Galileo TECHNOLOGY

Communications Systems on Silicon™

APPLICATION NOTE
Initializing Ethernet Ports on the GT-642xx and GT-961xx Devices

25-Feb-01
Rev. 1.0
AN-66

1. Introduction

Galileo Technology's GT-642xx and GT-961xx devices are designed to support two independent 10/100Mb Ethernet ports[1]. The 10/100Mb Ethernet unit handles all functionality associated with moving packet data between local memory or PCI and the Ethernet ports.

This application note outlines the basic operation of the Ethernet ports and the steps involved in initializing the system to transmit and receive packets. Sample code for basic Ethernet port operation (which was written for the GT-961xx devices) is included with this document.

2. Ethernet Unit Description

2.1 System Initialization

Before using the Ethernet ports, initialize the Memory, set up the SDMA descriptor pointers, choose the Ethernet port parameters, set the internal routing registers and initialize the SDMA, as described in the following subsections.

2.1.1 Memory Initialization

Before initializing the descriptors, prepare a chained list of descriptors and packet buffers. The descriptor consists of four longwords, including the command and status values, the number of bytes, the location of the buffer in memory and a pointer to the next descriptor and status values, the number of bytes, the location of the buffer in memory, and a pointer to the next descriptor.

The following restrictions apply to the descriptors:

- Each descriptor must be 4LW in length, and it must be 4LW aligned (i.e. Descriptor_Address[3:0]=0000).
- Descriptors may reside anywhere in the CPU address space except for NULL address (0x00000000), which is used to indicate the end of a descriptor chain.
- The last descriptor in the linked chain must have a NULL value in its Next Descriptor Pointer field.
- Rx buffers associated with Rx descriptors are limited to 64K bytes and must be 64-bit aligned.
  Minimum size for Rx buffers is 8 bytes.
- Tx buffers associated with Tx descriptors are limited to 64K bytes and can reside anywhere in memory.
  However, buffers with a payload smaller than 8 bytes must be aligned to a 64-bit boundary.

For more information on initializing the descriptors, refer to the "10/100Mb Ethernet Unit: Operational Description" section in the GT-961xx or GT-642xx datasheet.

Refer to the following functions in the sample code:

SetRxDesc(DESC_RX0, DATA_RX0)	This function initializes the receive (Rx) descriptors.
SetTxDesc(DESC_TX0, DATA_TX0)	This function initializes the transmit (Tx) descriptors.

2.1.2 SDMA Descriptor Pointers

The Ethernet unit integrates powerful SDMA engines, which automatically manage data movement between buffer memory and the ports, and guarantee wire-speed operation on all ports (even when all ports are in 100Mb Full-Duplex mode). The Galileo Technology devices have two SDMA engines per Ethernet port—one dedicated for receive and the other for transmit. The SDMA logic handles multiple priority queues per port, providing support for priority-sensitive data in both directions. There are four receive priority queues and two transmit priority queues per port.

1. When the GT-642xx is configured to the RMII interface (instead of MII interface), it has three Ethernet ports.

The First and Current Descriptor pointers should point to the first longword (32 bits) of the first descriptor initialized by the *Set-RxDesc* and *SetTxDesc* functions (see sample code for additional instructions). The sample code uses the high priority queues and initializes the device to receive a packet on Ethernet 1, then send the packet back out the same port.

CTDP(E1,1)	Current Tx descriptor pointer of Ethernet 1, high priority queue.
CRDP(E1,3)	Current Rx descriptor pointer of Ethernet 1, highest priority queue.
FRDP(E1,3)	First Rx descriptor pointer of Ethernet 1, highest priority queue.

2.1.3 Ethernet Port Initialization

Each 10/100Mb port is fully compliant with the IEEE 802.3 and 802.3u standards and integrates the MAC function and a dual-speed MII interface. The port's speed (10 or 100Mbps) and duplex mode (Half- or Full-Duplex) can be auto-negotiated through the PHY and do not require user intervention. The port also features 802.3x flow-control mode for Full-Duplex and backpressure mode for Half-Duplex. The sample code sets the ports to Full-Duplex, turns on auto-negotiation, and, most importantly, enables transmit/receive on the Ethernet ports by initializing the following registers:

PCR(E1)	Port configuration registers for Ethernet 1
PCXR(E1)	Port configuration extend registers for Ethernet 1

2.1.4 Internal Routing and Pin Assignment

To keep the pin count of the Galileo Technology device as low as possible and the package as small as possible, many of the pins on the device are multiplexed. These pins can be used for various tasks. To use the Ethernet ports, the pins must be configured accordingly. Pins must be defined as input or output, and must be connected to the appropriate internal unit. In the sample code, Ethernet 1 is used, and specific pins are connected to this unit internally, by initializing the following registers:

MRR	**Main Routing register:** Defines which pins should be connected to which function (MPSC, Ethernet, general purpose, etc.). The sample code does not initialize this register because the default is set to map Port MII1 to Ethernet 1.
GPC2	**General Purpose Port 2 Configuration register:** This register determines if the multiplexed pins are used as I/O pins or as general purpose pins. The sample code sets all of port 2 to peripheral functions, thus setting the pins to MII1 (and MII0).
GPIO2	**General Purpose Port 2 I/O register:** This register configures the direction of the multiplexed pins as input or output. The sample code initializes these registers to set MII1 for transmit and receive, and ensure that the data nibble output is synchronous. A detailed description of each pin and its functions can be found in the "Pin Information" section of the GT-961xx and GT-642xx datasheets.

