structure pointed to by pCell. The results are place in this data structure so that it can easily be passed on to scuCellCpyRamToSb().

RETURNS

OK, or ERROR if an invalid Core Memory pattern is encountered.

WARNINGS

This function does not return the crc8 decoded from the cell since this can not be placed directly in the Core by the CIF anyway. The CIF always calculates the crc8 for a cell and replaces that found in the Shared Buffer.

SEE ALSO

scuDrv, scuMap8To7(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuACAlarmISR() - SCU AC Alarm ISR

SYNOPSIS

void scuACAlarmISR (void)

DESCRIPTION

This function services the AC alarm interrupt. This interrupt indicates that a problem has been sensed in the AC power that the switch is using. For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

WARNINGS

SEE ALSO

scuIntLib, scuIntMon(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

NAME

scuCbtErrISR() - SCU Cell Bus-tap Error ISR

SYNOPSIS

void scuCbtErrISR (void)

DESCRIPTION

This function services the cell bus-tap error interrupt. This interrupt indicates that the cell bus-tap on the Terminator Card has detected 16 (?) CRC8 errors on the cell data passing through the cell bus-tap.

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

scuCdmCntlRegRead(2) CREDITNET SWITCH SW MANUAL scuCdmCntlRegRead(2)

NAME

scuCdmCntlRegRead() - SCU Read from CDM Control Register

SYNOPSIS

UINT16 scuCdmCntlRegRead (void)

DESCRIPTION

This function reads a 16-bit word from the CDM's control register, masks out any unused (or reserved) bits, and returns this to the calling function.

RETURNS

CDM Control Register contents.

SEE ALSO

scuDrv, scuCdmCntlRegWrite(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuCdmCntlRegWrite(2)   CREDITNET SWITCH SW MANUAL   scuCdmCntlRegWrite(2)

NAME

scuCdmCntlRegWrite() - SCU Write to CDM Control Register

SYNOPSIS

```
void scuCdmCntlRegWrite
  (
  UINT16  controlWord  /* Word to write to CDM Control Reg. */
  )
```

DESCRIPTION

This function writes a 16-bit control word to the CDM's control register without masking any bits.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuCdmCntlRegRead(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCdmInit() - SCU Cell Distribution Manager Initialization

SYNOPSIS

void scuCdmInit (void)

DESCRIPTION

This function initializes CDM's DBM and wDBM RAM as well as the current chalkmarks and the CDM Control Register. Both Current chalkmarks are initialized to 0. DBM RAM is initialized based on scuConfig and wDBM RAM is initialized base on IVC 0 in scuConfig.

WARNINGS

This function should not be invoked during normal switch operation since it can obliterate the current state of the switch.

RETURNS

Nothing. Absoltively, posolutely nothing.

SEE ALSO

scuInit, scuHwInit(), scuDefDSInit(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

scuCdmStaleCadrRegRead(2) CREDITNET SWITCH SW MANUAL scuCdmStaleCadrRegRead(2)

NAME

scuCdmStaleCadrRegRead() - SCU CDM Stale CADR Reg. Read

SYNOPSIS

UINT16 scuCdmStaleCadrRegRead (void)

DESCRIPTION

This function reads a cell address from the CDM's stale CADR register which contains the last CADR found stale during a chalkmark search.

RETURNS

CDM Stale CADR Register contents.

SEE ALSO

scuDrv, scuChalkSearchReset(), scuChalkSearchEnable(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCdmStatRegRead() - SCU Read from CDM Status Register

SYNOPSIS

UINT16 scuCdmStatRegRead (void)

DESCRIPTION

This function reads a 16-bit status word from the CDM's status register, masks out any unused (or reserved) bits, and returns this to the calling function.

RETURNS

CDM Status Register contents.

SEE ALSO

scuDrv, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuCellCpyRamToSb(2)   CREDITNET SWITCH SW MANUAL   scuCellCpyRamToSb(2)

NAME

scuCellCpyRamToSb() - Copy Cell GPU RAM->Shared Buffer

SYNOPSIS

```
STATUS scuCellCpyRamToSb
    (
    CELL_STRUCT *  ramCell,   /* Source cell from which to copy.       */
    UINT16         cellIndex  /* Destination cell index in Shar. Buf.  */
    )
```

DESCRIPTION

This function copies a 55-byte cell from the GPU's RAM to the Shared Buffer in the CIF. Calling functions must pass it a pointer to the cell (its data structure) and the destination cell index into the Shared Buffer. The function takes the parts of the cell, which are the cell payload, cell header and cell ivc, and places them in their corresponding locations in the cell buffer in the Shared Memory. It also initializes the location where the cell's CRC8 will be appended, by the CIF, to 0.

RETURNS

OK (always).

SEE ALSO

scuDrv, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCellCpySbToRam() - Copy Cell Shared Buffer->GPU RAM

SYNOPSIS

```
STATUS scuCellCpySbToRam
  (
  UINT16        cellIndex,  /* Source cell index in shar. buf. */
  CELL_STRUCT *  ramCell    /* Destination cell.               */
  )
```

DESCRIPTION

This function copies a 55-byte cell from the CIF's Share Buffer to the GPU's RAM. Calling functions must pass it a pointer to the cell data structure at which to copy the cell and the cell index into the Shared Buffer. The function takes the parts of the cell from the Shared Buffer and places them in their corresponding fields in the cell data data structure which are the cell payload, cell header and cell ivc.

RETURNS

OK (always).

SEE ALSO

scuDrv, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCellXferFromCore() - SCU Cell Transfer from Core Memory

SYNOPSIS

```
STATUS scuCellXferFromCore
  (
  UINT16  sbCellIndex,  /* Cell Index into Shared Buffer.    */
  UINT16  cifCntlFlag,  /* Word to write to CIF Control Reg. */
  UCHAR   rId,          /* R_ID value used for read cycle.   */
  UCHAR   rLog,         /* R_LOG value used for read cycle.  */
  UINT16  rIvc,         /* R_IVC value used for read cycle.  */
  UINT16  rCadr         /* CADR indicating Core cell to read */
  )
```

DESCRIPTION

This function reads a cell from the Core Memory into the CIF's Shared Buffer at the location indicated by sbCellIndex. The read cycle is carried out according to the parameters passed from the calling function which includes a 16 bit word indicating the value to write to the CIF Control register. The RD_ONE_CELL bit in the control register must still be set in order to cause the read cycle. This function polls until the RD_ONE_CELL bit clears in the CIF Control register before returning to the calling function. This ensures that the CIF has completed the read cycle when this function returns.

RETURNS

OK, or ERROR if unable to read the cell.

SEE ALSO

scuDrv, scuCellXferToCore(), scuCellXferToFromCore(), scuCoreCellRead(), scuReadXCells(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCellXferToCore() - SCU Cell Transfer to Core Memory

SYNOPSIS

```
STATUS scuCellXferToCore
  (
  UINT16  sbCellIndex,  /* Cell Index into Shared Buffer .   */
  UINT16  cifCntlFlag,  /* Word to write to CIF Control Reg. */
  UCHAR   wId,          /* W_ID value used for write cycle.  */
  UCHAR   wLog,         /* W_LOG value used for write cycle. */
  UINT16  wIvc          /* W_IVC used for write cycle.       */
  )
```

DESCRIPTION

This function writes a cell in the CIF's Shared Buffer, indicated by sbCellIndex, into the Core Memory. The write cycle is carried out according to the parameters passed from the calling function which includes a 16-bit word indicating the value to write to the CIF Control register. The WR_ONE_CELL bit in the control register must still be set in order to cause the write cycle. This function polls until the WR_ONE_CELL bit clears in the CIF Control register before returning to the calling function. This ensures that the CIF has completed the write cycle when this function returns.

RETURNS

OK, or ERROR if unable to write the cell.

SEE ALSO

scuDrv, scuCellXferFromCore(), scuCellXferToFromCore(), scuWriteXCells(), scuCoreCellWrite(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCellXferToFromCore() - SCU Cell Transfer to/from Core

SYNOPSIS

```
STATUS scuCellXferToFromCore
  (
  UINT16  wSbCellIndex,  /* Cell Index into Shared Buffer.    */
  UINT16  rSbCellIndex,  /* Cell Index into Shared Buffer.    */
  UINT16  cifCntlFlag,   /* Word to write to CIF Control Reg. */
  UCHAR   wId,           /* W_ID value used for write cycle.  */
  UCHAR   wLog,          /* W_LOG value used for write cycle. */
  UINT16  wIvc,          /* W_IVC value used for write cycle. */
  UCHAR   rId,           /* R_ID value used for read cycle.   */
  UCHAR   rLog,          /* R_LOG value used for read cycle.  */
  UINT16  rIvc,          /* R_IVC value used for read cycle.  */
  UINT16  rCadr          /* CADR indicating Core cell to read */
  )
```

DESCRIPTION

This function reads a cell from the Core Memory into the CIF's Shared Buffer and writes a cell from the CIF's Shared Buffer to the Core Memory. The cell to be written into the Core Memory is indicated by wSbCellIndex and the cell to be read from the Core Memory is placed in the Shared Buffer at the location indicated by rSbCellIndex. The read and write cycles are carried out according to the parameters passed from the calling function which includes a 16-bit word indicating the value to write to the CIF Control register. The RD_ONE_CELL and WR_ONE_CELL bits in the control register must still be set in order to cause the read cycle. This function polls until the RD_ONE_CELL and WR_ONE_CELL bits are clear in the CIF Control register before returning to the calling function. This ensures that the CIF has completed the read and write cycle when this function returns.

RETURNS

OK, or ERROR if unable to perform the read or write cycle.

SEE ALSO

scuDrv, scuCellXferFromCore(), scuCellXferToCore(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuChalkSearchDisable() - SCU CDM Chalkmark Search Disable

SYNOPSIS

void scuChalkSearchDisable (void)

DESCRIPTION

This function deasserts the chalkmark search enable bit in the CDM control register to disable the CDM chalkmark search for stale cell addresses.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuChalkSearchEnable(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuChalkSearchEnable(2) CREDITNET SWITCH SW MANUAL scuChalkSearchEnable(2)

NAME

scuChalkSearchEnable() - SCU CDM Chalkmark Search Enable

SYNOPSIS

void scuChalkSearchEnable (void)

DESCRIPTION

This function asserts the chalkmark search enable bit in the CDM control register to enable the CDM to perform the chalkmark search for stale cell addresses. The search will start looking for stale CADR at CADR 0 if the search has just been reset or will continue looking at the CADR after the last stale CADR found if a stale CADR was indeed just found.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuChalkSearchReset(), scuChalkSearchDisable(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuChalkSearchReset() - SCU CDM Chalkmark Search Reset

SYNOPSIS

void scuChalkSearchReset (void)

DESCRIPTION

This function clears the chalkmark search enable bit in the CDM control register as well as resetting the CM Search address and clearing the CM Alarm and CM Done flags in the CDM Status register. A subsequent call to scuChalkSearchEnable() after this function will cause the chalkmark search to be started at CADR 0.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuChalkSearchEnable(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuCifCntlRegRead() - SCU Read from CIF Control Register

SYNOPSIS

UINT16 scuCifCntlRegRead (void)

DESCRIPTION

This function reads a 16-bit word from the CIF's control register, masks out all unused (or reserved) bits, and returns it to the calling function.

RETURNS

CIF Control Register contents.

SEE ALSO

scuDrv, scuCifCntlRegWrite(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCifCntlRegWrite() - SCU Write to CIF Control Register

SYNOPSIS

```
void scuCifCntlRegWrite
   (
   UINT16  controlWord  /* Word to write to CIF Control Reg. */
   )
```

DESCRIPTION

This function writes a 16-bit control word to the CIF's control register without masking any bits.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuCifCntlRegRead(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuCifHPWFifoStatRead(2) CREDITNET SWITCH SW MANUAL scuCifHPWFifoStatRead(2)

NAME

scuCifHPWFifoStatRead() - SCU CIF HP W_IVC FIFO Status Read

SYNOPSIS

UINT32 scuCifHPWFifoStatRead (void)

DESCRIPTION

This function returns the status of the CIF's high priority W_IVC FIFO which is in the range of 0 to 3. The status values have the following meanings:

0 - FIFO Empty
1 - FIFO Almost Empty (less than 128 elements)
2 - FIFO Almost Full (within 128 elements of full)
3 - FIFO Full (contains 4K elements)

RETURNS

High priority W_IVC FIFO status.

SEE ALSO

scuDrv, scuCifLPWFifoStatRead(), scuCifRFifoStatRead(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCifInit() - SCU Cell Interface Initialization

SYNOPSIS

void scuCifInit (void)

DESCRIPTION

This function initializes CIF's R_IVC and W_IVC Lookup Tables, based on the global SCU Configuration DS 'scuConfig'. It also initializes the Shared Buffer (filling it with 0) and the CIF Control Register.

WARNINGS

This function should not be invoked during normal switch operation since it can obliterate the current state of the switch.

RETURNS

Nothing. Absoltively, posolutely nothing.

SEE ALSO

scuInit, scuHwInit(), scuDefDSInit(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuCifLPWFifoStatRead(2) CREDITNET SWITCH SW MANUAL scuCifLPWFifoStatRead(2)

NAME

scuCifLPWFifoStatRead() - SCU CIF LP W_IVC FIFO Status Read

SYNOPSIS

UINT32 scuCifLPWFifoStatRead (void)

DESCRIPTION

This function returns the status of the CIF's low priority W_IVC FIFO which is in the range of 0 to 3. The status values have the following meanings:

0 - FIFO Empty
1 - FIFO Almost Empty (less than 128 elements)
2 - FIFO Almost Full (within 128 elements of full)
3 - FIFO Full (contains 4K elements)

RETURNS

Low priority W_IVC FIFO status.

