Simplifying System Integration™

78Q8430
Driver Manual for ST 5100/OS-20 with NexGen TCP/IP Stack

March, 2008
Rev. 1.0

Teridian Semiconductor Corp., 6440 Oak Canyon, Suite 100, Irvine, CA 92618
TEL (714) 508-8800, FAX (714) 508-8877, http://www.teridian.com

Table of Contents

Figures

Tables

1 Introduction

The Teridian Semiconductor Corporation (TSC) 78Q8430 is a single chip 10/100 Ethernet MAC and PHY controller supporting multi-media offload. The device is optimized to enhance throughput and offload network protocol tasks from the host processor for demanding multi-media applications found in Set Top Boxes, IP video, and Broadband Media Appliances.

This document describes the 78Q8430 software device driver for ST/OS-20. The document is based on the following driver software version:

- SW Revision TSC8430B_V1.01, 03/07/2008.

A 78Q8430 Demo Board (D8430T3B_STEM) is available to support development of embedded applications in conjunction with an ST STi5100 IPSTB development platform and ST ST20R2.0.5 SW tools. The 78Q8430 ST/OS-20 device driver includes the operating system (OS) and platform independent files and the OS and platform (CPU, board) dependent files. The ST/OS-20 device driver uses a specific configuration of the OS and platform which is dependent on the generic Teridian Ethernet device driver. This driver runs on the STi5100 IPSTB hardware platform with the NexGen TCP/IP protocol stack for IP video streaming demo application software.

2 System Requirements

2.1 Hardware Requirements

The following list describes the minimum hardware requirements for a 78Q8430 ST/OS-20 based development platform:

- 78Q8430 demo board (D8430T3B_STEM).
- Software development PC with the following minimum requirements: Pentium® 4 CPU with 256 MB RAM and 40 GB hard drive, running either Windows® 2000 or Windows XP.
- IP Server PC with the following minimum requirements: Pentium 4 CPU with 256 MB RAM and 40 GB hard drive, 10/100 ports for 78Q8430 demo board connection, running either Windows 2000 or Windows XP.
- 10/100 HUB or switch.
- STi5100 evaluation platform. The STi5100 communicates with the 78Q8430 registers at base memory address 0x43038000.
- ST Microconnect JTAG emulator. This device loads the IPSTB software into the STi5100 evaluation platform.

2.2 Software Requirements

The following list describes the minimum software requirements for embedded applications programming on a 78Q8430 ST/OS-20 based development platform:

- ST20 Toolset: STi5100 BSP Version 2.0.5 Patch 1.
- IPBox: contains web_server, htdocs, and video_server folders.
- IPSTB application: Ipstba3_esp – 5100.

3 Device Driver Structure

This 78Q8430 ST/OS-20 device driver software is a customized version of the generic Teridian Ethernet device driver software. It is configured with wrapper code for the NexGen TCP/IP protocol stack and other protocols (RTSP, RTP) to stream the MPEG-2 transport stream. The wrapper code connects the generic device driver API to the NexGen TCP/IP stack.

3.1 Device Driver Files

3.1.1 File Partitions

The device driver software includes 4 groups of files:

- OS and platform independent files:
  - tsccore.c
  - commem.h
  - comregs.h
- TSC OS and platform dependent files:
  - tscport.c
  - tscport.h
  - [optional] wrapper files: ether_tsc78q8430.c, ether_tsc78q8430.h
- Target OS and platform dependent files:
  - targets.cfg
  - mb390_mem.cfg
- Modified TCP/IP protocol stack files:
  - ipncs.c
  - tcpncs.c
  - udpncs.c

3.1.2 File Directory Structure

Table 1, Table 2 and Table 3 list the directory and file structure for the 78Q8430 driver software and a brief description of each file.

Table 1: Teridian Source File Tree

Directory Path	File Name	File Description
C:\ipstba5\src\nexgen_drv	ether_tsc78q8430.c	Wrapper file which includes Teridian source files
	tscport.c	OS and H/W dependent code
	tsctest.c	Test application code
	tsccore.c	Core driver code
C:\ipstba5\include	ether_tsc78q8430.h	Wrapper file which include Teridian header files
	tscport.h	OS and H/W dependent headers
	commem.h	Common memory, data structure declaration
	comregs.h	78Q8430 Register declaration

Table 2: ST/OS-20 Configuration Source File Tree

Directory Path	File Name	File Description
C:\ipstba5\config\platform	targets.cfg	IPSTB Target configuration
	mb390_mem.cfg	FMI bus configuration for 78Q8430 registers and SRAM

Table 3: NEXGEN TCP/IP Files for Hardware Checksum

Directory Path	File Name	File Description
C:\ipstba5\src\nexgen_drv	ipncs.c (ip.c)	Add IP checksum HW/SW option
	udpncs.c (udp.c)	Add UDP checksum HW/SW option
	tcpncs.c (tcp.c)	Add TCP checksum HW/SW option

3.2 ST/OS-20 Header Files

The 78Q8430 device driver software requires the following ST/OS-20 header files to be included:

```
#include <task.h>
#include <stdio.h>
#include <stdlib.h>
#include <message.h>
#include <string.h>
#include <heap.h>
#include <cache.h>
#include <debug.h>
#include <interrup.h>
#include <ostime.h>
#include <c1timer.h>
#include <time.h>
#include <semaphor.h>
#include <debug.h>

#include "stddefs.h"
#include "commem.h"
#include "comregs.h"
```

3.3 Data Structures

The 78Q8430 device driver for ST/OS-20 interfaces to the NexGen TCP/IP stack with the structures described below.

3.3.1 NG_TSC_STRUCT

NG_TSC_STRUCT and DEVICE_CONTROL_STRUCT structures are defined in the TSC Ethernet source module commem.h. Figure 1 shows the call graph for the NG_TSC_STRUCT structure.

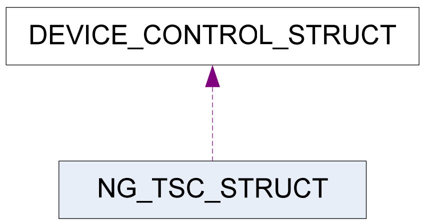

Figure 1: NG_TSC_STRUCT Call Graph

3.3.2 NET_CONTROL_STRUCT

The NET_CONTROL_STRUCT structure is defined in the TSC Ethernet source module commem.h. Figure 2 shows the call graph for the NET_CONTROL_STRUCT structure.