2.1.5 SDMA Initialization

After the Memory has been initialized, the SDMA descriptor pointers have been set, the Ethernet port parameters have been chosen, the internal routing registers have been set and the SDMA has been initialized, the SDMA command registers should be initialized. Initializing these registers tells the SDMA to begin arbitration between the queues and transmit/receive data. See sample code for more details.

SDCR(E1)	SDMA configuration register for Ethernet 1.
SDCMR(E1)	SDMA command register for Ethernet 1.

2.2 Ethernet Operation

When the steps outlined in Section 2.1 have been completed, the Ethernet ports operate as described in the following sub-sections.

2.2.1 Transmit from Galileo Technology Devices

The SDMA starts and performs arbitration between the transmit queues according to the value programmed in Port_Configuration_Extend<PRIOtx>. The SDMA then fetches the first descriptor from the specific queue it has decided to serve, and starts transferring data from the memory buffer to the Tx-FIFO. When either 384 bytes of packet data are in the Tx FIFO or the entire packet is in the Tx FIFO (for packets shorter than 384 bytes), the port initiates transmission of the packet across the MII. While data is read from the Tx FIFO, new data is written into the Tx FIFO by the SDMA. If a packet spans more than one buffer in memory, the SDMA fetches new descriptors and buffers as necessary.

When transmission has been completed, status is written to the first longword of the last descriptor. The Next Descriptor's address, which belongs to the next packet in the queue, is written to the current descriptor pointer register.

This process (starting with SDMA arbitration) is repeated as long as packets are pending in the transmit queues.

2.2.2 Receive to Galileo Technology Devices

The port waits for a receive frame to arrive at the MII interface. When this occurs, receive data is packed and transferred to the Rx FIFO. At the same time, an address filtering test is performed to determine if the packet is destined to this port. If the packet passes the address filtering check, a decision is made regarding the destination queue to which this packet should be transferred. When this is done, actual data transfer to memory takes place.

Note

Unless `Port_Control<HDM>` is set to pass addresses not found in the Address table or `Port_Control<PM>` is set, packets which fail address filtering are dropped and not transferred to memory.

For packets that span more than one buffer in memory, the SDMA fetches new descriptors as necessary. However, the first descriptor pointer is not changed until packet reception has been completed. Then status is written to the first longword of the first descriptor, and the Next Descriptor's address is written to the first and current descriptor pointer registers.

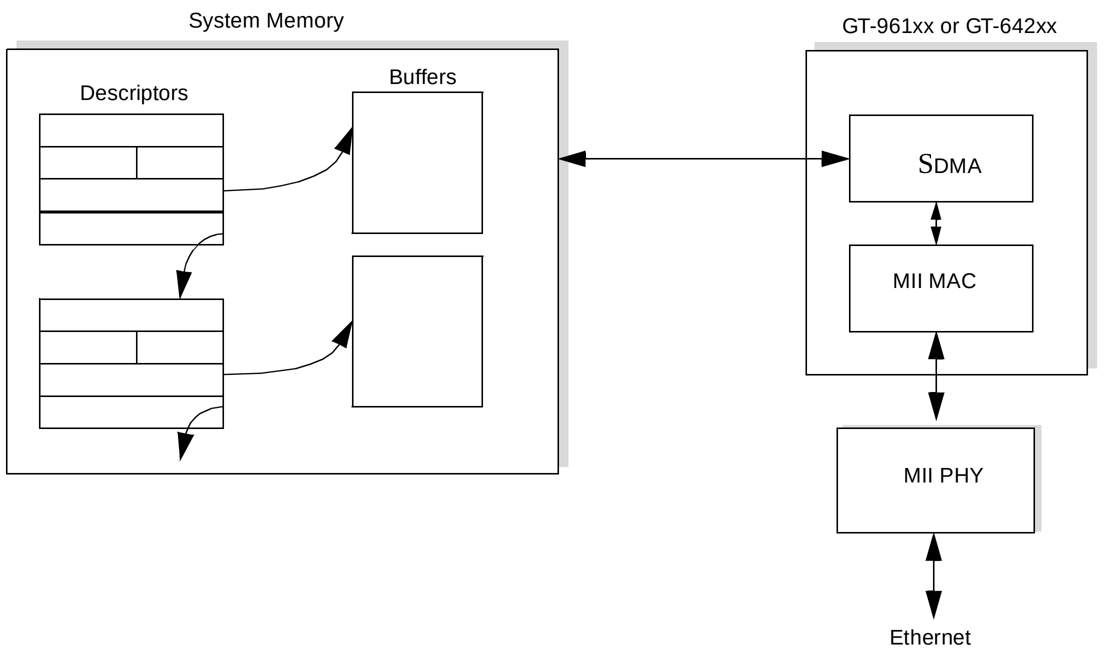
2.2.3 Resources Used

The Ethernet port interfaces directly to an MII (Media Independent Interface) PHY compliant with the IEEE 802.3u standard. The MII port provides a simple management interface capable of supporting both 10Mbps and 100Mbps data rates in Half- or Full-Duplex mode.

During transmit and receive operations, the SDMA accesses the MII MAC, which accesses the MII PHY on Tx, and the MII PHY accesses the MII MAC, which access the SDMA on Rx. If transmit and receive operations are enabled, the SDMA looks to system memory at the location pointed to by the descriptor pointers.

Figure 1 illustrates the interaction of the resources involved in the 10/100Mb Ethernet port activities.

Figure 1: System Resources Used During Ethernet Operation


System Memory
Descriptors
Buffers
GT-961xx or GT-642xx
SDMA
MII MAC
MII PHY
Ethernet