SEE ALSO

scuDrv, scuCifHPWFifoStatRead(), scuCifRFifoStatRead(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuCifRFifoStatRead() - SCU CIF R_IVC FIFO Status Read

SYNOPSIS

UINT32 scuCifRFifoStatRead (void)

DESCRIPTION

This function returns the status of the CIF's R_IVC FIFO which is in the range of 0 to 3. The status values have the following meanings:

0 - FIFO Empty
1 - FIFO Almost Empty (less than 128 elements)
2 - FIFO Almost Full (within 128 elements of full)
3 - FIFO Full (contains 4K elements)

RETURNS

R_IVC FIFO status.

SEE ALSO

scuDrv, scuCifHPWFifoStatRead(), scuCifLPWFifoStatRead(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuCifRcvCrc8ErrISR(2) CREDITNET SWITCH SW MANUAL scuCifRcvCrc8ErrISR(2)

NAME

scuCifRcvCrc8ErrISR() - SCU CIF Receive CRC8 Error ISR

SYNOPSIS

void scuCifRcvCrc8ErrISR (void)

DESCRIPTION

This function services the CIF's receive CRC8 error interrupt. This interrupt indicates that the CIF detected a CRC8 error in one of the cells that the CIF was reading from the Core Memory.

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

NAME

scuCifRdPoll() - SCU CIF Read Cycle Poll

SYNOPSIS

STATUS scuCifRdPoll (void)

DESCRIPTION

This function polls the CIF Control register until the RD_ONE_CELL bit has been cleared which indicates that the CIF is not busy doing a read cycle. It has the ability to timeout this polling if the RD_ONE_CELL bit is not cleared in a reasonable amount of time. It is not necessary to call this if you are using scuCellXferFromCore() to perform a write cycle since it does this polling before it is done executing.

RETURNS

OK when the CIF is no longer performing a read cycle or ERROR if this polling times out.

SEE ALSO

scuDrv, scuCellXferFromCore(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuCifRdWrPoll() - SCU CIF Read/Write Cycle Poll

SYNOPSIS

STATUS scuCifRdWrPoll (void)

DESCRIPTION

This function polls the CIF Control register until the RD_ONE_CELL and WR_ONE_CELL bits are cleared indicating that the CIF is not busy doing a read or write cycle. It has the ability to timeout this polling if the bits are not cleared in a reasonable amount of time. It is not necessary to call this if you are using scuCellXferToFromCore() to perform a write/read cycle since it does this polling before it is done executing.

RETURNS

OK when the CIF is no longer performing a read or write cycle or ERROR if this polling times out.

SEE ALSO

scuDrv, scuCellXferToFromCore(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuCifRivcFifoRead() - SCU CIF R_IVC FIFO Read

SYNOPSIS

```
STATUS scuCifRivcFifoRead
   (
   UINT16 *  rIvc  /* Location to store IVC read from FIFO. */
   )
```

DESCRIPTION

This function reads the R_IVC from the head of the R_IVC FIFO.

RETURNS

OK, or ERROR if the R_IVC FIFO is empty.

SEE ALSO

scuDrv, scuCifWivcFifoRead(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuCifStatRegRead(2) CREDITNET SWITCH SW MANUAL scuCifStatRegRead(2)

NAME

scuCifStatRegRead() - SCU Read from CIF Status Register

SYNOPSIS

UINT16 scuCifStatRegRead (void)

DESCRIPTION

This function reads a 16-bit status word from the CIF's status register, masks out all unused (or reserved) bits, and returns it to the calling function.

RETURNS

CIF Status Register contents.

SEE ALSO

scuDrv, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCifWivcFifoRead() - SCU CIF W_IVC FIFO Read

SYNOPSIS

```
STATUS scuCifWivcFifoRead
   (
   UINT16 *  wIvc,       /* Location to store IVC read from FIFO.  */
   UINT16 *  wCadr,      /* Location to store CADR read from FIFO. */
   char      fifoSelect  /* 0 = high priority; 1 = low priority.   */
   )
```

DESCRIPTION

This function reads the W_IVC and W_CADR from the head of the W_IVC FIFO indicated by the W_IVC FIFO select flag.

RETURNS

OK, or ERROR if the FIFO to be read is empty or an invalid fifoSelect flag is specified (ie. not 0 or 1).

SEE ALSO

scuDrv, scuCifRivcFifoRead(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization” scuCifWrPoll(2) CREDITNET SWITCH SW MANUAL scuCifWrPoll(2)

NAME

scuCifWrPoll() - SCU CIF Write Cycle Poll

SYNOPSIS

STATUS scuCifWrPoll (void)

DESCRIPTION

This function polls the CIF Control register until the WR_ONE_CELL bit has been cleared which indicates that the CIF is not busy doing a write cycle. It has the ability to timeout this polling if the WR_ONE_CELL bit is not cleared in a reasonable amount of time. It is not necessary to call this if you are using scuCellXferToCore() to perform a write cycle since it does this polling before it is done execut-ing.

RETURNS

OK when the CIF is no longer performing a write cycle or ERROR if polling times out.

SEE ALSO

scuDrv, scuCellXferToCore(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuCoreCellRead() - SCU Core Cell Read

SYNOPSIS

```
STATUS scuCoreCellRead
    (
    UINT16         sbCellIndex,  /* Cell Index into Shared Buffer.    */
    CELL_STRUCT *  ramCell,      /* Destination Cell in GPU RAM.      */
    UCHAR          rId,          /* R_ID value used for read cycle.   */
    UCHAR          rLog,         /* R_LOG value used for read cycle.  */
    UINT16         rIvc,         /* R_IVC value used for read cycle.  */
    UINT16         rCadr         /* R_CADR from which to read.        */
    )
```

DESCRIPTION

This function reads a cell from the Core Memory and places it in the GPU RAM. It first calls scuCellXferFromCore to perform a read cycle with the RD_ONE_CELL, RD_CYC_VALID and CRC8_CALC_ENABLE bits in the CIF control register set. This reads the cell from CADR rCadr using IVC rIVC and places it in the Shared Buffer in the Cell Buffer indicated by sbCellIndex. It then calls function scuCellCpySbToRam() to take the cell from the Shared Buffer and place it in GPU RAM at the location pointed to by ramCell.

RETURNS

OK, or ERROR if unable to read the cell.

SEE ALSO

scuDrv, scuCellCpySbToRam(), scuCellXferFromCore(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuCoreCellWrite() - SCU Core Cell Write

SYNOPSIS

```
STATUS scuCoreCellWrite
   (
   UINT16         sbCellIndex,  /* Cell Index into Shared Buffer.    */
   CELL_STRUCT *  ramCell,      /* Source cell in GPU RAM.           */
   UCHAR          wId,          /* W_ID value used for write cycle.  */
   UCHAR          wLog          /* W_LOG value used for write cycle. */
   )
```

DESCRIPTION

This function writes a cell in GPU RAM into the Core Memory. It calls function scuCellCpyRamToSb() to take the cell pointed to by ramCell and place it in the CIF Shared Buffer at the Cell Buffer indicated by sbCellIndex. It then calls scuCellXferToCore() to perform the write cycle with the WR_ONE_CELL, WR_CYC_VALID and CRC8_CALC_ENABLE bits in the CIF control register set and using a W_IVC equal to that found in ramCell.

RETURNS

OK, or ERROR if unable to write the cell.

SEE ALSO

scuDrv, scuCellCpyRamToSb(), scuCellXferToCore(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuCoreClkAlarmISR() - SCU Core Memory Clock Alarm ISR

SYNOPSIS

void scuCoreClkAlarmISR (void)

DESCRIPTION

This function services the Core Memory clock signal alarm interrupt. This interrupt indicates that the Core Memory has detected some improper clock signals which could result in the loss of all data in the Core Memory and its inability to accept more cells.

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

scuCoreParErrISR(2) CREDITNET SWITCH SW MANUAL scuCoreParErrISR(2)

NAME

scuCoreParErrISR() - SCU Core Memory Parity Error ISR

SYNOPSIS

void scuCoreParErrISR (void)

DESCRIPTION

This function services the Core Memory parity error interrupt. This interrupt indicates that the Core Memory has detected 16 parity errors in the CADRs passing through its CADR pipeline. (This could mean that as many as 32 cell buffers in the Core Memory have been corrupted.)

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

NAME

scuCtcInit() - SCU Cell Time Counter Initialization

SYNOPSIS

void scuCtcInit (void)

DESCRIPTION

This function initializes the CTC to place it in RT (Real- time) counter mode. Users must call scuCtcIntTimProg() in order to use the CTC for IVC-based interval timing.

RETURNS

Nothing. Absoltively, posolutely nothing.

SEE ALSO

sculnit, scuCtcRealTimProg(), scuCtcIntTimProg(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCtcIntTimProg() - SCU CTC Interval Timer Program

SYNOPSIS

```
STATUS scuCtcIntTimProg
   (
   UINT16  startIvc,   /* IVC written to start register. */
   UCHAR   startFlag,  /* 1 = R_IVC; 0 = W_IVC.          */
   UINT16  stopIvc,    /* IVC written to stop register.  */
   UCHAR   stopFlag    /* 1 = R_IVC; 0 = W_IVC.          */
   )
```

DESCRIPTION

This function places the CTC in Intefval Timing (Cell Timer) mode then programs the start and stop registers.

RETURNS

OK, or ERROR if the CTC is performing a Cell Timing or an invalid IVC has been specified by the calling function.

SEE ALSO

scuDrv, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

scuCtcIntTimRead(2) CREDITNET SWITCH SW MANUAL scuCtcIntTimRead(2)

NAME

scuCtcIntTimRead() - SCU CTC Interval Timer Register Read

SYNOPSIS

UINT32 scuCtcIntTimRead (void)

DESCRIPTION

This function reads the value of the CTC's interval timer register.

RETURNS

CTC interval timer register contents.

SEE ALSO

scuDrv, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

scuCtcRealTimProg(2) CREDITNET SWITCH SW MANUAL scuCtcRealTimProg(2)

NAME

scuCtcRealTimProg() - SCU CTC Real-time Timer Program

SYNOPSIS

STATUS scuCtcRealTimProg (void)

DESCRIPTION

This function places the CTC in Real-Time Timing mode. In this mode, the CTC increments the Timer register once per cell period no matter what. When the timer reaches its maximum value of 0x3ffffff it raps around to zero and continues incrementing from zero.

RETURNS

OK, or error if the CTC is busy performing a interval timing.

SEE ALSO

scuDrv, scuCtcIntTimProg “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

scuCtcStartRegRead(2) CREDITNET SWITCH SW MANUAL scuCtcStartRegRead(2)

NAME

scuCtcStartRegRead() - SCU CTC Start Register Read

SYNOPSIS

UINT32 scuCtcStartRegRead (void)

DESCRIPTION

This function reads the value of the CTC's Start register. It only reads the following bits:

```
Bits   Description
----   -----------
0-13   Start IVC
 14    Start Mode (0 = W_IVC; 1 = R_IVC)
 15    CTC Mode (0 = Cell Time; 1 = Real Time)
```

Even though the CTC Status bits appear in this register, they are not read here but are instead read by scuCtcStatRegRead().

RETURNS

CTC start register contents.

SEE ALSO

scuDrv, scuCtcStopRegRead(), scuCtcStatRegRead(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCtcStatRegRead() - SCU CTC Status Register Read

SYNOPSIS

UINT32 scuCtcStatRegRead (void)

DESCRIPTION

This function reads the value of the CTC's Status bits which can be read from both the CTC Start and Stop registers (in bits 16 - 18). This function returns the status bits as follows:

Bits	Description
----	-----------
0	Found Start Condition
1	Found Stop Condition
2	Searching for Start Condition.

RETURNS

CTC status bits.

SEE ALSO

scuDrv, scuCtcStartRegRead(), scuCtcStopRegRead(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCtcStopRegRead() - SCU CTC Stop Register Read

SYNOPSIS

UINT32 scuCtcStopRegRead (void)

DESCRIPTION

This function reads the value of the CTC's Stop register. It only reads the following bits:

```
Bits   Description
----   -----------
0-13   Stop IVC
 14    Stop Mode (0 = W_IVC; 1 = R_IVC)
```

Even though the CTC Status bits appear in this register, they are not read here but are instead read by scuCtcStatRegRead().

RETURNS

CTC stop register contents.

SEE ALSO

scuDrv, scuCtcStartRegRead(), scuCtcStatRegRead(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuCurChalkRead(2) **CREDITNET SWITCH SW MANUAL** **scuCurChalkRead(2)**

NAME

scuCurChalkRead() - SCU Current Chalkmark Read

SYNOPSIS

UINT16 scuCurChalkRead (void)

DESCRIPTION

This function reads the current chalkmarks from the CDM's current chalkmark register. It returns a value in the range 0 to 3 with bit 0 indicating the stare of current CM 0 and bit 1 indicating the current CM 1.

Returned Value	CM 0	CM 1
0	0	0
1	0	1
2	1	0
3	1	1

RETURNS

((current CM 1 << 1) | current CM 0).

