Mercury Computer Systems, Inc.

FM3TR Receiver Component

FM3TR Waveform Reference Implementation

SDR Forum Contract

March 23, 2007

Revision 1.0

Table of Contents

1 Component Name

FM3TR Receiver (FM3TR_WaveformReceiver)

2 Component Processing Summary

The FM3TR receiver dissembles packetized data by removing the overhead as well as separating the control information from the application data.

3 Where used

Both voice and data waveforms.

4 Data Input and Output Ports

The FM3TR receiver has one provides (“FM3TR_WaveformReceiverIn”) and one uses (“FM3TR_WaveformReceiverOut”) port. The input port accepts a sequence of octets representing one bit each. These data can be represented as either a hard bit (“0” or “1”) or a soft bit from the MSKDemodulator component.
blocks of data from either the CVSDEncoder or RsHopEncoder components, adds the appropriate packetizing overhead and pushes to the output data port. Both ports operate on char sequences.

5 Control Interfaces

The FM3TR receiver inherits the control interfaces from CF::Resource. Additionally, the component contains a control interface, “MAC_FlowCtrl_Out,” which is used for pausing data flow to components.

6 Component SCA Properties

Aside from the DLL execparams, the FM3TR receiver has no additional SCA properties.

7 Component Attributes/Key Variables

Below is a list of several key variables to the FM3TR packetizer with a brief description of their purpose.

`m_a_code`	This is a binary sequence (±1) that has good auto- and cross-correlation properties. The variable itself is actually an array of chars, 32-elements long. The “a” code is used only once and at the beginning of the frame so that the receiver to know when to start logging data.
`m_s_code`	Similar to `m_a_code`, `m_s_code` is a 32-element binary sequence with good auto- and cross-

	correlation properties. The “s” code is used many times by the packetizer to convey control information, including operational mode and hop rate. `m_inv_s_code` is simply the inverse of `m_s_code`.

8 Processing Details

The FM3TR waveform consists of two major sequences of data; the preamble, and the subsequent data frames. The folowing methods are used to synchronize and extract control and data information from the incoming bit stream.

8.1 Method: Acquire()

The Acquire() method simply extracts the preamble from the incoming bit stream. This synchronizes the receiver with the input sequence by searching for the a-code, and seven subsequence s-codes. It uses the Correlate() method to extract the codes from the bit stream.

8.2 Method: Correlate()

The Correlate() method simply compares two bit sequences for similar values. For an *N*-bit correlator, the output is

$$r_{xy} = \sum_{i=0}^{N-1} \left(-1\right)^{b_x[i]-b_y[i]}$$

8.3 Method: EatBits()

The EatBits() method is used to ignore bits associated with rise, fall, guard, and transition times. It advances the pointer in the input buffer and decreases the buffer length.

8.4 Method: ExtractHop()

The ExtractHop() method retrives 80 application data bits from the data hops.

8.5 Method: ReadMore()

Because packets arrive fragmented (a frame might be split into several CORBA sequences as they are passed from one component to the next), several methods might require more data before they can continue. The ReadMore() method is called from several methods, including ExtractHop(), InitCorrelator(), and EatBits(). It simply waits until a sufficient amount of data is put in the input buffer before continuing.

8.6 Method: Sync()

The Sync() method retrives two s-codes from the synchronization hop.