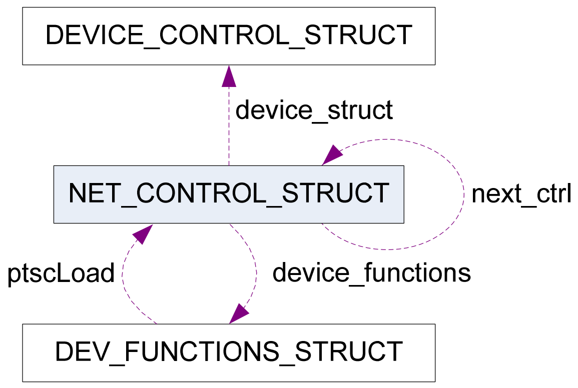

Figure 2: NET_CONTROL_STRUCT and DEV_FUNCTIONS_STRUCT Call Graph

3.3.3 DEV_FUNCTIONS_STRUCT

The DEV_FUNCTIONS_STRUCT structure is defined in the TSC Ethernet source module commem.h. Figure 2 shows the call graph for the DEV_FUNCTIONS_STRUCT structure.

3.3.4 DEVICE_CONTROL_STRUCT

The DEVICE_CONTROL_STRUCT structure is defined in the TSC Ethernet source module commem.h. STETHER_ functions refer to this control block as PDEV_CTRL. See the TSC Ethernet source module tscport.c for its usage. Figure 3 shows the call graph for the DEVICE_CONTROL_STRUCT structure.

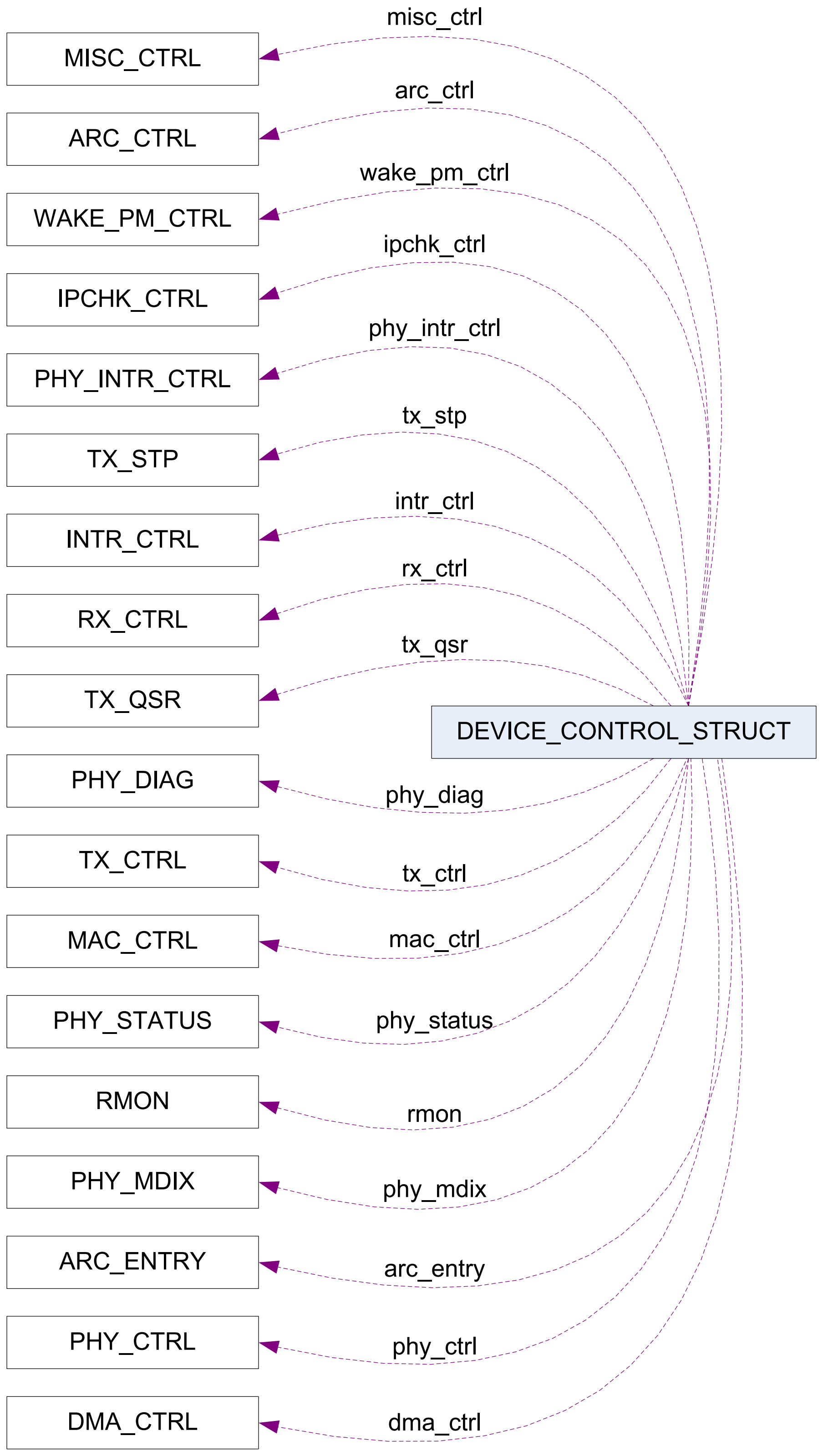

Figure 3: DEVICE_CONTROL_STRUCT Call Graph

3.4 Device Driver Options

Table 4 lists the configuration options for the device driver. Table 5 lists the software default values for several important 78Q8430 registers and parameters used by the driver. The file Commen.h contains these default values. To change the default values, make the changes in Commem.h and recompile the driver. A discussion of these values and how they are chosen follows the table.

Table 4: Device Driver Configuration Options

Option Name	Settings	Description

DRIVER_MODE	INTERRUPT_MODE = 1
	POLLING_MODE = 2
		Choice of polling mode or interrupt driven.
		Polling mode is used only for diagnostics.

TSC_CPU (TSCCPU)	ST5100 = 3	Selects target CPU platform for the driver.

Table 5: Driver Default Values for Important 78Q8430 Registers and Parameters

Driver Variable	78Q8430 Register		Register Field	Default Value
	Name	Address		
MEM_LEAK_OPT_VALUE	NA			0x0F
EAR_INT_DELAY_CNT	IDCR	0x180	IDC	0x1FFF
WM_INT_FREE_BLOCK	WMVR	0x190	Interrupt	0x0F
WM_HR_FREE_BLOCK			Headroom	0x04
WM_PAUSE_FREE_BLOCK			Pause	0x09

1. MEM_LEAK_OPT_VALUE: memory leak detection optimal value.

The driver uses the MEM_LEAK_OPT_VALUE parameter to detect a memory leak condition. Several conditions must be considered when deciding what this value should be.