SEE ALSO

scuDrv, scuCurChalkSet(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuCurChalkSet() - SCU Current Chalkmarks Set

SYNOPSIS

```
STATUS scuCurChalkSet
  (
  UINT16  curChalkmark  /* Current chalkmark values to set. */
  )
```

DESCRIPTION

This function sets the CDM's two current chalkmarks. Users pass this function a 16-bit integer with use current CM 0 in bit 0 and current CM 1 in bit 1. SCU chalkmark (CM) HW will current CM 0 for chalkmark updating and searching if the casting bit (for a CADR or IVC) is set to 0. SCU chalkmark HW will use CM 1 if the casting bit (for a CADR or IVC) is set to 1.

RETURNS

OK, or ERROR if the value passed as curchalkmark is greater than 3.

SEE ALSO

scuDrv, scuCurChalkSet(), scuDbmSet(), scuDbmInit(), scuDbmCapture(), scuWDbmCapture(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuDbmCapture() - SCU Distribution Bitmap Capture

SYNOPSIS

```
UINT32 scuDbmCapture
   (
   UINT16  ivc  /* IVC of DBM to be captured.          */
   )
```

DESCRIPTION

This function returns the casting bit and DBM for the IVC specified by IVC.

RETURNS

A 32 bit integer with ivc's casting bit in bit 18 and ivc's DBM in bits 15 - 0 or ERROR if invalid IVC has been specified .

SEE ALSO

scuDrv, scuWDbmCapture(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuDbmInit() - SCU Distribution Bitmap Initialization

SYNOPSIS

```
STATUS scuDbmInit
   (
   IVC_STRUCT *  ivcData,  /* IVC configuration data.       */
   UINT16        numIvcs   /* Number of IVCs to initialize. */
   )
```

DESCRIPTION

This function sets the DBMs and casting bits for numIvcs IVCs based on the IVC configuration data pointed to by ivc-Data.

RETURNS

OK, or ERROR if numIvcs is greater than the maximum IVC value.

SEE ALSO

scuDrv, scuInit, scuCdmInit(), scuDbmSet(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuDbmSet() - SCU Distribution Bitmap Entry Set

SYNOPSIS

```
STATUS scuDbmSet
   (
   UINT16  ivc,       /* IVC.              */
   UINT16  dbmEntry,  /* Distribution bitmap */
   UCHAR   castBit    /* Casting bit       */
   )
```

DESCRIPTION

This function initialize the DBM and casting bit for IVC ivc to dbmEntry and castBit respectively.

RETURNS

OK, or ERROR if invalid input parameters have been specified.

SEE ALSO

scuDrv, scuDbmInit(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization” scuDbmSetup(2) CREDITNET SWITCH SW MANUAL scuDbmSetup(2)

NAME

scuDbmSetup() - SCU Distribution Bitmap Setup

SYNOPSIS

```
STATUS scuDbmSetup
   (
   IVC_STRUCT *  ivcData,   /* IVC configuration data.            */
   UINT16        numIvcs,   /* Number of IVCs to initialize.      */
   UINT16        ivc,       /* IVC for single IVC init.           */
   UINT16        dbmEntry,  /* Distribution bitmap single IVC init */
   UCHAR         castBit    /* Casting bit for single IVC init     */
   )
```

DESCRIPTION

This function sets up a DBM and casting bit for one or a set of IVCs. If numIvcs is 1 then IVC ivc will have its DBM set to dbm and its casting bit set to castBit. If numIvcs is greater than 1 then numIvcs are initialize based on IVC configuration data pointed to by ivcData. If numIvcs is greater than the maximum IVC value then this function returns ERROR.

WARNING

This function is not the primary function(s) for DBM initialization. Users should use scuDbmSet() to initialize the DBM and casting bit for a single IVC or use scuDbmInit to initialize the DBMs and casting bits for many IVCs.

RETURNS

OK, or ERROR if invalid input parameters have been specified.

SEE ALSO

scuDrv, scuInit, scuCdmInit(), scuDbmSet(), scuDbmInit(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization” scuDefDSInit(2) CREDITNET SWITCH SW MANUAL scuDefDSInit(2)

NAME

scuDefDSInit() - SCU Default Data Structure Initialization

SYNOPSIS

void scuDefDSInit (void)

DESCRIPTION

This function initializes the SCU Configuration Data structure scuConfig to default values. It initialize all IVCs to the following values:

Parameter	Default Value
W_CSID	0x0
R_CSID	0x0
DBM	0x0
LP W_FIFO En.	Disabled (0x1)
HP W_FIFO En.	Disabled (0x1)
R_FIFO En.	Disabled (0x1)
Cast Bit	Unicast (0x0)

This function also initializes the CRM's global Core Memory BF threshold to 0x32f0 (80% full) and port thresholds for ports 1 - 15 to 0. The port threshold for the SCU to are initialized to the following:

Theshold (for SCU)	Default Value
Low BF Thresh.	0x01000
Mid BF Thresh.	0x1000
Up BF Thresh.	0x3C00
RPI Thresh.	0x3C00

RETURNS

Nothing. Absoltively, posolutely nothing.

WARNING

This function should not be invoked during normal switch operation since it can obliterate the current state of the switch.

SEE ALSO

scuInit, scuInitCtl(), scuHwInit(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuFanAlarmISR(2) CREDITNET SWITCH SW MANUAL scuFanAlarmISR(2)

NAME

scuFanAlarmISR() - SCU Fan Alarm ISR

SYNOPSIS

void scuFanAlarmISR (void)

DESCRIPTION

This function services the fan alarm interrupt. This inter rupt indicates that one or more of the switch's fans have failed and that switch components could be damaged due to overheating.

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

scuFlmCmbfCountRead(2) CREDITNET SWITCH SW MANUAL scuFlmCmbfCountRead(2)

NAME

scuFlmCmbfCountRead() - SCU FLM CMBF Counter Read

SYNOPSIS

UINT16 scuFlmCmbfCountRead (void)

DESCRIPTION

This function reads FLM's Core Memory Buffer Fill (CMBF) Counter.

RETURNS

FLM's CMBF Counter contents.

SEE ALSO

scuDrv, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuFlmCntlRegRead(2) CREDITNET SWITCH SW MANUAL scuFlmCntlRegRead(2)

NAME

scuFlmCntlRegRead() - SCU Read from FLM Control Register

SYNOPSIS

UINT16 scuFlmCntlRegRead (void)

DESCRIPTION

This function reads a 16-bit word from the FLM's control register, masks out all unused (or reserved) bit and returns it to the calling function.

RETURNS

FLM Control Register contents.

SEE ALSO

scuDrv, scuFlmCntlRegWrite(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuFlmCntlRegWrite() - SCU Write to FLM Control Register

SYNOPSIS

```
void scuFlmCntlRegWrite
  (
  UINT16  controlWord  /* Word to write to FLM Control Reg. */
  )
```

DESCRIPTION

This function writes a 16-bit control word to the FLM's control register without masking any bits.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuFlmCntlRegRead(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuFlmEmptyISR() - SCU FLM Free List Empty ISR

SYNOPSIS

void scuFlmEmptyISR (void)

DESCRIPTION

This function services the FLM Free List empty interrupt. This interrupt indicates that the Free List does not contain any CADRs for use during write cycles. (Cell write cycles carried out by all PMs and the SCU are nulled when the FLM Free List is empty.)

For now, this function simply increments its counter unless it is at its maximum value and reads the FLM Status Register to remove the interrupt.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

scuFlmFLArrayPush(2) CREDITNET SWITCH SW MANUAL scuFlmFLArrayPush(2)

NAME

scuFlmFLArrayPush() - SCU FLM Free List Array Push

SYNOPSIS

```
STATUS scuFlmFLArrayPush
   (
   UINT16 *  cadrArray,  /* Ptr to CADRs to place on Free List.     */
   UINT16    numCadrs    /* Number of CADRs to initialize.          */
   )
```

DESCRIPTION

This function pushes numCadrs CADRs pointed to by cadrArray onto the FLM's Free CADR List. numCadrs should be greater than 0 and less than (MAX_NUM_CADR) or else ERROR will be returned.

RETURNS

OK, or ERROR if numCadrs is greater than the maximum CADR or if an invalid CADR is specified in cadrArray.

SEE ALSO

scuDrv, scuFlmFLPush(), scuFlmFLCInit(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuFlmFLCInit() - SCU FLM Free List Complete Initialization

SYNOPSIS

STATUS scuFlmFLCInit (void)

DESCRIPTION

This function initializes the FLM Free List with the complete set of CADRs from 0 to to (MAX_NUM_CADR - 1).

RETURNS

OK.

SEE ALSO

scuDrv, scuInit, scuFlmInit(), scuFlmFLPush(), scuFlmFLArrayPush(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuFlmFLPush() - SCU FLM Free List Push

SYNOPSIS

```
STATUS scuFlmFLPush
   (
   UINT16  cadr  /* CADR to push onto Free List. */
   )
```

DESCRIPTION

This function pushes cadr onto the FLM's Free CADR List.

RETURNS

OK, or ERROR if invalid CADR has been specified.

SEE ALSO

scuDrv, scuFlmFLArrayPush(), scuFlmFLCInit(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuFlmFailISR() - SCU FLM Free List Fail ISR

SYNOPSIS

void scuFlmFailISR (void)

DESCRIPTION

This function services the FLM Free List fail interrupt. This interrupt indicates that the FLM's head and tail pointers do not correspond to the number of cells in the Core Memory according to the CMBF count.

For now, this function simply increments its counter unless it is at its maximum value and reads the FLM Status Register to remove the interrupt.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

NAME

scuFlmFreeListInit() - SCU FLM Free List Initialization

SYNOPSIS

```
STATUS scuFlmFreeListInit
   (
   UINT16 *  cadrArray,  /* Ptr to CADRs to place on Free List.     */
   UINT16    numCadrs,   /* Number of CADRs to initialize.          */
   UINT16    singleCadr  /* Cadr for single IVC init.               */
   )
```

DESCRIPTION

This function places one or a set of CADRs on the FLM's Free CADR depending on the value of num-Cadrs. If numCadrs is 1 then it pushes only singleCadr onto the Free List. If num- Cadrs is greater than 1 and less than the maximum CADR value, then numCadrs pointed to by cadrArray are pushed onto the Free List. If numCadrs is equal to the maximum CADR value, then CADR 0 to maximum CADR value minus 1 are pushed onto the Free List. The following summarizes this functions behavior:

numCadrs	Free List (FL) Initialization
1	Put only singleCadr on the FL.
>1 & <MAX	Put numCadr CADR from cadrArray on FL
MAX	Put CADR 0 - (MAX - 1) on FL

WARNING

This is not the main function(s) for use to initialize the FLM Free CADR List. Most users should instead use scuFlmFLPush() to push one CADR onto the Free List, scuFlmFLArrayPush() to push a set of CADR onto the Free List and scuFlmFLCInit() to initialize the Free List with the entire range of CADRs from 0 to (MAX_CADR - 1).

RETURNS

OK, or ERROR if invalid input parameters have been specified.

SEE ALSO

scuDrv, scuFlmFLPush(), scuFlmFLArrayPush(), scuFlm- FLCInit(), "Design Specification - Cred-itNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuFlmFullISR(2) CREDITNET SWITCH SW MANUAL scuFlmFullISR(2)

NAME

scuFlmFullISR() - SCU FLM Full ISR

SYNOPSIS

void scuFlmFullISR (void)

DESCRIPTION

This function services the FLM's Free List Full interrupt. This interrupt indicates that the FLM contains all 16K CADRs.

For now, this function simply increments its counter unless it is at its maximum value and reads the FLM Status Register to remove the interrupt.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

scuFlmHalt(2) CREDITNET SWITCH SW MANUAL scuFlmHalt(2)

NAME

scuFlmHalt() - SCU FLM Halt

SYNOPSIS

void scuFlmHalt (void)

DESCRIPTION

This function asserts the FLM halt bit in the FLM control register. This prevents the FLM from popping W_CADRs off of the Free List, asserting FLM fail, asserting FLM wrapped and incrementing the CMBF Counter.

WARNING

Calling this function will freeze the FLM and render it unable to output CADRs. It should not be called when the switch is running normally.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuFlmHaltClear() "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuFlmHaltClear(2) **CREDITNET SWITCH SW MANUAL** **scuFlmHaltClear(2)**

NAME

scuFlmHaltClear() - SCU FLM Halt Clear

SYNOPSIS

void scuFlmHaltClear (void)

DESCRIPTION

This function deasserts the FLM halt bit in the FLM control register. This enables the FLM to pop W_CADRs off of the Free List, assert FLM fail, assert FLM wrapped and increment the CMBF Counter. This function must be called after performing an FLM Reset since that places the FLM in halt after resetting it.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuInit, scuFlmInit(), scuFlmReset(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

scuFlmInit(2) **CREDITNET SWITCH SW MANUAL** **scuFlmInit(2)**

NAME

scuFlmInit() - SCU FLM Initialization

SYNOPSIS

STATUS scuFlmInit (void)

DESCRIPTION

This function performs complete initialization of the FLM including a reset, placing the complete set of CADRs from 0 to 16K-1 on the Free CADR List, clearing the FLM Halt and priming the first 4 CADRs into the FLM feedback register pipeline.

WARNINGS

This function should not be invoked during normal switch operation since it can obliterate the current state of the switch.

RETURNS

OK, or ERROR if the FLM Free CADR List RAM can not be initialized.

