Computer Systems, Inc.
MERCURY

File Input Component

FM3TR Waveform Reference Implementation

SDR Forum Contract

March 23, 2007

Revision 1.0

Table of Contents

1 Component Name

FileInput_MAC_LLC

2 Component Processing Summary

The FileInput_MAC_LLC component reads an external file, divides it into blocks ("packets"), and pushes them to the next component. It improves upon the regular FileInput component by introducing media access control (MAC) and logical link control (LLC) interfaces such that packets may be retransmitted if necessary. It also encodes information about the file in a special header packet that the FileOutput_MAC_LLC component can interpret. This ensures that in the absense of a noisy channel, the output file will be received correctly.

3 Where used

The FileInput_MAC_LLC component is used in all extended data waveforms.

4 Data Input and Output Ports

The FileInput_MAC_LLC component has just one uses port, "FileInput_MAC_LLC_Out," which pushes a sequence of octets.

5 Control Interfaces

The FileInput_MAC_LLC inherits the control interfaces from CF::Resource.

Additionally, the FileInput_MAC_LLC component has three control interfaces:

`MAC_ErrCtrl_Input`	Packet error control feedback input interface
`MAC_FlowCtrl_Input`	Flow control input for adaptive data rate
`FileInput_MAC_LLC_LLC_Ctrl_Out`	LLC control output interface for FM3TR_Packetizer component

6 Component SCA Properties

Aside from the DLL execparams, the FileInput_MAC_LLC component has three properties. The table below lists these properties along with a brief description.

Simple Name	*CORBA Type*	*Description*
`Input_File_Name`	`string`	Name and path to the file to be read.
`Input_Data_Type`	`string`	Type of data to be read (includes "octet," "char," "short," "ushort," "long,"

		"ulong," "float," and "double").
Num_Data_Elements	ushort	Number of data elements to read at a time

These properties are the same as the regular FileInput component.

7 Component Attributes/Key Variables

Below is a list of several key variables to the FileInput_MAC_LLC component with a brief description of their purpose.

file_in	Input file
data_type	Type of data to be read from the file
data_size	Size of data to be read from the file
packetCounter	Number of packets currently read from the file
eof	Boolean value set to true of the end of the file has been reached
num_data_elements	Number of data elements in the file
m_bPauseFlow	Pause execution of Run() method
packetFeedbackCounter	Number of packets received
errorPackets	Running list of erroneous packets
retransmissionMode	Boolean value specifying if the entire file has been sent, and thus the retransmission of erroneous packets can begin

8 Processing Details

Packets are passed from the FileInput_MAC_LLC component in blocks of size Num_Data_Elements, as determined by the properties file. Data are usually passed to the RsBlockEncoder component in sequences of length 72 as this is precisely the size that the encoder requires. Furthermore, the input data are assumed to be ASCII which have only seven significant bits, a symbol the RsBlockEncoder can recognize.
The FileInput_MAC_LLC component improves upon the limitations of the regular FileInput component by introducing control interfaces for packet re-transmissions, encoding a special header to tell the receiver both the number of packets to expect as well as the number of elements in the last packet, and finally zero-padding the last block such that the sequence has exactly 72 elements. The methods to achieve this are described in brief below.

8.1 Method: AssembleHeader()

The RsBlockEncoder component expects data in blocks, each with 72 elements. An obvious problem will arise if the input file does not contain an integer number of 72: the RsBlockEncoder will not have enough data to process. The FileInput_MAC_LLC component fixes this problem by first sending a special header packet that includes information about the file, including the total number of packets and the number of elements in the last packet. The AssembleHeader() method parses the file to obtain this information and writes it to an output buffer. This header contains the following pieces of information:

- Calling platform address
- Called platform address
- Number of packets in the file
- Number of elements in the last packet

8.2 Method: AssemblePacket()

Because the FileInput_MAC_LLC component has the capability of retransmitting packets, it is necessary to access a specific packet. The AssemblePacket() method achieves this by rewriting the block of data in the output buffer corresponding to the appropriate packet.

8.3 Method: AssembleDummyPacket()

Due to the FM3TR specification that a block consists of exactly 10½ hops, an even number of blocks need to be pushed to the RsBlockEncoder component for the data to reach the RsBlockDecoder. This is because the RsHopEncoder must wait for its input buffer to fill (equivalent to a full "hop") before it can encode the data. An odd number of blocks implies that only half of the 11^{th} hop has reached the RsHopEncoder, which in turn implies that the RsBlockDecoder has only 10 full hops and not the necessary 10½. The FileOutput_MAC_LLC component will not receive the first block of transmitted data until that half hop finally reaches the RsHopEncoder. For this reason, it is necessary to flush the components' buffers with a full block of "blank" data which the receiver can ignore. This is necessary as the MAC and LLC extensions require that the number of acknowledgements (ACKs) and negative acknoweledgments (NAKs) be counted before proceeding, thus after each retransmission ad "dummy" packet must be transmitted to flush the components' buffers. The AssembleDummyPacket() achieves this. Although this seems inefficient at first, one must keep in mind that packet errors are often rare, and thus the additional overhead for packet retransmissions is not detrimental to the system throughput.

8.4 Method: SignalHighWatermark()

When SignalHighWatermark() is invoked by the MAC_XMIT component, data flow is paused.

8.5 Method: SignalLowWatermark()

When SignalLowWatermark () is invoked by the MAC_XMIT component, data flow is resumed.

8.6 Method: SignalEmpty()

When SignalEmpty() is invoked by the MAC_XMIT component, data flow is resumed.

8.7 Method: SignalNAK()

When SignalNAK() is invoked by the MAC_XMIT component and the erroneous packet is not a "dummy" packet, its ID stored to the `errorPackets` list for retransmission. Retransmitted packet are assumed to be received correctly unless SignalNAK() is invoked on its ID.