Theoretically, subtracting the number of memory blocks used by the QUEs (Nq) from the total number of memory blocks available should equal the number of free blocks. In practice, this may not be exactly true due to the fact that a dynamic QUE may hold one memory block in reserve. Additionally, one or more memory blocks may be allocated or de-allocated between the time the number of free blocks are read and the time it takes to complete the calculation.

Given that there are 127 total memory blocks available and Nf is the number of currently free memory blocks, the number of unaccounted memory blocks (Na) can be found from the following equation:

$$Na = 127 - Nq - Nf$$

If Na is greater than MEM_LEAK_OPT_VALUE, it is an indication that there are some memory leaks and the driver issues a software reset.

2. EAR_INT_DELAY_CNT: delay in the interrupt if early interrupt is used.

The early receive interrupt has a delay timer feature. This feature is intended to leverage the deep receive buffer to decrease interrupt handling overhead in the host. Normally, the early receive interrupt is triggered as soon as any data for a received frame is placed into the receive QUE. The receive interrupt delay timer delays this interrupt for a programmable amount of time to allow the receive QUE to accumulate more data. In this way, under conditions of heavy load, several frames can be serviced by a single receive interrupt.

The interrupt timer is linked to the PHY speed such that the timer value is measured in byte times, or in other words, a single tick on the interrupt delay timer is equal to the amount of time it would take the PHY to receive a single byte. The timer does not require that an actual byte be received so the interrupt delay feature will not cause small frames to be left in the QUE while waiting for more data. Anytime data is added to the receive QUE and the interrupt delay timer is enabled, the timer is started if it is not already running. If the interrupt delay timer is already running when data is added to the receive QUE, the running timer is not affected. When a BLOCK is removed from the receive QUE, the interrupt delay timer is reset. This means that the driver must completely empty the receive QUE each time it services the delay timer interrupt or risk stranding data in the QUE.

3. WM_INT_FREE_BLOCK: Watermark interrupt value.

The watermark values are set based on the minimum number of free memory blocks that must be available to avoid the specified action. In the case of the interrupt watermark, the specified action is an interrupt. The value of the interrupt watermark should be set low enough that it is not triggered under ordinary circumstances, as this would increase the interrupt service load of the system. The interrupt water mark should also be set high enough that the interrupt is triggered while there is still enough free memory to keep the system moving long enough to take action and avoid data loss due to a lack of memory.

4. WM_HR_FREE_BLOCK: Headroom Watermark.

The Headroom watermark specifies the number of free memory BLOCKS below which the MAC receiver is halted. This effectively reserves some blocks of memory for MAC transmit. The default value for the Headroom watermark is 0x04. This allows the MAC transmit to have at least 4 blocks of memory to send a packet.

5. WM_PAUSE_FREE_BLOCK: Free blocks before sending pause.

The PAUSE watermark specifies the minimum number of free memory BLOCKS that triggers the automatic transmission of the PAUSE frame.

4 ST IPSTB NexGen 78Q8430 Ethernet API

This section shows an example of the specific integration of the 78Q8430 device driver in the STi5100 IPSTB reference design. The simple NexGen interface code (contained in the ether_tsc78q8430.c and ether_tsc78q8430.h files) connects the device driver to the NexGen TCP/IP protocol stack.

The API described below is defined in the TSC driver source modules tscport.h and tscport.c. The API consists of the following functions:

- STETHER_CopyData ()
- STETHER_Config ()
- STETHER_Config_ARC ()
- STETHER_HandleCompletedTXBuffers ()
- STETHER_InterruptHandler ()
- STETHER_Open ()
- STETHER_Receive()
- STETHER_Send ()
- STETHER_Start ()

Note: The STi5100 communicates with the 78Q8430 registers at base memory address 0x43038000.

4.1 STETHER_CopyData ()

Prototype:

void STETHER_CopyData(NGifnet * netp)

Description:

Copies data from the 78Q8430 to NexGen buffers in a task. The function does a serialized call to STETHER_receive.

Parameters:

netp	network interface type NGethifnet_tsc

Returns:

none

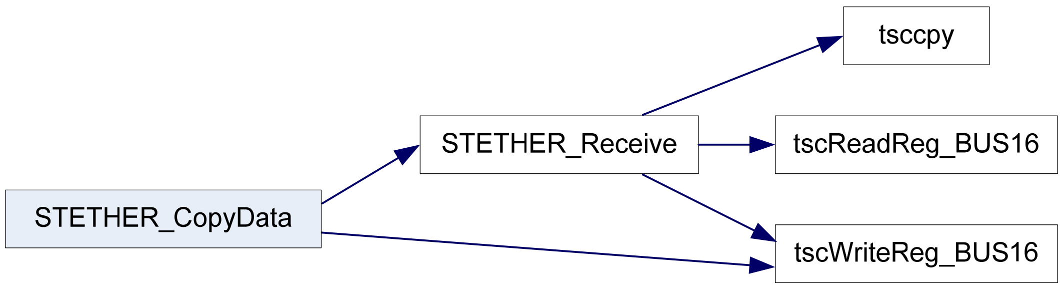

Figure 4: STETHER_CopyData Call Graph

4.2 STETHER_Close ()

Prototype:

int STETHER_Close(NGifnet *netp)

Description:

Driver cleanup and Ethernet controller shutdown are handled by this function. It stops the driver, deletes the task and serialization, unloads the driver and de-queues and frees any pending buffers.

Parameters:

netp	network interface type NGethifnet_tsc

Returns:

NG_EOK if OK.
NG_EALREADY Error if interface is not up.

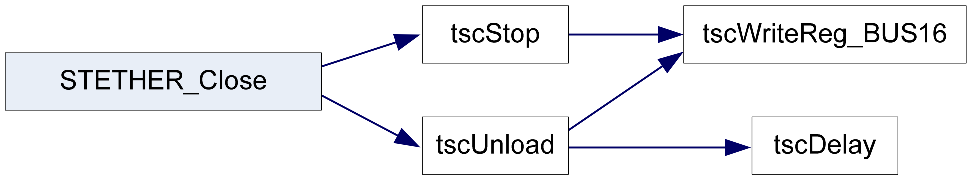

Figure 5: STETHER_Close Call Graph

4.3 STETHER_Config ()

Prototype:

void STETHER_Config (U32 Eth_BaseAddr, U8 Addr_Shift,U8 Tsc_Interrupt,U8 Int_Level,
U8 Use16Bit,U8 Trans_Len)

Description:

This routine configures the 78Q8430 interface. It records the input parameters in globally available variables for use by the driver.