SEE ALSO

scuInit, scuHwInit(), scuFlmReset(), scuFlmHaltClear(), scuFlmFLCInit(), scuFlmPrime(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuFlmPrime() - SCU FLM Prime

SYNOPSIS

void scuFlmPrime (void)

DESCRIPTION

This function primes the FLM Free CADR pipeline (between the FLM Free List RAM and the backplane) with the next four CADRs to come from the FLM RAM. This is necessary to flush out old addresses from the pipeline when the FLM is initialized.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuInit, scuFlmInit(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization” scuFlmReset(2) **CREDITNET SWITCH SW MANUAL** scuFlmReset(2)

NAME

scuFlmReset() - SCU FLM Reset

SYNOPSIS

void scuFlmReset (void)

DESCRIPTION

This function resets the FLM emptying the Free CADR List and placing it in halt.

WARNING

Calling this function will freeze the FLM effectively erase all CADRs in it rendering it unable to output CADRs. It should not be called when the switch is running normally.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuInit, scuFlmInit(), scuFlmHaltClear(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

scuFlmStatRegRead(2) CREDITNET SWITCH SW MANUAL scuFlmStatRegRead(2)

NAME

scuFlmStatRegRead() - SCU Read from FLM Status Register

SYNOPSIS

UINT16 scuFlmStatRegRead (void)

DESCRIPTION

This function reads a 16-bit status word from the FLM's status register, masks out all unused (or reserved) bits and returns it to the calling function.

RETURNS

FLM Status Register contents.

SEE ALSO

scuDrv, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuFlmWrapISR(2) CREDITNET SWITCH SW MANUAL scuFlmWrapISR(2)

NAME

scuFlmWrapISR() - SCU FLM Free List Wrap ISR

SYNOPSIS

void scuFlmWrapISR (void)

DESCRIPTION

This function services the FLM Free List wrap interrupt. Every time this interrupt occurs, it indicates that the FLM has cycled through 16K CADRs.

For now, this function simply increments its counter unless it is at its maximum value and reads the FLM Status Register to remove the interrupt.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

NAME

scuHVDS1TogCheck() - SCU HV DIP Switch Toggle Check

SYNOPSIS

STATUS scuHVDS1TogCheck (void)

DESCRIPTION

This function checks to see if DIP Switch 1 has been toggled.

RETURNS

OK if DS1 is toggled or ERROR if it times out.

SEE ALSO

scuHVDiag

scuHVEncDecTest(2) **CREDITNET SWITCH SW MANUAL** **scuHVEncDecTest(2)**

NAME

scuHVEncDecTest() - SCU HV Encoded/Decoder Test

SYNOPSIS

STATUS scuHVEncDecTest (void)

DESCRIPTION

This function verifies that the CIF's encoder and decoder circuits correctly encode and decode between 56-byte unencoded cells to 64-byte encoded cells. Users will need to connect a logic analyzer up to various points in the CIF Data Path to verify the encoded version of the cell going to and coming from the Core Memory. It provides the user with the normal and encoded version of a cell so that they can be compared with the data seen moving through the CIF Data Path.

RETURNS

OK.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

SEE ALSO

scuHVDiag, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics"

NAME

scuHVIdLogTest() - SCU HV ID/LOG Test

SYNOPSIS

STATUS scuHVIdLogTest (void)

DESCRIPTION

This function verifies that the integrity of the W_ID, W_LOG, R_ID and R_LOG registers and that the values in these registers are driven onto the CCB during the SCU's CCB slot. These signals should probably be checked using a logic analyzer.

RETURNS

OK.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

SEE ALSO

scuHVDiag, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics"

scuHVPMResetTest(2) CREDITNET SWITCH SW MANUAL scuHVPMResetTest(2)

NAME

scuHVPMResetTest() - SCU HV PM Reset Test

SYNOPSIS

STATUS scuHVPMResetTest (void)

DESCRIPTION

This function verifies that the SCU correctly TDMs the reset signal presented to the Port Modules based on the contents of the SCU's PM Reset Register.

RETURNS

OK.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

SEE ALSO

scuHVDiag, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics"

scuHwInit(2) **CREDITNET SWITCH SW MANUAL** **scuHwInit(2)**

NAME

scuHwInit() - SCU Hw Initialization Initialization

SYNOPSIS

STATUS scuHwInit (void)

DESCRIPTION

This function initializes all SCU HW blocks including the FLM, CDM, PIF and CIF. These blocks are initialized based on the global SCU Data Structure scuConfig.

WARNING

This function should not be invoked during normal switch operation since it can obliterate the current state of the switch.

RETURNS

OK, or ERROR if an error is encountered initializing any of the SCU HW blocks.

SEE ALSO

sculnit, scuDefDSInit(), scuInitCtl(), scuPifInit(), scuCifInit(), scuCdmInit(), scuFlmInit(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuInitCtl() - SCU Initialization Control

SYNOPSIS

STATUS scuInitCtl (void)

DESCRIPTION

This function controls the process of entire SCU initialization. It allocates memory for the global SCU configuration data structure, scuConfig and initializes it to default values. It then initializes the SCU HW based on scuConfig.

RETURNS

OK, or ERROR if a fault is encountered initializing SCU HW or allocating memory for scuConfig.

WARNING

This function should not be invoked during normal switch operation since it can obliterate the current state of the switch.

SEE ALSO

scuInit, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

scuIntInit(2) CREDITNET SWITCH SW MANUAL scuIntInit(2)

NAME

scuIntInit() - SCU Interrupt Initialization

SYNOPSIS

STATUS scuIntInit (void)

DESCRIPTION

This function connect the ISRs in scuIntLib for the SCU interrupts to their corresponding interrup vectors (except for the interrupt from the PM) and initializes all SCU interrupt counters to 0.

RETURNS

OK or ERROR if a problem occurs connnecting interrupts.

SEE ALSO

scuIntLib, scuIntLib, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)" scuIntMon(2) **CREDITNET SWITCH SW MANUAL** scuIntMon(2)

NAME

scuIntMon() - SCU Interrupt Monitor

SYNOPSIS

void scuIntMon (void)

DESCRIPTION

This function monitors SCU Interrupt counters by checking to see if they are non-zero, sending their values to the UI then clearing the counters back to zero. The interrupt counters are read and zeroed when all interrupts are locked (except NMI) to avoid the possibility of the counters being incremented by their ISRs after being read and before being cleared.

RETURNS

Nothing.

SEE ALSO

scuIntLib, intCntrMon().

scuItTimCompISR(2) CREDITNET SWITCH SW MANUAL scuItTimCompISR(2)

NAME

scuItTimCompISR() - SCU Interval Timer Timing Complete ISR

SYNOPSIS

void scuItTimCompISR (void)

DESCRIPTION

This function services the CTC's Interval Timer timing complete interrupt. This interrupt indicates that the CTC has completed timing an interval and the results are available in the Interval Timer register.

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

scuIvcInit(2) CREDITNET SWITCH SW MANUAL scuIvcInit(2)

NAME

scuIvcInit() - SCU IVC Initialization

SYNOPSIS

```
STATUS scuIvcInit
   (
   IVC_STRUCT *  ivcData,  /* IVC configuration data.       */
   UINT16        numIvcs   /* Number of IVCs to initialize. */
   )
```

DESCRIPTION

This function completely initializes numIvcs IVCs within the SCU based on the IVC configuration data pointed to by ivc-Data.

RETURNS

OK, or ERROR if either scuRivcLutSetup(), scuWivcLutSetup() or scuDbmSetup() return ERROR.

SEE ALSO

scuDrv, scuRivcLutSetup(), scuWivcLutSetup(), scuDbmSetup(), scuIvcSet(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuIvcSet() - SCU Single IVC Setup

SYNOPSIS

```
STATUS scuIvcSet
  (
  UINT16  ivc,        /* IVC for single IVC init.                    */
  UCHAR   wCsid,      /* W_CSID for single IVC init                  */
  UCHAR   hpWFifoEn,  /* High pri. W_IVC FIFO enable for single IVC */
  UCHAR   lpWFifoEn,  /* Low pri. W_IVC FIFO enable for single IVC  */
  UCHAR   rCsid,      /* R_CSID for single IVC init                  */
  UCHAR   rFifoEn,    /* R_IVC FIFO enable for single IVC            */
  UINT16  dbm,        /* Distribution bitmap single IVC init         */
  UCHAR   cast        /* Casting bit for single IVC                  */
  )
```

DESCRIPTION

This function completely initilizes one IVC within the SCU. It initializes IVC ivc based on the parameters following ivc (ie. wCsid, hpWFifoEn...).

RETURNS

OK, or ERROR if either scuRivcLutSet(), scuWivcLutSet() or scuDbmSet() return ERROR.

SEE ALSO

scuDrv, scuRivcLutSet(), scuWivcLutSet(), scuDbmSet(), scuIvcInit(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)” scuIvcSetup(2) **CREDITNET SWITCH SW MANUAL** scuIvcSetup(2)

NAME

scuIvcSetup() - SCU IVC Setup

SYNOPSIS

```
STATUS scuIvcSetup
   (
   IVC_STRUCT *  ivcData,    /* IVC configuration data.                   */
   UINT16       numIvcs,    /* Number of IVCs to initialize.             */
   UINT16       ivc,        /* IVC for single IVC init.                  */
   UCHAR        wCsid,      /* W_CSID for single IVC init                */
   UCHAR        hpWFifoEn,  /* High pri. W_IVC FIFO enable for single IVC */
   UCHAR        lpWFifoEn,  /* Low pri. W_IVC FIFO enable for single IVC  */
   UCHAR        rCsid,      /* R_CSID for single IVC init                */
   UCHAR        rFifoEn,    /* R_IVC FIFO enable for single IVC          */
   UINT16       dbm,        /* Distribution bitmap single IVC init       */
   UCHAR        cast        /* Casting bit for single IVC                */
   )
```

DESCRIPTION

This function sets one or a set of IVCs by calling functions scuRivcLutSetup(), scuWivcLutSetup and scuDbmSetup. If numIvcs is 1 then IVC ivc is initialized based on the parameters following ivc in the functions parameter list. If numIvcs is greater than 1 then numIvcs are initialized based on the IVC configuration data pointed to by ivcData.

WARNING

This is not the main function(s) for use to initialize IVCs within the SCU. Most users should instead use scuIvcSet() to initialize on IVC or scuIvcInit() to initialize many IVCs.

RETURNS

OK, or ERROR if either scuRivcLutSetup(), scuWivcLutSetup() or scuDbmSetup() return ERROR.

SEE ALSO

scuDrv, scuRivcLutSetup(), scuWivcLutSetup(), scuDbmSetup(), scuIvcSet(), scuIvcInit(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuMap8To7() - SCU Map 8 TO 7 (byte)

SYNOPSIS

```
STATUS scuMap8To7
  (
  UCHAR *  byte  /* Pointer to 8-bit word to map. */
  )
```

DESCRIPTION

This function uses the map7To8Tab to map a Core ready byte (one which usually has the same number of ones and zeroes) to a 7-bit word which can be merged with other 7-bit words to produce 8-bit bytes. This is useful when decoding CreditNet Switch cells, as they would appear in the Core Memeory. The translated 7-bit word actually replaces the byte passed to the function.

RETURNS

OK, or ERROR if the byte to be mapped is not a valid Core Memory pattern.

SEE ALSO

scuDrv, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuPSAlarmISR() - SCU Power Supply Alarm ISR

SYNOPSIS

void scuPSAlarmISR (void)

DESCRIPTION

This function services the power supply alarm interrupt. This interrupt indicates that one or more of the switch's power supplies has failed.

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

scuPifInit(2) **CREDITNET SWITCH SW MANUAL** scuPifInit(2)

NAME

scuPifInit() - SCU Processor Interface Initialization

SYNOPSIS

void scuPifInit (void)

DESCRIPTION

This function initializes certain registers in the SCU's Processor Interface (PIF).

WARNINGS

This function should not be invoked during normal switch operation since it can obliterate the current state of the switch.

RETURNS

Nothing. Absoltively, posolutely nothing.

SEE ALSO

scuInit, scuHwInit(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuRFifoAlFullISR() - SCU R_IVC FIFO Almost Full ISR

SYNOPSIS

void scuRFifoAlFullISR (void)

DESCRIPTION

This function services the R_IVC FIFO almost full interrupt. This interrupt indicates that the R_IVC FIFO is 127 elements away from being full and therefore unable to store further information.

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

NAME

scuRFifoMidISR() - SCU R_IVC FIFO Mid-range ISR

SYNOPSIS

void scuRFifoMidISR (void)

DESCRIPTION

This function services the R_IVC FIFO mid-range interrupt. This interrupt indicates that the R_IVC FIFO contains 127 elements.

For now, this function simply increments its counter unless it is at at least its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

NAME

scuReadXCells() - Read X Cells from the Core Memory

SYNOPSIS

```
void scuReadXCells
  (
  UINT16  rIvc,   /* R_IVC value used for read cycle.             */
  UINT16  rCadr,  /* R_CADR used for the read cycle.              */
  UINT32  x       /* # cells to write.                            */
  )
```

DESCRIPTION

This function reads the cell located in in the Core Memory at R_CADR rCadr into cell buffer 0 of the SCU's shared buffer x times using a R_IVC of rIvc.

WARNING

This function has little practical use but is mainly used for debugging.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuCellXferFromCore(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuRivcLutInit() - SCU R_IVC Lookup Table Initialization

SYNOPSIS

```
STATUS scuRivcLutInit
   (
   IVC_STRUCT *  ivcData,  /* IVC configuration data.       */
   UINT16        numIvcs   /* Number of IVCs to initialize. */
   )
```

DESCRIPTION

This function sets up a R_IVC Lookup Table entry for set of IVCs. Calling functions pass it a pointer to an array of IVC data structures and the number of entries to initialize. This function returns ERROR if users set numIvcs greater than the maximum number of IVCs that the switch can support.