Parameters:

Eth_BaseAddr	78Q8430 base address
Addr_Shift	address shift
Tsc_Interrupt	tsc interrupt
Int_Level	interrupt level
Use16Bit	use 16-bit bus width
Trans_Len	transfer packet length

Returns:

none

4.4 STETHER_Config_ARC ()

Prototype:

void STETHER_Config_ARC(NGifnet *netp)

Description:

This function configures the ARC table with the contents of the preset arc_entry[] array found in DEVICE_CONTROL_STRUCT. The ARC address has the same format as a standard 6 byte MAC address. It implements all the available default rules.

Parameters:

netp	network interface type NGethifnet_tsc

Returns:
none

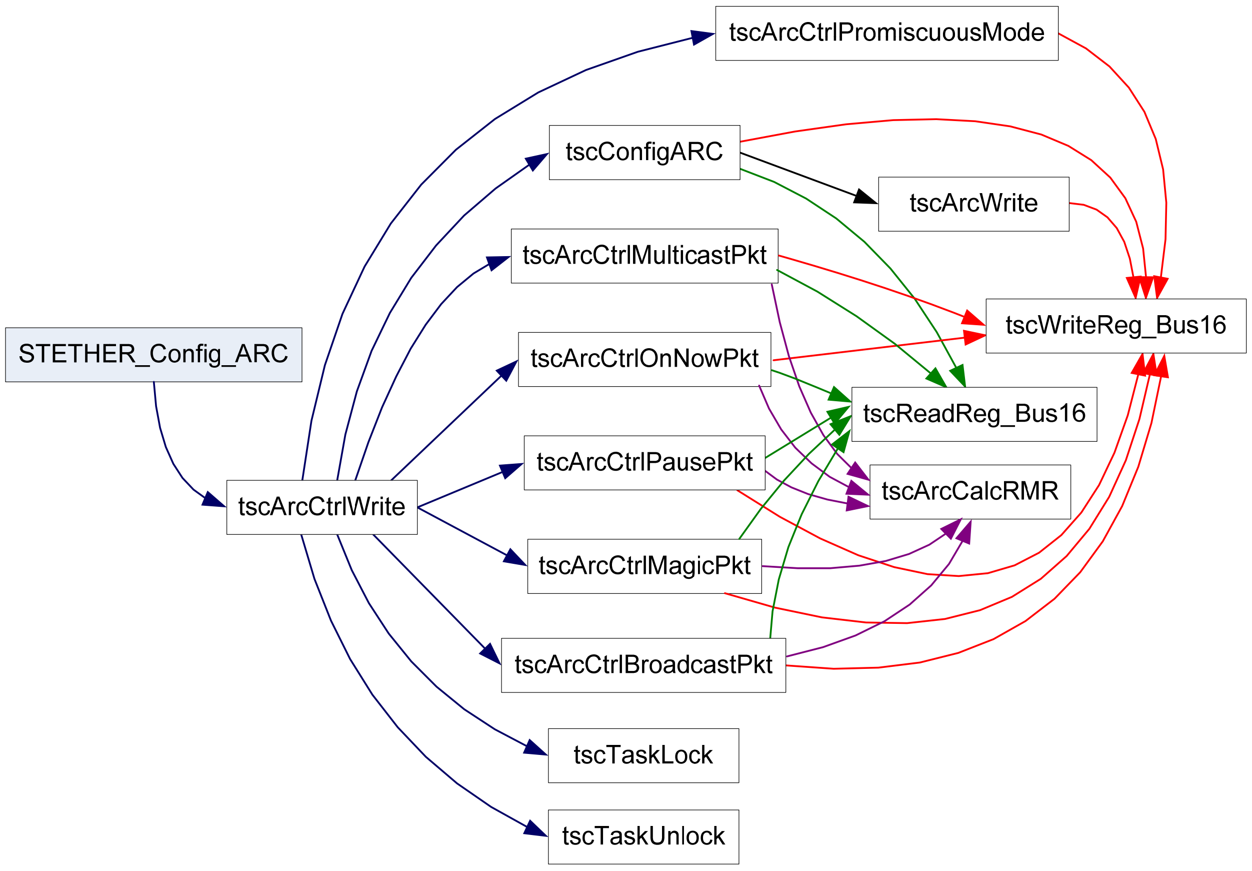

Figure 6: STETHER_Config_ARC Call Graph

4.5 STETHER_HandleCompletedTXBuffers ()

Prototype:
INT4 STETHER_HandleCompletedTXBuffers (NGifnet *netp)

Description:
This routine handles TX and error interrupts derived from **tscIsr()** and **tscDpr()**, which in turn, are the result of STETHER_Send calls for data transmission. It extracts and records the status for the completed transmissions if statistics collection is active. On exit, it checks for any additional pending transmissions and if found, invokes STETHER_Send to fulfill the TX request.

Parameters:

netp	network interface type NGethifnet_tsc

Returns:
Number of TX packets processed.

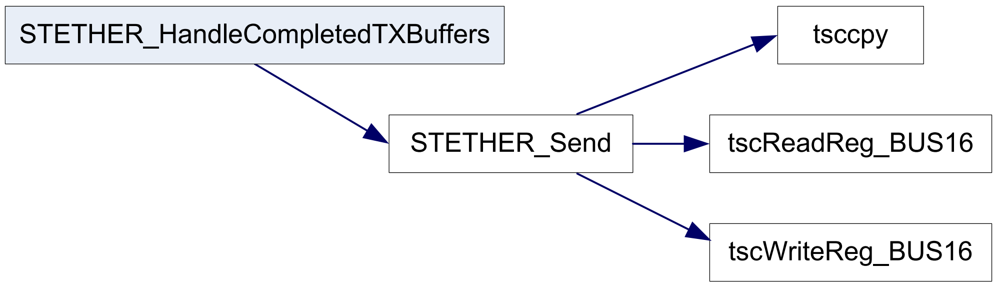

Figure 7: STETHER_HandleCompletedTXBuffers Call Graph

4.6 STETHER_InterruptHandler ()

Prototype:

void STETHER_InterruptHandler(NGifnet* netp)

Description:

This function is the interrupt handler wrapper. It handles TX, RX and error interrupts derived from **tscIsr()** and **tscDpr()**. The function begins by isolating the interrupt source and then clearing the interrupt(s). If statistics are being collected, it updates the interrupt related portion. Based on its interrogation of the interrupt source, it may call STETHER_HandleCompletedTxBuffers.

Parameters:

netp	network interface type NGethifnet_tsc

Returns:

none

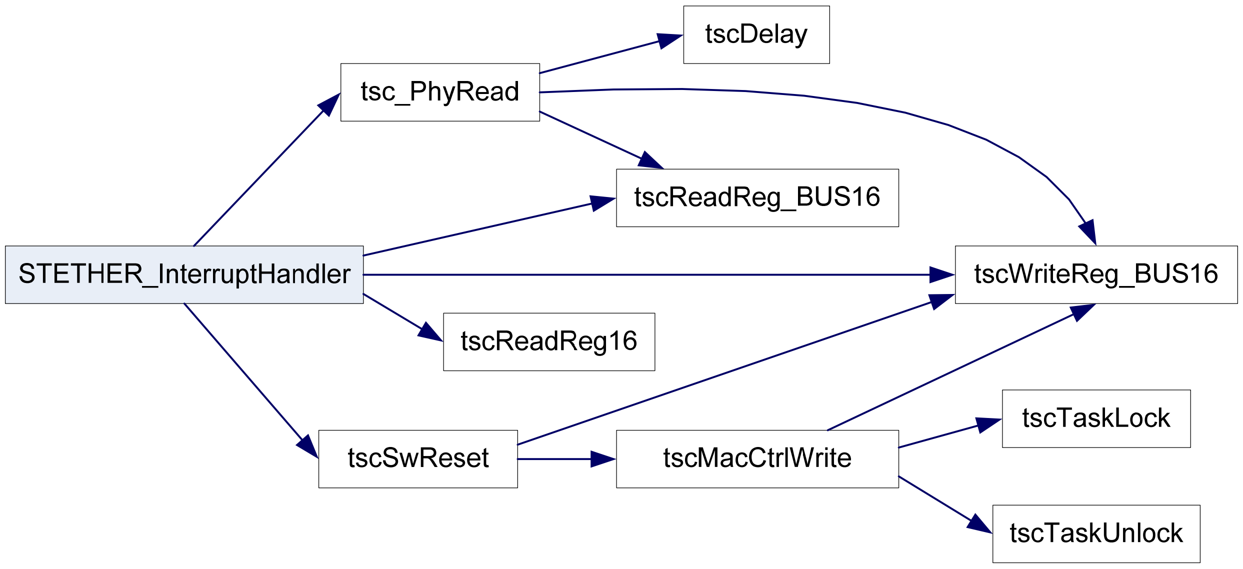

Figure 8: STETHER_InterruptHandler Call Graph

4.7 STETHER_Open ()

Prototype:

int STETHER_Open (NGifnet * *netp*)

Description:

Driver initialization begins with this function. It initializes the default configuration in the device control structure. It calls **tscDeviceInit()** to set up the hardware with the default configuration.

Parameters:

netp	network interface type NGethifnet_tsc

Returns:

NULL if OK.
err Error code in case of error.

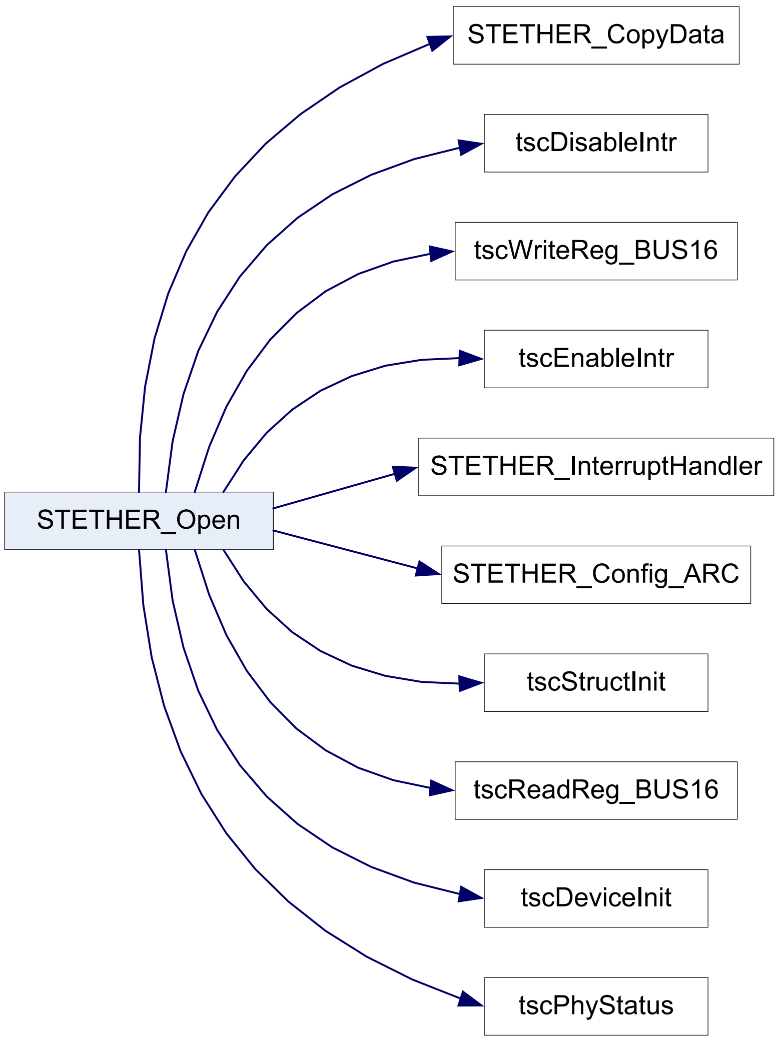

Figure 9: STETHER_Open Call Graph (First Level)

4.8 STETHER_Receive()

Prototype:
void STETHER_Receive(NGifnet *netp)

Description:
This routine checks for valid RX frames and copies their data from the 78Q8430 buffers into the system buffers.

Parameters:

netp	network interface type NGethifnet_tsc

Returns:
none

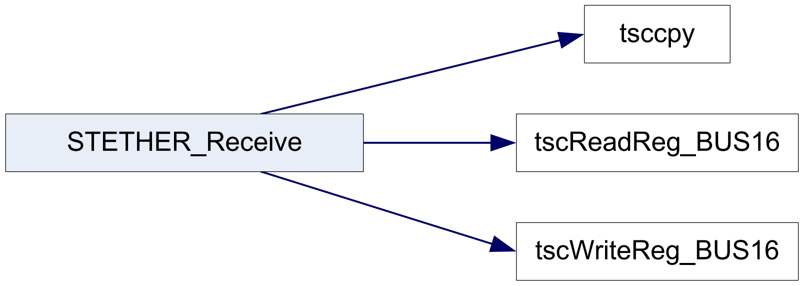

Figure 10: STETHER_Receive Call Graph

4.9 STETHER_Send ()

Prototype:
ST_ErrorCode_t **STETHER_Send**(NGifnet *netp, NGbuf *bufp)

Description:
This function sends new message buffer data to the device. It copies data from system memory to the 78Q8430 data buffers and then triggers a 78Q8430 transmit event.

Parameters:

netp	network interface type NGethifnet_tsc

Returns:
ST_NO_ERROR if no errors are encountered.
ST_ERROR_NO_MEMORY if no TX FDs are available from the driver core.