RETURNS

OK, or ERROR if invalid input parameters have been specified.

SEE ALSO

scuDrv, scuRivcLutSet(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

scuRivcLutSet(2) **CREDITNET SWITCH SW MANUAL** **scuRivcLutSet(2)**

NAME

scuRivcLutSet() - SCU R_IVC Lookup Table Entry Set

SYNOPSIS

```
STATUS scuRivcLutSet
   (
   UINT16  ivc,     /* IVC.                */
   UCHAR   rCsid,   /* R_CSID              */
   UCHAR   rFifoEn  /* R_IVC FIFO enable bit */
   )
```

DESCRIPTION

This function sets up a R_IVC Lookup Table entry for one IVC. IVC ivc will be initialized with a R_CSID corresponding to rCsid and a R_IVC FIFO enable bit corresponding to rFifoEn.

RETURNS

OK, or ERROR if invalid input parameters have been specified.

SEE ALSO

scuDrv, scuRivcLutInit(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuRivcLutSetup() - SCU R_IVC Lookup Table Setup

SYNOPSIS

```
STATUS scuRivcLutSetup
   (
   IVC_STRUCT *  ivcData,  /* IVC configuration data.          */
   UINT16        numIvcs,  /* Number of IVCs to initialize.    */
   UINT16        ivc,      /* IVC for single IVC init.         */
   UCHAR         rCsid,    /* R_CSID for single IVC init       */
   UCHAR         rFifoEn   /* R_IVC FIFO enable for single IVC */
   )
```

DESCRIPTION

This function sets up a R_IVC Lookup Table entry for one or a set of IVCs. If the user sets numIvcs to 1 then IVC ivc is initialized with a R_CSID of rCsid, and a R_IVC FIFO enable bit set to rFifoEn. If numIvcs is greater than 1 then numIvcs IVCs are configured in the R_IVC Lookup table according to the IVC configuration data pointed to by ivc-Data. This function returns ERROR if users set numIvcs greater than the maximum number of IVCs that the switch can support.

WARNING

This is not the main function(s) for use to initialize the R_IVC Lookup Table. Most users should instead use scuRivcLutInit() to initialize more than one entry or scuRivcLutSet() to initialize one entry in the table.

RETURNS

OK, or ERROR if invalid input parameters have been specified.

SEE ALSO

scuDrv, scuRivcLutSet(), scuRivcLutInit() “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization” scuSCdmCmSrchTest(2) **CREDITNET SWITCH SW MANUAL** scuSCdmCmSrchTest(2)

NAME

scuSCdmCmSrchTest() - SCU CDM Chalkmark Search Test

SYNOPSIS

STATUS scuSCdmCmSrchTest (void)

DESCRIPTION

This function tests the CDM to ensure that it correctly finds all CADRs with stale chalkmarks which fit the stale chalkmark criteria. It also verifies that the CDM sets the Stale Chalkmark bit and interrupts the GPU when a stale chalkmark is found. It relies on stale chalkmark ISR to build a list of stale CMs when stale the CM interrupt occurs during the test. As a result, this test will fail if per- formed with mixed-mode interrupts masked out on the GPU.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if an unexpected HW response is received.

SEE ALSO

scuSDiag, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”,

NAME

scuSCdmCmUpdTest() - SCU CDM Chalkmark Updating Test

SYNOPSIS

STATUS scuSCdmCmUpdTest (void)

DESCRIPTION

This function tests that the CDM correctly updates a CADR's wDBM with the correct current chalkmark (unicast or multicast based on the wDBMs casting bit) but only under the correct circumstances.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

This test does not focus heavily on address decoding and and related faults in the wDBM RAM but these are covered thouroughly in scuSMWDbmRamTest().

RETURNS

OK, or ERROR if an unexpected HW response is received.

SEE ALSO

scuSDiag, scuSWDbmRamTest(), scuSChalkmarkCheck(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”,

NAME

scuSCdmMonFATest() - SCU CDM Monitoring and FA Test

SYNOPSIS

STATUS scuSCdmMonFATest (void)

DESCRIPTION

This function tests the CDM's ability to monitor cell write cycles, update the DBMs (Distribution Bitmaps) and assert the Free Address (FA) signal when appropriate. It verifies that only DBM bit 0 is cleared when the SCU performs valid write cycles. This test also veifies the the FLM reclaims CADRs correctly when the CDM asserts FA.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR.

SEE ALSO

scuSDiag, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics",

NAME

scuSChalkmarkCheck() - SCU Chalkmark Check

SYNOPSIS

```
void scuSChalkmarkCheck
   (
   UINT16    startCadr,  /* CADR to start checking CMs. */
   UINT16    stopCadr,   /* CADR to stop checking CMs.  */
   UINT16    uniCM,      /* Unicast Chalkmark.          */
   UINT16    multiCM,    /* Multiicast Chalkmark.       */
   STATUS *  result,     /* Ptr to caller's result.     */
   UINT32    substate    /* Error message if error.     */
   )
```

DESCRIPTION

This function checks a range of CADRs for stale Chalkmarks (CMs). If stale CMs are found, it sets the calling functions result to ERROR and optionally sends a message to the switch UI. CMs are read by capturing the wDBM for a certain CADR. This function is very similar to a SW version of the CM search and has spots where hooks can be added to reclaim CADRs whose CMs are stale (where the printf() calls are placed).

Implementing this function reduced the need to repeat many lines of code in scuSCdmCmUpdTest().

RETURNS

Nothing but it does update the value pointed to by *result.

SEE ALSO

scuSDiag, scuSCdmCmUpdTest(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

NAME

scuSCifCrc8Test() - SCU CIF CRC8 Test

SYNOPSIS

STATUS scuSCifCrc8Test (void)

DESCRIPTION

This function tests the CIF's CRC8 generation/checking circuit to verify that it correctly calculates and checks the CRC8, which is calculated for the first 55 bytes of a cell and appended in byte 56, but only when it is enabled. It also verifies that the CRC8 interrupt and CRC8 Error bit in the CIF Status register are working correctly.

WARNINGS

This test alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR.

SEE ALSO

scuSDiag, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”,

NAME

scuSCifDataPathTest() - SCU CIF Data Path Test

SYNOPSIS

STATUS scuSCifDataPathTest (void)

DESCRIPTION

This function tests the CIF's ingress and egress data paths to/from the Core Memory. It also verifies correct functioning of the Shared Buffer's dual port interface from the Core Memory side. It tests the functioning of various bits in the CIF's Status and Control registers as well as verifying that the CIF does not transfer cells to/from the Core Memory on cycles that are invalid or have been nulled.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

This test does not cover much of the Core Memory as it only uses a few CADRs. Failures in the Core Memory however could lead to the failure of this test though so be sure to keep this in mind when debugging.

RETURNS

OK, or ERROR.

SEE ALSO

scuSDiag, scuSCoreTest(), scuSMSharBuffTest(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics",

scuSCifFifoTest(2) **CREDITNET SWITCH SW MANUAL** **scuSCifFifoTest(2)**

NAME

scuSCifFifoTest() - SCU CIF FIFO Test

SYNOPSIS

STATUS scuSCifFifoTest (void)

DESCRIPTION

This function tests the CIF's three IVC FIFOs to verify that they capture W_IVC, W_CADR and R_IVC data for IVCs that have the appropriate enable bits asserted in the R_IVC or W_IVC Lookup tables. It also verifies that the FIFOs do not cap- ture data under inapporpriate conditions such as nulled or invalid read or write cycles. This test verifies the data integrity of the FIFOs as well as the status bits and interrupts associated with them.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR.

SEE ALSO

scuSDiag, scuSFifoStatCheck, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics"

NAME

scuSCtcIt1Test() - SCU CTC Interval Timer Test 1

SYNOPSIS

STATUS scuSCtcIt1Test (void)

DESCRIPTION

This function tests the CTC's Interval Timer (IT) to verify that it only starts timing under the correct conditions for which it is programmed to start, that it continues to increment until reaching its maximum value of 0x3800000 when it does not find the programmed stop condition and that it interrupts the GPU when it reaches the maximum value and stops.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR.

SEE ALSO

scuSDiag, scuSCtcCptTest(), scuSCtcIt2Test(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics",

NAME

scuSCtcIt2Test() - SCU CTC Interval Timer Test 2

SYNOPSIS

STATUS scuSCtcIt2Test (void)

DESCRIPTION

This function tests the CTC's Interval Timer (IT) to verify that it only stops timing under the correct conditions for which it is programmed to stop, that it is capable of timing an interval which is less than one cell period long (counter = 0) and that it interrupts the GPU when and only when its timing is complete.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR.

SEE ALSO

scuSDiag, scuSCtcCptTest(), scuSCtcIt1Test(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics",

NAME

scuSDiagCtl() - SCU Start-up Diagnostics Controller

SYNOPSIS

```
STATUS scuSDiagCtl
   (
   UINT32  scuTestMask  /* Indicates which SCU tests to run. */
   )
```

DESCRIPTION

This function controls the execution of the SCU start-up diagnostics tests in scuSDiag.c and scuSMDiag.c. It invokes any subset of SCU diagnostics based on a test mask passed by the user. It also determines the results of each test it invokes and passes a messages to the switch User Interface to indicate the result. The following indicates the mapping between the bits in the test mask and the tests to be performed:

Bit	Test Run	Bit	Test Run
31	Reserved	15	Reserved
30	Reserved	14	scuSMBtcRegTest()
29	scuSCoreTest()	13	scuSMPbusRegTest()
28	scuSCtcIt2Test()	12	scuSMCtcRegTest()
27	scuSCtcIt1Test()	11	scuSMCrmRegTest()
26	scuSCtcCptTest()	10	scuSMWDbmRamTest()
25	scuSCrmProtTest()	9	scuSMDbmRamTest()
24	scuSCrmPortCntTest()	8	scuSMDbmConCapTest()
23	scuSCdmCmSrchTest()	7	scuSMCdmRegTest()
22	scuSCdmCmUpdTest()	6	scuSMFlmRamStatTest()
21	scuSFlmIntegTest()	5	scuSMFlmRegTest()
20	scuSFlmBasicTest()	4	scuSMParCrcTest()
19	scuSCdmMonFATest()	3	scuSMRivcLutTest()
18	scuSCifCrc8Test()	2	scuSMWivcLutTest()
17	scuSCifDataPathTest()	1	scuSMSharBuffTest()
16	scuSCifFifoTest()	0	scuSMCifRegTest()

WARNINGS

The tests invoked by this function may severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if any of the tests it invokes fail.

SEE ALSO

scuSDiag, scuSDiag, scuSMDiag, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics” scuSFifoStatCheck(2) **CREDITNET SWITCH SW MANUAL** scuSFifoStatCheck(2)

NAME

scuSFifoStatCheck() - SCU CIF FIFO Status Check

SYNOPSIS

```
void scuSFifoStatCheck
   (
   INT32     fifo,     /* 0=HP W_IVC; 1=LP W_IVC; 2=R_IVC.           */
   UINT32    value,    /* Status value expected (0 - 3).              */
   STATUS *  result,   /* Ptr to caller's result.                     */
   UINT16    fifoCnt,  /* Expected # entries in FIFO.                 */
   UINT32    substate  /* Error message substate if error.            */
   )
```

DESCRIPTION

This function compares the status of the selected CIF FIFO to a desired value. If the two are not the same then it sets *result to ERROR and sends out an SCU Start-up Diagnostics message to the UI with the substate value passed from the calling function. For diagnostics purposes, it also prints out the expected number of entries in the FIFO and the actual value of the FIFO's status bits.

Implementing this function allowed savings of the following lines (or similar lines) of code from scuSCifFifoTest() which would have been repeated many, many times:

```
if(scuCifHPWFifoStatRead() != ALMOST_EMPTY)
    {
        gpuUIErrorSend(
            SCU_S_DIAG_MSG | MSG_ERROR_0 | HPWF_CAPTURE_ON_INVALID_WRITE );
        result = ERROR;
        if((*GPU960_DIP_SWITCH & 0x2) == 0x2)
            printf("Expected HP W_IVC FIFO Count:%4d      Status      bits      0x%2x/n",
                (int)fifoCnt, (*fifoStatRead[fifo])());
    }
```

RETURNS

Nothing but it does alter the value pointed to by *result.

SEE ALSO

scuSDiag, scuSCifFifoTest, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

NAME

scuSFlmBasicTest() - SCU Flm Basic Functionality Test

SYNOPSIS

STATUS scuSFlmBasicTest (void)

DESCRIPTION

This function tests the FLM's ability to recycle CADRs during write cycles which are invalid and to increment and decrement the Core Memory Buffer Fill (CMBF) count when appropriate. It also verifies that the FLM behaves correctly when the FLM Halt bit is set.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR.

SEE ALSO

scuSDiag, scuFlmIntegTest(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics", scuSFlmIntegTest(2) **CREDITNET SWITCH SW MANUAL** scuSFlmIntegTest(2)

NAME

scuSFlmIntegTest() - SCU FLM Integrity Test

SYNOPSIS

STATUS scuSFlmIntegTest (void)

DESCRIPTION

This function tests CMBF count over its entire range of values and under boundary conditions. It also tests the operation of the Free List Wrap (FLW), Free List Fail (FLF) and Free List Empty signals.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR.