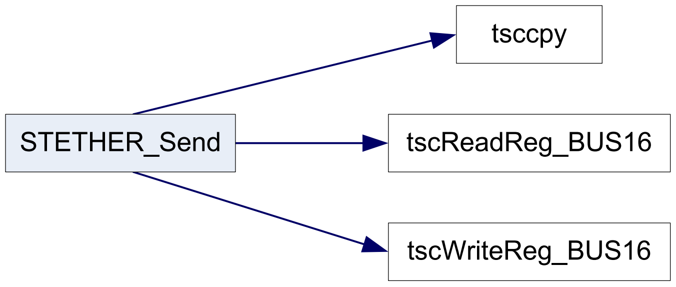

Figure 11: STETHER_Send Call Graph

4.10 STETHER_Start ()

Prototype:

void STETHER_Start(NGifnet *netp)

Description:

Driver initialization occurs in this function. It copies the first available outgoing packet to the 78Q8430. It then starts TX and un-queues any TX request that it might have processed.

Parameters:

netp	network interface type NGethifnet_tsc

Returns:

none

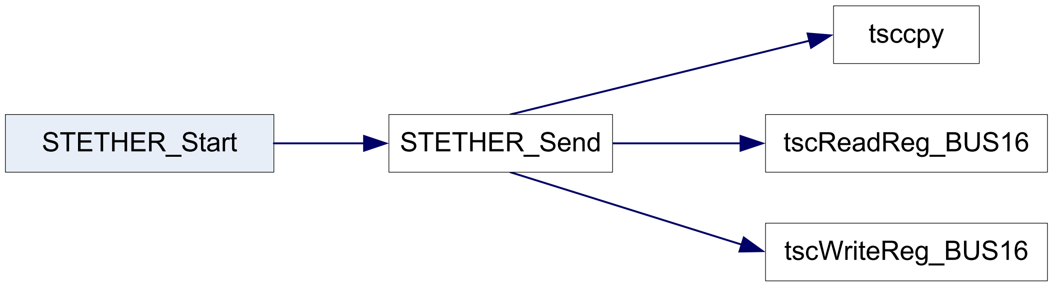

Figure 12: STETHER_Start Call Graph

5 STi5100 IPSTB Platform Example

The STi5100 IPSTB demonstrates the 78Q8430 Ethernet driver capability in an internet streaming video application. This section describes how to set up the platform, build the driver and NexGen software, and run an example which plays and stops an MPEG2 movie. Figure 13 shows the components and connections for the STi5100 IPSTB platform. Refer to the *78Q8430 STEM Demo Board User Manual* for additional information on the hardware setup.

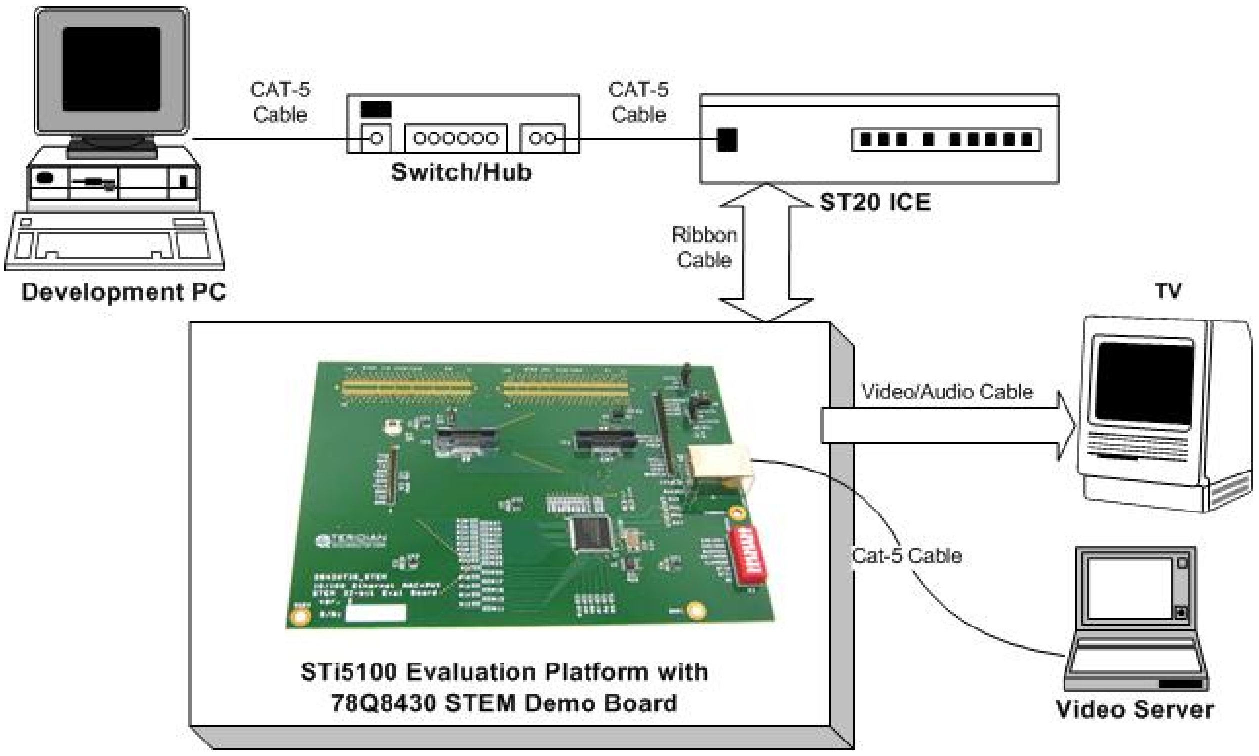

Figure 13: IPSTB Platform Block Diagram

5.1 Setup

The path names (in *italics*) given in the following steps are for illustrative purposes. If the software has been installed in different directories than those given below, replace the path in the example with the appropriate path for your installation.

5.1.1 Host PC Environment

Changes to the PC environment are needed when using the STi5100 platform or when changing between the STi5100 and STi5514 platforms. The STi5514 platform uses the old ST20R1.9.6 tool set while the STi5100 platform uses the newer ST20R2.0.5 tool set. Use the following procedure to modify the environment:

STEP 1: Modify the Windows Environment to use the ST20R2.0.5 tool set.
Append *C:\STM*\ST20R2.0.5\bin; to the front of the system variable 'Path'. As an example, the new path might be *C:\STM*\ST20R2.0.5\bin;*C:\STM*\ST20R1.9.6\bin;%SystemRoot%\system32;……

STEP 2: Modify the system variable 'ST20ROOT' as follows:

ST20ROOT *C:\STM*\ST20R2.0.5

STEP 3: Reboot the PC.