SEE ALSO

scuSDiag, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”,

scuSMCdmConCapTest(2) CREDITNET SWITCH SW MANUAL scuSMCdmConCapTest(2)

NAME

scuSMCdmConCapTest() - SCU SM CDM Configuration/Capture Test

SYNOPSIS

STATUS scuSMCdmConCapTest (void)

DESCRIPTION

This function tests that the CDM Distribution Bit-map (DBM) can be configured correctly, that DBM entries are copied to the working DBM (wDBM) on the appropriate cell write cycles and that both DBM and wDBM entries can be captured. It also verifies that DBMs are not configured or captured under inappropriate conditions.

This test does not perform an exhaustive test of the DBM and wDBM RAMs but this is instead left to scuSMDbmRamTest() and scuSMWDbmRamTest().

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if an unexpected HW response is received.

SEE ALSO

scuSMDiag, scuSMDbmRamTest(), scuSMWDbmRamTest(), “Design Specification - CreditNet Switch - Switch Control Unit(SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

NAME

scuSMCdmRegTest() - SCU SM CDM Register Test

SYNOPSIS

STATUS scuSMCdmRegTest (void)

DESCRIPTION

This function tests the read/write integrity of certain CDM registers and also verifies that the CDM Status register is initialized correctly after the SCU has been reset.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if a faulty register is encountered or has been initialized incorrectly by the HW.

SEE ALSO

scuSMDiag, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

NAME

scuSMCifRegTest() - SCU SM CIF Register Test

SYNOPSIS

STATUS scuSMCifRegTest (void)

DESCRIPTION

This function tests the read/write integrity of certain CIF registers and also verifies that certain CIF registers are initialized correctly after the CIF has been reset.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if a faulty register is encountered or has been initialized incorrectly by the HW.

SEE ALSO

scuSMDiag, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

NAME

scuSMCtcRegTest() - SCU SM CTC Register Test

SYNOPSIS

STATUS scuSMCtcRegTest (void)

DESCRIPTION

This function tests the read/write integrity of certain CTC registers and also verifies that certain CTC registers are initialized correctly after the SCU has been reset.

WARNINGS

This test alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if a faulty register is encountered or has been initialized incorrectly by the HW.

SEE ALSO

scuSMDiag, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

scuSMDbmRamTest(2) CREDITNET SWITCH SW MANUAL scuSMDbmRamTest(2)

NAME

scuSMDbmRamTest() - SCU SM DBM RAM Test

SYNOPSIS

STATUS scuSMDbmRamTest (void)

DESCRIPTION

This function tests the CDM Distribution Bit-map (DBM) RAM integrity. This test covers both the DBM and casting bits for each IVC. It does this through a series of DBM configurations and DBM captures. The DBM RAM testing is broken into the following sections:

1. Sequential address test.
2. Multi-pass random pattern test (or fixed if DEBUG).
3. Walking 1s test on one Free List location.
4. Walking 0s test on one Free List location.

Throughout the test, the casting bit is set the same as bit 0 in the Distribution Bit-map.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if an unexpected HW response is received.

SEE ALSO

scuSMDiag, scuSMWDbmRamTest(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

NAME

scuSMFlmRamStatTest() - SCU SM Diag. FLM RAM and Status Test

SYNOPSIS

STATUS scuSMFlmRamStatTest (void)

DESCRIPTION

This function tests the FLM's Free List Ram and the functioning of some of the bits in the FLM Status register. The Free List RAM testing is broken into the following sections:

1. Sequential address test.
2. Multi-pass random pattern test (or fixed if DEBUG).
3. Walking 1s test on one Free List location.
4. Walking 0s test on one Free List location.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if a faulty location or incorrect FLM status is encountered.

SEE ALSO

scuSMDiag, scuSFlmBasicTest(), scuSFlmIntegTest(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics"

NAME

scuSMFlmRegTest() - SCU SM FLM Register Test

SYNOPSIS

STATUS scuSMFlmRegTest (void)

DESCRIPTION

This function tests the read/write integrity of the FLM Control register and also verifies that the FLM Control, Status and CMBF registers are intialized correctly.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if a faulty register is encountered or has been initialized incorrectly by the HW.

SEE ALSO

scuSMDiag, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

NAME

scuSMPbusRegTest() - SCU SM P-Bus Register Test

SYNOPSIS

STATUS scuSMPbusRegTest (void)

DESCRIPTION

This function tests the read/write integrity of certain P- Bus registers and also verifies that certain P-Bus registers are initialized correctly after the P-Bus has been reset.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if a faulty register is encountered or has been initialized incorrectly by the HW.

SEE ALSO

scuSMDiag, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

NAME

scuSMRivcLutTest() - SCU SM Diag. R_IVC Lookup Table Test

SYNOPSIS

STATUS scuSMRivcLutTest (void)

DESCRIPTION

This function tests the CIF's R_IVC Lookup Table memory. It first checks that each location can hold its own least significant 16 address bits and then calls function xMemTest to perform a random memory test providing 99.9% fault coverage. xMemTest() does not display very detailed information when memory errors are encountered.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if a faulty location is encountered.

SEE ALSO

scuSMDiag, xMemTest(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics"

NAME

scuSMSharBuffTest() - SCU SM Diag. Shared Buffer Test

SYNOPSIS

STATUS scuSMSharBuffTest (void)

DESCRIPTION

This function tests the CIF's Shared Buffer RAM which is used to temporarily store cells being transferred to/from the Core Memory by the CIF. It first checks that each location can hold its own address and then calls function xMemTest() to perform a random memory test providing 99.9% fault coverage. xMemTest() does not display very detailed information when memory errors are encountered.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if a faulty location is encountered.

SEE ALSO

scuSMDiag, xMemTest(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics"

NAME

scuSMWDbmRamTest() - SCU SM Working DBM RAM Test

SYNOPSIS

STATUS scuSMWDbmRamTest (void)

DESCRIPTION

This function tests the CDM Working Distribution Bit-map (wDBM) RAM integrity. It covers both the DBM and the working Casting Bit. Coverage of the Chalkmark bit, which is also stored in this RAM, is performed by scuSCdmCmUpdTest(). The DBM RAM testing is broken into the following sections:

1. Sequential address test.
2. Multi-pass random pattern test (or fixed if DEBUG).
3. Walking 1s test on one Free List location.
4. Walking 0s test on one Free List location.

Throughout the test, the casting bit is set the same as bit 0 in the working Distribution Bit-map.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if an unexpected HW response is received.

SEE ALSO

scuSMDiag, scuSCdmCmUpdTest(), scuSMDbmRamTest(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics” scuSMWivcLutTest(2) **CREDITNET SWITCH SW MANUAL** scuSMWivcLutTest(2)

NAME

scuSMWivcLutTest() - SCU SM Diag. W_IVC Lookup Table Test

SYNOPSIS

STATUS scuSMWivcLutTest (void)

DESCRIPTION

This function tests the CIF's W_IVC Lookup Table memory. It first checks that each location can hold its own least significant 16 address bits and then calls function xMemTest t perform a random memory test providing 99.9% fault coverage. xMemTest() does not display very detailed information when memory errors are encountered.

WARNINGS

This test severely alters the state of the SCU and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if a faulty location is encountered.

SEE ALSO

scuSMDiag, xMemTest(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - SCU, Core and Terminator Card Diagnostics"

NAME

scuSRegCheck() - SCU Register Bit Check

SYNOPSIS

```
void scuSRegCheck
   (
   INT32     reg,       /* Register to be checked.          */
   UINT16    bit,       /* Register bit to be checked.      */
   UINT16    bitState,  /* SET or CLEAR.                    */
   STATUS *  result,    /* Ptr to caller's result.          */
   UINT32    substate   /* Error message substate if error. */
   )
```

DESCRIPTION

This function checks to see if a particular bit in one of the SCU's registers is in a particular state. If this bit is not in the specified state, then it sets *result to ERROR and sends out an SCU Start-up Diagnostics message to the UI with the substate value passed from the calling function. It will not send out an message to the UI if a substate of 0x0 is specified. This function is capable of checking the status of the following registers:

-CIF Status Register
-CIF Control Register
-FLM Status Register
-FLM Control Register
-CDM Status Register
-CDM Control Register
-CTC Start Register
-CTC Stop Register
-CTC Status Register (physically in start and stop registers)

Implementing this function allowed saving the following or similar lines of code from being repeated many times in many scuSDiag functions:

```
if(((*regRead[reg])() & bit) != (bit & bitState))
    {
    if(substate != 0x0)
        gpuUIErrorSend(SCU_S_DIAG_MSG | MSG_ERROR_0 | substate);
    *result = ERROR;
    }
```

RETURNS

Nothing but it does alter the value pointed to by *result.

SEE ALSO

scuSDiag, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

scuSSBByteCheck(2) CREDITNET SWITCH SW MANUAL scuSSBByteCheck(2)

NAME

scuSSBByteCheck() - SCU Shared Buffer Byte Check

SYNOPSIS

```
void scuSSBByteCheck
   (
   char *    sbImage,  /* Image to cmp to Shar Buff.       */
   STATUS *  result,   /* Ptr to caller's result.          */
   UINT32    substate  /* Error message substate if error. */
   )
```

DESCRIPTION

This function compares the contents of the Shared Buffer to an image containing the expected contents of the Shared Buffer. If the two images are not the same then it sets *result to ERROR and sends out an SCU Start-up Diagnostics message to the UI with the substate value passed from the calling function. It will not send out an message to the UI if a substate of 0x0 is specified.

RETURNS

Nothing but it does alter the value pointed to by *result.

SEE ALSO

scuSDiag, scuSCifDataPathTest(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

scuPortCardTest(1) CREDITNET SWITCH SW MANUAL scuPortCardTest(1)

NAME

scuPortCardTest - SCU Port Card Test

SYNOPSIS

scuInitPortCardTest() - SCU Initialization Port Card Test
scuPutCellInCore() - SCU Put Cell In Core
scuGetCellsFromCore() - SCU Get Cells From Core
scuGetCellFromCore() - SCU Get Cell From Core
printCellStruct() - Print Cell Structure
swapCell() - Swap Cell

```
STATUS scuInitPortCardTest
   (IVC_TABLE *ivcTable, UINT16 numIvcs)
STATUS scuPutCellInCore
   (UINT16 wivc, CELL_STRUCT *cell)
STATUS scuGetCellsFromCore
```

```
  (IVC_TABLE *ivcTable, UINT16 numIvcs,
  CELL_STRUCT *coreCells)
STATUS scuGetCellFromCore
  (CELL_STRUCT *coreCellPtr, UINT16 *ivc, UINT16 *cadr)
void printCellStruct
  (CELL_STRUCT *coreCellPtr)
void swapCell
  (CELL_STRUCT *coreCellPtr)
```

DESCRIPTION

This module contains a set of functions to verify whether cells can be written from a port card to the Core Memory, and read from the Core Memory to the Portcard. The main test scuInitPortCardTest() is invoked by scuDaisyChainTest() to set up for the loopback test.

WARNINGS

These tests severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

INCLUDE FILES

scu.h, scuDrv.h, scuInit.h, scuPortCardTest.h, generateCell.h

SEE ALSO

printCellStruct(2) CREDITNET SWITCH SW MANUAL printCellStruct(2)

NAME

printCellStruct() - Print Cell Structure

SYNOPSIS

```
void printCellStruct
  (
  CELL_STRUCT  *coreCellPtr  /* Ptr to the CELL_STRUCT */
  )
```

DESCRIPTION

This function prints the contents of a CELL_STRUCT. It prints a cell in the following format:
IVC:
Header:
Payload:

RETURNS

NA.

SEE ALSO

scuPortCardTest,

NAME

scuFullPortCardTest() - SCU Full Port Card Test

SYNOPSIS

```
STATUS scuFullPortCardTest
   (
   UINT16  portCard,     /* Port card slot to be tested */
   BOOL    bSummaryOnly  /* 0=detail; 1=summary         */
   )
```

DESCRIPTION

This function verifies whether cells can be written from a Portcard to the Core Memory, and read from the Core Memory to the Portcard. It calls scuInitPortCardTest(), to set up for the Portcard cell test. It then generates four cells, writes the cells to the core, and reads them back. A summary of the number of cells written to the core and the number of cells recovered from the core is printed at the end of the test.

RETURNS

OK, or ERROR if Portcard slot is invalid.

SEE ALSO

scuPortCardTest,

scuGetCellFromCore(2)   CREDITNET SWITCH SW MANUAL   scuGetCellFromCore(2)

NAME

scuGetCellFromCore() - SCU Get Cell From Core

SYNOPSIS

```
STATUS scuGetCellFromCore
   (
   CELL_STRUCT  *coreCellPtr,  /* Ptr to the CELL_STRUCT */
   UINT16       *ivc,          /* 14-bit IVC value       */
   UINT16       *cadr          /* CADR                   */
   )
```

DESCRIPTION

This function gets one cell from the core. It reads the high priority FIFO to get an IVC and CADR.

RETURNS

OK, or ERROR if unable to write from core.

SEE ALSO

scuPortCardTest,

Rev: 30 Jan 95 1

NAME

scuGetCellsFromCore() - SCU Get Cells From Core

SYNOPSIS

```
STATUS scuGetCellsFromCore
   (
   IVC_TABLE    *ivcTable,  /* Ptr to array of IVC_TABLE structures */
   UINT16       numIvcs,    /* Number of IVCs structures in table   */
   CELL_STRUCT  *coreCells  /* Ptr to buffer for cells from core    */
   )
```

DESCRIPTION

This function walks the low priority WIVC FIFO. For each entry, it reads the IVC and CADR. The contents of the cell address are read from the core and added to the coreCells buffer. Each IVC found in the FIFI is looked up in the ivcTable and its ivcFound value is set to TRUE.

RETURNS

OK (always).

WARNINGS

When this functions is called, all cells should be enqueued, scheduled and placed back into core memory. It scans through the LP W_IVC fifo, looking for the loopbacked cells.

SEE ALSO

scuPortCardTest,

NAME

sculnitPortCardTest() - SCU Initialization Port Card Test

SYNOPSIS

```
STATUS scuInitPortCardTest
   (
   IVC_TABLE  *ivcTable,  /* Ptr to array of IVC_TABLE structure */
   UINT16     numIvcs     /* no. of IVCs structures in table     */
   )
```

DESCRIPTION

This function sets up for the port card loopback test. It resets the Switch Control Unit (SCU), clears the Free List Memory (FLM) halt bit, sets the port thresholds to the maximum, initializes the WIVC lookup table with ivcOut values and fills the FLM with sequential cell addresses.

RETURNS

OK (always).

WARNINGS

SEE ALSO

scuPortCardTest, scuDaisyChainTest

NAME

scuPutCellInCore() - SCU Put Cell In Core

SYNOPSIS

```
STATUS scuPutCellInCore
   (
   UINT16       wivc,  /* IVC of cell to write         */
   CELL_STRUCT  *cell  /* Ptr to a CELL_STRUCT or NULL */
   )
```

DESCRIPTION

This function assigns the WIVC to the cell and places it in the Core Memory. The Free List Manager (FLM) determines where the cell is written in the core.

RETURNS

OK, or ERROR if unable to write cell.

SEE ALSO

scuPortCardTest,

NAME

scuShortPortCardTest() - SCU Short Port Card Test

SYNOPSIS

STATUS scuShortPortCardTest (void)

DESCRIPTION

This function generates a cell, copies the cell into the CIF shared buffer, transfers the cell from the shared buffer to the core, reads the Core Memory Buffer Fill Count (CMBFC) and scans through the WIVC FIFO.

RETURNS

OK, or ERROR if unable to write cell.

SEE ALSO

scuPortCardTest,

swapCell(2) **CREDITNET SWITCH SW MANUAL** swapCell(2)

NAME

swapCell() - Swap Cell

SYNOPSIS

```
void swapCell
  (
  CELL_STRUCT  *coreCellPtr  /* Ptr to the CELL_STRUCT */
  )
```

DESCRIPTION

This function performs byte swapping on the header and payload of the cell. The IVC is not touched. It is called BEFORE writing a cell to the shared buffer and AFTER reading a cell from the shared buffer. It does not need to be called if scuPutCellInCore() and scuGetCellsFromCore() are used.

RETURNS

NA.

SEE ALSO

scuPortCardTest,

NAME

scuStaleCadrISR() - SCU Stale CADR ISR

SYNOPSIS

void scuStaleCadrISR (void)

DESCRIPTION

This function services the CDM's stale CADR interrupt. This interrupt indicates that a stale CADR has been found while the CDM was performing a chalkmark search.

For now, this function simply increments its counter unless it is at its maximum value as well as clearing the CM_ALARM_FLAG in the CDM Status Register and re-enabling the CM Search. This ISR does not yet do anything with the stale CADR found by the chalkmark search.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

scuVerCorePat(2) CREDITNET SWITCH SW MANUAL scuVerCorePat(2)

NAME

scuVerCorePat() - SCU Verify Core Pattern

SYNOPSIS

```
STATUS scuVerCorePat
   (
   UINT16  corePattern  /* Core pattern to be verified. */
   )
```

DESCRIPTION

This function verifies that the bytes in corePattern are valid Core Memory bytes.

RETURNS

OK, or ERROR if either byte in corePattern is not a valid Core Memory byte.

SEE ALSO

scuDrv, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuVerify() - SCU Verify

SYNOPSIS

STATUS scuVerify (void)

DESCRIPTION

This function verifies that the GPU running this code is connected to an SCU.

RETURNS

OK if the GPU is connected to an SCU or ERROR if it is not.

SEE ALSO

scuDrv, sysInitCtl(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

NAME

scuWDbmCapture() - SCU Working Distribution Bitmap Capture

SYNOPSIS

```
UINT32 scuWDbmCapture
  (
  UINT16  cadr  /* IVC of working DBM to be captured */
  )
```

DESCRIPTION

This function returns the contents of the working DBM for a particular CADR passed from the calling function.

RETURNS

A 32 bit integer with cadr's casting bit in bit 18, its chalkmark in bit 16 and its DBM in bits 15 - 0 or ERROR if invalid CADR has been specified.

SEE ALSO

scuDrv, scuDbmCapture(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "CreditNet Switch - System Initialization"

scuWFifoHPAlFullISR(2) CREDITNET SWITCH SW MANUAL scuWFifoHPAlFullISR(2)

NAME

scuWFifoHPAlFullISR() - SCU HP W_IVC FIFO Almost Full ISR

SYNOPSIS

void scuWFifoHPAlFullISR (void)

DESCRIPTION

This function services the high priority W_IVC FIFO almost full interrupt. This interrupt indicates that the high priority W_IVC FIFO is 127 elements away from being full and therefore unable to store further information.

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

scuWFifoHPMidISR(2) CREDITNET SWITCH SW MANUAL scuWFifoHPMidISR(2)

NAME

scuWFifoHPMidISR() - SCU HP W_IVC FIFO Mid-range ISR

SYNOPSIS

void scuWFifoHPMidISR (void)

DESCRIPTION

This function services the high priority W_IVC FIFO mid-range interrupt. This interrupt indicates that the high priority W_IVC FIFO contains at least 127 elements.

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

scuWFifoLPAlFullISR(2) CREDITNET SWITCH SW MANUAL scuWFifoLPAlFullISR(2)

NAME

scuWFifoLPAlFullISR() - SCU LP W_IVC FIFO Almost Full ISR

SYNOPSIS

void scuWFifoLPAlFullISR (void)

DESCRIPTION

This function services the low priority W_IVC FIFO almost full interrupt. This interrupt indicates that the low priority W_IVC FIFO is 127 elements away from being full and therefore unable to store further information.

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

scuWFifoLPMidISR(2) CREDITNET SWITCH SW MANUAL scuWFifoLPMidISR(2)

NAME

scuWFifoLPMidISR() - SCU LP W_IVC FIFO Mid-range ISR

SYNOPSIS

void scuWFifoLPMidISR (void)

DESCRIPTION

This function services the low priority W_IVC FIFO mid-range interrupt. This interrupt indicates that the low priority W_IVC FIFO contains at least 127 elements.

For now, this function simply increments its counter unless it is at its maximum value.

RETURNS

Nothing.

SEE ALSO

scuIntLib, scuIntMon(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”

NAME

scuWivcLutInit() - SCU W_IVC Lookup Table Initialization

SYNOPSIS

```
STATUS scuWivcLutInit
   (
   IVC_STRUCT *  ivcData,  /* IVC configuration data.       */
   UINT16        numIvcs   /* Number of IVCs to initialize. */
   )
```

DESCRIPTION

This function sets up a W_IVC Lookup Table entry for a set of IVCs. numIvcs IVCs will be initialized based on the IVC data pointed to by ivcData. This function returns ERROR if users set numIvcs greater than the maximum number of IVCs that the switch can support.

RETURNS

OK, or ERROR if invalid input parameters is encountered in ivcData.

SEE ALSO

scuDrv, scuWivcSet(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

NAME

scuWivcLutSet() - SCU W_IVC Lookup Table Entry Set

SYNOPSIS

```
STATUS scuWivcLutSet
   (
   UINT16  ivc,        /* IVC.                          */
   UCHAR   wCsid,      /* W_CSID                        */
   UCHAR   hpWFifoEn,  /* High pri. W_IVC FIFO enable bit */
   UCHAR   lpWFifoEn   /* Low pri. W_IVC FIFO enable bit  */
   )
```

DESCRIPTION

This function sets up a W_IVC Lookup Table entry for one IVC. IVC ivc will be initialized with a W_CSID corresponding to wCsid and high and low priority W_IVC FIFO enable bits corresponding to hpWFifoEn and lpWFifoEn respectively.

RETURNS

OK, or ERROR if invalid input parameters have been specified.

SEE ALSO

scuDrv, scuWivcLutInit(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization” scuWivcLutSetup(2) CREDITNET SWITCH SW MANUAL scuWivcLutSetup(2)

NAME

scuWivcLutSetup() - SCU W_IVC Lookup Table Setup

SYNOPSIS

```
STATUS scuWivcLutSetup
   (
   IVC_STRUCT  *ivcData,   /* IVC configuration data.                  */
   UINT16      numIvcs,    /* Number of IVCs to initialize.            */
   UINT16      ivc,        /* IVC for single IVC init.                 */
   UCHAR       wCsid,      /* W_CSID for single IVC initi              */
   UCHAR       hpWFifoEn,  /* High pri. W_IVC FIFO enable - single IVC */
   UCHAR       lpWFifoEn   /* Low pri. W_IVC FIFO enable - single IVC  */
   )
```

DESCRIPTION

This function sets up a W_IVC Lookup Table entry for one or a set of IVCs. If the user sets numIvcs to 1 then IVC ivc is initialized with a W_CSID of wCsid, and high and low priority W_IVC FIFO enable bits set to hpWFifoEn and lpWFifoEn respectively. If numIvcs is greater than 1 then numIvcs IVCs are configured in the W_IVC Lookup table according to the IVC configuration data pointed to by ivcData. This function returns ERROR if users set numIvcs greater than the maximum number of IVCs that the switch can support.

WARNING

This is not the main function(s) for use to initialize the W_IVC Lookup Table. Most users should instead use scuWivcLutInit() to initialize more than one entry or scuWivcLutSet() to initialize one entry in the table.

RETURNS

OK, or ERROR if invalid input parameters have been specified.

SEE ALSO

scuDrv, scuWivcLutSet(), scuWivcLutInit(), “Design Specificcation - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

scuWriteXCells(2) CREDITNET SWITCH SW MANUAL scuWriteXCells(2)

NAME

scuWriteXCells() - Write X Cells into the Core Memory

SYNOPSIS

```
void scuWriteXCells
  (
  UINT16  wIvc,  /* W_IVC value used for write cycle.   */
  UINT32  x      /* # cells to write.                   */
  )
```

DESCRIPTION

This function writes the cell located in cell buffer 0 of the SCU's shared buffer into the Core Memory x times using a W_IVC of wIvc.

WARNING

This function has little practical use but is mainly used for debugging.

RETURNS

Nothing.

SEE ALSO

scuDrv, scuCellXferToCore(), “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”

scuDaisyChainTest(1) CREDITNET SWITCH SW MANUAL scuDaisyChainTest(1)

NAME

scuDaisyChainTest - SCU Daisy Chain Test

SYNOPSIS

scuDaisyChainTest() - SCU Daisy Chain Test

```
STATUS scuDaisyChainTest
  (UINT16 options)
```

DESCRIPTION

This module contains one function scuDaisyChainTest(), to test whether all the port cards can write to and read from the Core Memory.

WARNINGS

This test severely alters the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

INCLUDE FILES

scu.h, scuDrv.h, scuPortCardTest.h, generateCell.h, scuDaisyChainTest.h

SEE ALSO

scuPortCardTest, generateCell

NAME

scuSCoreTest - SCU Start-Up Core Memory (CM) Test

SYNOPSIS

scuSCoreTest() - SCU Start-up CM Test
scuCMTestGeneratePatterns() - SCU CM Test Generate Patterns
scuCMTestEntireCore() - SCU CM Test Entire Core
scuCMCompareCell() - SCU CM Compare Cell
scuCMTestLostAddresses() - SCU CM Test Lost Addresses
scuCMTestOneAddress() - SCU CM Test One Address
scuCMWalkingTest() - SCU CM Walking Test

```
STATUS scuSCoreTest
  (void)
STATUS scuCMTestGeneratePatterns
  (UINT16 patternType, UINT16 *patternBuff16k)
STATUS scuCMTestEntireCore
  (UINT16 addressType, UINT16 readBackOrder,
  UINT16 *patternBuff16k, UINT32 *crc8ErrorCount)
STATUS scuCMCompareCell
  (UINT16 cifIndex, UINT16 sPattern, UINT16 current_CADR,
  UINT32 *crc8ErrorCount)
STATUS scuCMTestLostAddresses
  (UINT32 *crc8ErrorCount)
STATUS scuCMTestOneAddress
  (UINT16 cadr, UINT32 *crc8ErrorCount)
STATUS scuCMWalkingTest
  (BOOL testType)
```

DESCRIPTION

This module contains a set functions for testing CM for stuck bits, leakage, address decoding errors and address and data line shorts froma black box point of view. It also builds a list of cell addresses whichare faulty whenever possible and attempts to isolate the 16-bit word within CM that was found to be faulty when the test fails. This test cannot determine whether the failure occurred because of a faulty memory chip because the error can also be caused by CM's commutation logic and data paths. Any error information collected by this test however can still provide some isolation information in order to help debug the cause of the errors. The byte patterns used in this test are restricted to the set of bytes that are left after 7 to 8 encoding has been performed on a cell. Because of this, the patterns used to test this memory must be chosen carefully. The main test scuSCoreTest() is invoked by scuSDiagCtl() during CreditNet Switch Initialization. The function scuSCoreTest() may also be invoked form VxWorks shell for debugging and faulty isolation purposes.