5.1.2 MPEG Video Server PC Environment

Use the following procedure to set up the video server PC:

STEP1: Set up the ST TSVOD server for unicast video streaming:
G:\Q8430\ST_IPSTB_SW\IPBox\servers\ TS_VOD_Server

STEP 2: Set up the ST Multicast server for multicast video streaming:
G:\Q8430\ST_IPSTB_SW\IPBox\servers\ Multicast

STEP 3: Set the video server PC IP address to 192.168.1.110.

STEP 4: Start the appropriate server before requesting a video stream.

5.1.3 ST Microconnect Target Configuration

The ST Microconnect Target Configuration IP addresses (via Ethernet) are as follows:

Host PC:	192.168.1.100
ST Microconnect:	192.168.1.30
Video Server PC:	192.168.1.110

The target configuration parameters are contained in the *C:\ipstba5*\config\platform\targets.cfg file. The target configuration portion of this file is as follows:

```
## Sample targets.cfg file
##
## Format of ST20 targets line:
##
##target lpt1        tap  "jpi_ppi lpt1"              board_runtime_init
##target jei-target   tap  "jei_soc jei-target-name"    board_runtime_init
##
## E.g.
##target myjei  tap   "jei_soc myjei tckdiv="    board_runtime_init
##target myjei2 tap   "jei_soc 10.1.1.1 tckdiv=" board_runtime_init
##
##
## End of sample targets.cfg
## lc
##target net tap "jei_soc 138.198.185.138 tckdiv=8" board_runtime_init
##
## david
##target net tap "jei_soc 138.198.185.133 tckdiv=4" board_runtime_init
##target usb tap "hti_usb usb tckdiv=4" board_runtime_init

## major
##target net tap "jei_soc 138.198.185.143 tckdiv=4" board_runtime_init

## hl
target jei108  tap "jei_soc 167.4.204.108 tckdiv=4" board_runtime_init
target jei112  tap "jei_soc 167.4.204.112 tckdiv=4" board_runtime_init
target jei62   tap "jei_soc 167.4.204.62 tckdiv=4" board_runtime_init
target jei96   tap "jei_soc 167.4.204.96 tckdiv=4" board_runtime_init
target jei99   tap "jei_soc 167.4.204.99 tckdiv=4" board_runtime_init
target jei110  tap "jei_soc 167.4.204.110 tckdiv=4" board_runtime_init
target jei111  tap "jei_soc 167.4.204.111 tckdiv=4" board_runtime_init

target tp5100 tap  "jei_soc 192.168.1.30 tckdiv=4" board_runtime_init
```

5.1.4 STi5100 IPSTB Configuration

The STi5100 configuration parameters are contained in the *C:\ipstba5*\config\board\mb390_mem.cfg file.

ST5100 FMI Bus Cycle Settings

The FMI Bus cycle settings in the file are shown below:

```
##-----------------------------------------------------------------------------
## Bank 2 - 32MBytes Stem1 Configured as 16-bit peripheral
##-----------------------------------------------------------------------------
## Parameters: -weuseoeconfig 0 -waitpolarity 0 -latchpoint 1 -datadrivedelay 0
##             -busreleasetime 2 -csactive 3 -oeactive 1 -beactive 2 -portsize 16
##             -devicetype 1

##             -cyclenotphaseread 1 -accesstimeread 1d -cse1timeread 2
##             -cse2timeread 0 -oee1timeread 0 -oee2timeread 0 -bee1timeread 0
##             -bee2timeread 0

##             -cyclenotphasewrite 1 -accesstimewrite 1d -cse1timewrite 2
##             -cse2timewrite 2 -oee1timewrite 0 -oee2timewrite 0 -bee1timewrite 0
##             -bee2timewrite 0

##             -strobeonfalling 0 -burstsize 0 -datalatency 0 -dataholddelay 0
##             -burstmode 0
##poke -d (STI5100_FMI_BANK2_DATA0) 0x001016D1  ##BE not active during rd
##poke -d (STI5100_FMI_BANK2_DATA1) 0x9d200000
##poke -d (STI5100_FMI_BANK2_DATA2) 0x9d220000
##poke -d (STI5100_FMI_BANK2_DATA3) 0x00000000

##ST IPSTB original settings
##  poke -d (STI5100_FMI_BANK2_DATA0) 0x041086e9
##  poke -d (STI5100_FMI_BANK2_DATA1) 0x0e024400
##  poke -d (STI5100_FMI_BANK2_DATA2) 0x0e024400
##  poke -d (STI5100_FMI_BANK2_DATA3) 0x00000000

##TTPMOD TSC modified settings

## Won't work with 7 cycles
## poke -d (STI5100_FMI_BANK2_DATA1) 0x87111100 ## 7 cycle Read, CSE1=CSE2=OEE1=OEE2=1
##  poke -d (STI5100_FMI_BANK2_DATA2) 0x87111100 ## 7 cycle Write, CSE1=CSE2=OEE1=OEE2=1

## The following timings work
##  poke -d (STI5100_FMI_BANK2_DATA1) 0x8C111100 ## 12 cycle Read, CSE1=CSE2=OEE1=OEE2=1
##  poke -d (STI5100_FMI_BANK2_DATA2) 0x8C111100 ## 12 cycle Write, CSE1=CSE2=OEE1=OEE2=1

##  poke -d (STI5100_FMI_BANK2_DATA1) 0x8A111100 ## 10 cycle Read, CSE1=CSE2=OEE1=OEE2=1
##  poke -d (STI5100_FMI_BANK2_DATA2) 0x8A111100 ## 10 cycle Write, CSE1=CSE2=OEE1=OEE2=1

##  poke -d (STI5100_FMI_BANK2_DATA1) 0x89111100 ## 9 cycle Read, CSE1=CSE2=OEE1=OEE2=1
##  poke -d (STI5100_FMI_BANK2_DATA2) 0x89111100 ## 9 cycle Write, CSE1=CSE2=OEE1=OEE2=1

  poke -d (STI5100_FMI_BANK2_DATA0) 0x04100691  ##LATCHPT=1, BUSRELEASE=1, CS/OE active R&W
  poke -d (STI5100_FMI_BANK2_DATA1) 0x88020202 ## 7 cycle Read, CSE1=OEE1=1,CSE2=OEE2=2
  poke -d (STI5100_FMI_BANK2_DATA2) 0x88020202 ## 7 cycle Write, CSE1=OEE1=1,CSE2=OEE2=2


  poke -d (STI5100_FMI_BANK2_DATA3) 0x00000000
```

ST5100 FMI Clock Rate Settings

The FMI clock rate settings in the file are shown below:

```
 ##TTPMOD TSC modified settings  ##
 ## sdram refresh bank 5
 ## flash runs @ 1/2 bus clk
 ## sdram runs @ bus clk
  poke -d (STI5100_FMI_GEN_CFG) 0x00000000
##  poke -d (STI5100_FMI_FLASH_CLK_SEL) 0x00000001  ##1/2 ST bus clock (50.4MHz)
##  poke -d (STI5100_FMI_FLASH_CLK_SEL) 0x00000002  ##1/3 ST bus clock (36MHz)
  poke -d (STI5100_FMI_FLASH_CLK_SEL) 0x00000000      ##1/1 ST bus clock (100.8MHz)
  poke -d (STI5100_FMI_CLK_ENABLE) 0x00000001
```

5.2 Build STi5100 IPSTB Code

5.2.1 Location of Source Files

The relevant TSC and NexGen source files required to build and run the STi5100 video streaming application example are listed below. The NexGen files include modifications to support the hardware checksum.