WARNINGS

These tests severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

INCLUDE FILES

scu.h, scuDrv.h, scuInit.h, scuSCoreTest.h, generateCell.h

SEE ALSO

scuDrv, scuSDiag, scuSDiagCtl(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "SCU and Core Memory Diagnostics (DD)", pages 151-153

NAME

scuWDbmIntegTest - SCU Working Distribution Bit Map (wDBM) Integration Test

SYNOPSIS

scuWDbmIntegTest() - SCU wDBM Integration Test
scuWDbmWalkingOneTest() - SCU wDBM Walking One Test
scuWDbmWalkingZeroTest() - SCU wDBM Walking Zero Test
scuWDbmMultiCastTest() - SCU wDBM Multi-Cast Test
setOwnershipTable() - Set Ownership Table

```
STATUS scuWDbmIntegTest
   (void)
STATUS scuWDbmWalkingOneTest
   (UINT16 pmExist)
STATUS scuWDbmWalkingZeroTest
   (UINT16 pmExist)
STATUS scuWDbmMultiCastTest
   (UINT16 pmExist)
void setOwnershipTable
   (UINT32 portNum)
```

DESCRIPTION

This module contains a set of tests that perform start-up integration diagnostics on the Distribution Bit Map (DBM) and the Working Distribution Bit Map (wDBM). They test the interactions of the SCU's DBMs and CMBF with PM cell reads. Walking one data path test, walking zero data path test, and multi-cast test are performed on the DBMs to check for stuck bits, leakage, and data line shorts.

The individual tests can also be called for debugging purposes from the VxWorks Shell.

This module informs users of status and results by sending messages to the switch User Interface. It can also print more detailed information out to the VxWorks Shell if DIP Switch 2 on the GPU is turned on. This information printed to the shell can be very useful for HW debugging.

WARNINGS

These tests severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

INCLUDE FILES

SEE ALSO
scuSDiag

scuDaisyChainTest(2) CREDITNET SWITCH SW MANUAL scuDaisyChainTest(2)

NAME

scuDaisyChainTest() - SCU Daisy Chain Test

SYNOPSIS

```
STATUS scuDaisyChainTest
  (
  UINT16  options  /* 0=det,uncont;1=sum,uncont;2=det,cont;3=sum,cont */
  )
```

DESCRIPTION

This function determines whether all the port cards can write cells into the core and read cells from the core. It calls scuInitPortCardTest() to set up for the loopback test, scuPutCellInCore() to write the cell into the core, and scuGetCellsFromCore() to read the cell from the core. A message will be printed if a port card fails to write the cell into the core. A summary of the number of recovered cells is printed at the end of the test.

RETRUNS

OK, or ERROR if no port card is found.

SEE ALSO

scuDaisyChainTest,

NAME

scuCMCompareCell() - scuCMCompareCell

SYNOPSIS

```
STATUS scuCMCompareCell
   (
   UINT16  cifIndex,        /* Cif buffer containing cell                    */
   UINT16  sPattern,        /* Expected pattern                              */
   UINT16  current_CADR,    /* Starting address of where cell was read from core */
   UINT32  *crc8ErrorCount  /* Pointer to crc-8 error counter                */
   )
```

DESCRIPTION

Compare the contents of a cell to a 16 bit pattern Print out a list of all bad bytes and check for a crc-8 error

RETURNS

OK, or ERROR.

WARNINGS

SEE ALSO

scuSCoreTest, _S_C_U _H_a_r_d_w_a_r_e _S_p_e_c_i_f_i_c_a_t_i_o_n _S_C_U _a_n_d _C_o_r_e _M_e_m_o_r_y_D_i_a_g_n_o_s_t_i_c_s _D_e_s_i_g_n

NAME

scuCMTestEntireCore() scuCMTestEntireCore

SYNOPSIS

```
STATUS scuCMTestEntireCore
   (
   UINT16  addressType,      /* Macro to describe how the addresses are generated             */
   UINT16  readBackOrder,    /* Macro to describe which order to read back and verify the cells */
   UINT16  *patternBuff16k,  /* Pointer to 16k 16 bit pattern array                          */
   UINT32  *crc8ErrorCount   /* Pointer to CRC-8 error counter variable                       */
   )
```

DESCRIPTION

This function writes cells to the entire core and verifies them by reading them back.

PARAMETERS

addressType - either SEQUENTIAL_ADDRESS_ORDER or
PHYSICAL_ADDRESS_ORDER
readBackOrder - either NORMAL_READ or REVERSE_READ
patternBuff16k - array of 16k 16 bit cell patterns
(must be filled in before
calling this function
crc8ErrorCount - gets incremented upon each CRC-8
error detected

RETURNS

OK, or ERROR.

WARNINGS

SEE ALSO

scuSCoreTest, _S_C_U _H_a_r_d_w_a_r_e _S_p_e_c_i_f_i_c_a_t_i_o_n _S_C_U _a_n_d _C_o_r_e _M_e_m_o_r_y_D_i_a_g_n_o_s_t_i_c_s _D_e_s_i_g_n

scuCMTestGeneratePatterns(2) CREDITNET SWITCH SW MANUAL scuCMTestGeneratePatterns(2)

NAME

scuCMTestGeneratePatterns() - scuCMTestGeneratePatterns

SYNOPSIS

```
STATUS scuCMTestGeneratePatterns
    (
    UINT16  patternType,     /* Macro that describes the type of pattern to generate    */
    UINT16  *patternBuff16k  /* A pre-allocated 16k 16 bit buffer to store the patterns */
    )
```

DESCRIPTION

This function generates an array of 16k 16 bit patterns. The patterns can be sequential, reverse or pseudo-random.

PARAMETERS

patternType - either SEQUENTIAL_ADDRESS_PATTERNS,
INVERSE_ADDRESS_PATTERNS or
BIT_PATTERNS
patternBuff16k - gets filled in with 16k 16
bit patterns
** MUST BE PRE-ALLOCATED **

RETURNS

OK, or ERROR.

WARNINGS

SEE ALSO

scuSCoreTest, _S_C_U _H_a_r_d_w_a_r_e _S_p_e_c_i_f_i_c_a_t_i_o_n _S_C_U _a_n_d _C_o_r_e _M_e_m_o_r_y_D_i_a_g_n_o_s_t_i_c_s _D_e_s_i_g_n

scuCMTestLostAddresses(2) CREDITNET SWITCH SW MANUAL scuCMTestLostAddresses(2)

NAME

scuCMTestLostAddresses() - scuCMTestLostAddresses

SYNOPSIS

```
STATUS scuCMTestLostAddresses
  (
  UINT32  *crc8ErrorCount  /* crc8 error counter */
  )
```

DESCRIPTION

Test core addresses 0, 1, 2, 3, 0x3ffb, 0x3ffc, 0x3ffd, 0x3ffe, 0x3fff

RETURNS

OK, or ERROR.

WARNINGS

SEE ALSO

scuSCoreTest, _S_C_U _H_a_r_d_w_a_r_e _S_p_e_c_i_f_i_c_a_t_i_o_n _S_C_U _a_n_d _C_o_r_e _M_e_m_o_r_y_D_i_a_g_n_o_s_t_i_c_s _D_e_s_i_g_n

scuCMTestOneAddress(2) CREDITNET SWITCH SW MANUAL scuCMTestOneAddress(2)

NAME

scuCMTestOneAddress() - SCU Core Memory Test One Address

SYNOPSIS

```
STATUS scuCMTestOneAddress
  (
  UINT16  cadr,          /* The core address to test       */
  UINT32  *crc8ErrorCount  /* Pointer to crc-8 error counter */
  )
```

DESCRIPTION

Test an individual core memory address

RETURNS

OK, or ERROR.

WARNINGS

SEE ALSO

scuSCoreTest, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)” “SCU and Core Memory Diagnostics (DD)”, pages 151-153

NAME

scuCMWalkingTest() - scuCMWalkingTest

SYNOPSIS

```
STATUS scuCMWalkingTest
  (
  BOOL  testType  /* Either zero or one.  Determines whether to perform a walking 1 or walking 0 test */
  )
```

DESCRIPTION

Perform a walking 1's or walking 0's test on the data lines

RETURNS

OK, or ERROR.

WARNINGS

SEE ALSO

scuSCoreTest, _S_C_U _H_a_r_d_w_a_r_e _S_p_e_c_i_f_i_c_a_t_i_o_n _S_C_U _a_n_d _C_o_r_e _M_e_m_o_r_y
_D_i_a_g_n_o_s_t_i_c_s _D_e_s_i_g_n

NAME

scuSCoreTest() - SCU Start-up CM Test

SYNOPSIS

STATUS scuSCoreTest (void)

DESCRIPTION

This function tests the Core Memory and the data path to/from it for stuck bits, leakage, address decoding errors and address and data line shorts. The tests performed are:

1. Test entire core with a payload pattern of CADR.
2. Test entire core with bit patterns.
3. Test entire core with a payload pattern of inverse CADR.
4. Walking one test on CM data path starting at CADR 0x4.
5. Walking zero test on CM data path starting at CADR 0x4.

RETURNS

OK, or ERROR if any one test performed on the Core Memory fails.

WARNINGS

These tests severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

SEE ALSO

scuSCoreTest, "Design Specification - CreditNet Switch - Switch Control Unit (SCU)", "SCU and Core Memory Diagnostics (DD)", pages 151-153

NAME

setOwnershipTable() - Set Ownership Table

SYNOPSIS

```
void setOwnershipTable
   (
   UINT32  portNum  /* PM */
   )
```

DESCRIPTION

This function sets the ownership table.

RETURNS

NA.

SEE ALSO

scuWDbmIntegTest,

NAME

scuWDbmIntegTest() - SCU wDBM Integration Test

SYNOPSIS

STATUS scuWDbmIntegTest (void)

DESCRIPTION

This function tests the DBM and the wDBM, and the data path to/from them for stuck bits, leakage, and data line shorts.

WARNINGS

The tests invoked by this function may severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if any of the tests performed on the DBMs fail.

SEE ALSO

scuWDbmIntegTest, scuSMDbmRamTest(), scuSMWDbmRamTest(), scuSCdmMonFATest(), "Design Specification - CreditNet Switch - Switch Control Unit (SCU)"

scuWDbmMultiCastTest(2) CREDITNET SWITCH SW MANUAL scuWDbmMultiCastTest(2)

NAME

scuWDbmMultiCastTest() - SCU wDBM Multi-Cast Test

SYNOPSIS

```
STATUS scuWDbmMultiCastTest
   (
   UINT16  pmExist  /* PM availability pattern */
   )
```

DESCRIPTION

This function performs a multi-cast test on the (w)DBM. One cell is written to the Core Memory and read back by all PMs. After each read, the Free Address (FA) signal is checked for unassertion. After all the PMs have read out the cell, the Free Address (FA) signal is checked for assertion.

WARNINGS

The tests invoked by this function may severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if DBMs not modified correctly and/or FA signal not asserted correctly for each read.

SEE ALSO

scuWDbmIntegTest, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

scuWDbmWalkingOneTest(2) CREDITNET SWITCH SW MANUAL scuWDbmWalkingOneTest(2)

NAME

scuWDbmWalkingOneTest() - SCU wDBM Walking One Test

SYNOPSIS

```
STATUS scuWDbmWalkingOneTest
   (
   UINT16  pmExist  /* PM availability pattern */
   )
```

DESCRIPTION

This function performs a walking one test on the (w)DBM. Cells are written to the Core Memory and read back by PM that is specified according to the (w)DBM. The Free Address (FA) signal is checked for assertion when a cell is read.

WARNINGS

The tests invoked by this function may severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if DBMs not modified correctly and/or FA signal not asserted correctly for each read.

SEE ALSO

scuWDbmIntegTest, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

scuWDbmWalkingZeroTest(2) CREDITNET SWITCH SW MANUAL scuWDbmWalkingZeroTest(2)

NAME

scuWDbmWalkingZeroTest() - SCU wDBM Walking Zero Test

SYNOPSIS

```
STATUS scuWDbmWalkingZeroTest
   (
   UINT16  pmExist  /* PM availability pattern */
   )
```

DESCRIPTION

This function performs a walking zero test on the (w)DBM. Cells are written to the Core Memory and read back by PM that is specified according to the (w)DBM. The Free Address (FA) signal is checked for unassertion when a cell is read.

WARNINGS

The tests invoked by this function may severely alter the state of the SCU and switch and should only be invoked during System Initialization or for debugging purposes.

RETURNS

OK, or ERROR if DBMs not modified correctly and/or FA signal not asserted correctly for each read.

SEE ALSO

scuWDbmIntegTest, “Design Specification - CreditNet Switch - Switch Control Unit (SCU)”, “CreditNet Switch - System Initialization”, “CreditNet Switch - SCU, Core and Terminator Card Diagnostics”

rFifoAlFullFlag(2) CREDITNET SWITCH SW MANUAL rFifoAlFullFlag(2)

NAME

rFifoAlFullFlag() rFifoAlFullFlag = rFifoMidFlag = 0x0;

SYNOPSIS

Fill up the R_IVC using the full range of IVC patterns then read them back to verify FIFO integrity. Along the way, verify FIFO boundary conditions, the Almost Full and Midrange interrupts occur when they should be and that the FIFO's status bits are working correctly.

SEE ALSO

scuSDiag