TSC Source Files

- Directory path: *C:\ipstba5*\src\nexgen_drv

ether_tsc78q8430.c	Wrapper file which include Teridian source files
tsccore.c	Core code
tscport.c	OS and H/W dependent code
tsctest.c	Test application code

- Directory path: *C:\ipstba5*\include

ether_tsc78q8430.h	Wrapper file which include Teridian header files
tscport.h	OS and H/W dependent headers
commem.h	Common memory, data structure declaration
comregs.h	Register declaration

NexGen Files with Hardware Checksum

- Directory path: *C:\ipstba5*\src\nexgen_drv

ipncs.c	Enable HW checksum usage in IP files
udpncs.c	Enable HW checksum usage in UDP files
tcpncs.c	Enable HW checksum usage in TCP files

5.2.2 Build the Software

Use the following procedure to build the software:

STEP 1: Use Windows File Manager to open a window.

STEP 2: Select the directory ***C:\ipstba5.***

STEP 3: Double click to execute the **ipstb_setup** link. A blue DOS window will appear displaying:

Using 5100ref root: *c:\ipstba5*
c:\ipstba5\src\ref_ipstb>

STEP 4: Make the new Ethernet device driver and NexGen code (create nexgen_drv.lib).

- Change to directory ***C:\ipstba5*\src\nexgen_drv.**
- gmake clean
- gmake

STEP 5: Make the new code for a complete IPSTB image (create ref_ipstb.lku).

- Change to directory ***C:\ipstba5*\src\ref_ipstb**
- gmake clean
- gmake

5.3 Run the STi5100 IPSTB Example

Use the following procedure to execute the IPSTB ref_ipstb.lku application to request an MPEG2 movie from the server:

STEP 1: From the Windows Start menu, select and execute **st20dev** (the ST20R2.0.5 tool set).

STEP 2: Select 'File' → 'Open Workspace' → '*C:\ipstba5*\ref_ipstb.stw'.

STEP 3: Make sure the ST Microconnect and STi5100 box are powered-up.

STEP 4: Select 'Build' → 'Start Debug' → 'Go'.

STEP 5: Select 'Debug' → 'Go'.
A second DOS window will appear displaying the Testtool prompt:

```
Testtool>
```

STEP 6: Request an MPEG2 movie from the server by entering the following command:

PlayManager_Play "rtsp://192.168.1.110:554/song?vidpid=34&audpid=33&Bitrate=1400"

STEP 7: Stop the movie using the following command:

PlayManager_Stop

The text file *C:\ipstba5*\Play_Commands.txt contains ready made commands with correct parameters to play various MPEG2 streams. Refer to the ST IPSTB user guide for more information on the PlayManager and IPSTB software.

6 Related Documentation

The following 78Q8430 documents are available from Teridian Semiconductor Corporation:

78Q8430 Preliminary Data Sheet
78Q8430 Layout Guidelines
78Q8430 Software Driver Development Guidelines
78Q8430 Driver Manual for ST 5100/OS-20 with NexGen TCP/IP Stack
78Q8430 STEM Demo Board User Manual
78Q8430 Driver Manual for ARM920T Linux
78Q8430 Embest Evaluation Board User Manual

7 Contact Information

For more information about Teridian Semiconductor products or to check the availability of the 78Q8430, contact us at:

6440 Oak Canyon Road
Suite 100
Irvine, CA 92618-5201

Telephone: (714) 508-8800
FAX: (714) 508-8878
Email: lan.support@teridian.com

For a complete list of worldwide sales offices, go to http://www.teridian.com.

Appendix A – Acronyms

API	Application Program Interface
Fast Ethernet	100Mbps Ethernet LAN as defined by IEEE 802.3u
FTP	File Transfer Protocol (RFC4823)
HTTP	Hyper Text Transport Protocol (RFC2854)
IP	Internet Protocol (RFC1112, also RFC0894)
MAC	Media Access Control IEEE-802.3
PC	Personal Computer
PHY	Physical
RAM	Random Access Memory
RMON	Remote monitoring MIBS – belong to SNMP protocol family
ROM	Read Only Memory
RTCP	Real-time Transport Control Protocol (RFC4571)
RTP	Real-time Transport Protocol (RFC4598)
RTSP	Real Time Streaming Protocol (RFC2326)
STB	Set Top Box (IP, Cable, Terrestrial, Satellite)
TCP	Transport Control Protocol (RFC3168, also RFC0793)
TCP/IP	TCP over IP protocols which is the core protocol for internet communications
TELNET	Network Virtual Terminal Protocol for terminal emulation (RFC0855)
TSC	TERIDIAN Semiconductor Corporation
UDP	User Datagram Protocol (RFC0768)
URL	Uniform Resource Locator (RFC1738)

Appendix B – Release Notes

Release Notes for the 78Q8430 ST/OS-20 Driver Version 1.01, date 03/07/2008.

The driver includes the following default settings:

- Strip CRC is on.
- Append CRC in Tx packet is on (the NexGen stack is customized for this).
- Jumbo packet support is off.

Release Package Contents

The software release includes the following components:

- 78Q8430 Software User Guide for ST/OS-20 (this document).
- 78Q8430 ST/OS-20 Driver source code (78Q8430_Drv_V1.01.zip).
  The readme file has 'how to' instructions to compile and build the 78Q8430 device drivers.
- ST Video Demo software integrated with the 78Q8430 driver (ipstba5.zip).
- Video server with demo videos (IPbox.zip).
  Contains the video server installation files and videos.

Software Build and Installation

Make configuration changes as needed in the tscport.h and commem.h files. In the ST video demo these files are in **C:/ipstba5/include/**.

Follow the procedures in Section 5.2.2 to build and install the STi5100 IPSTB software.

Changes from Previous Release

None (first release).

Known Problems

None.

Revision History

Revision	Date	Description
1.0	3/28/2008	First publication.