

Design Stellaris 2006 – Entry LM1740 Abstract

A Stellaris Based AIS Decoder

Short Text Description

As part of the global maritime safety system, ships over 300 tons are now required to broadcast their positions using the Universal Automatic Identification System, known as AIS. Transmissions in the marine VHF band are used to send both “dynamic” information - speed, position etc, and “static” information – name, callsign etc. This information is vital for safety at sea but is also very interesting to those, like me, who live near to the coast.

This project demonstrates how the AIS transmissions can be decoded in software using a cost effective Stellaris microcontroller and transferred to a host system using the standard NMEA protocol.

Embedded C, developed using the Keil μ Vision toolset, is used throughout and this project shows how this can be used to achieve fast interrupt-driven processing of synchronous data.

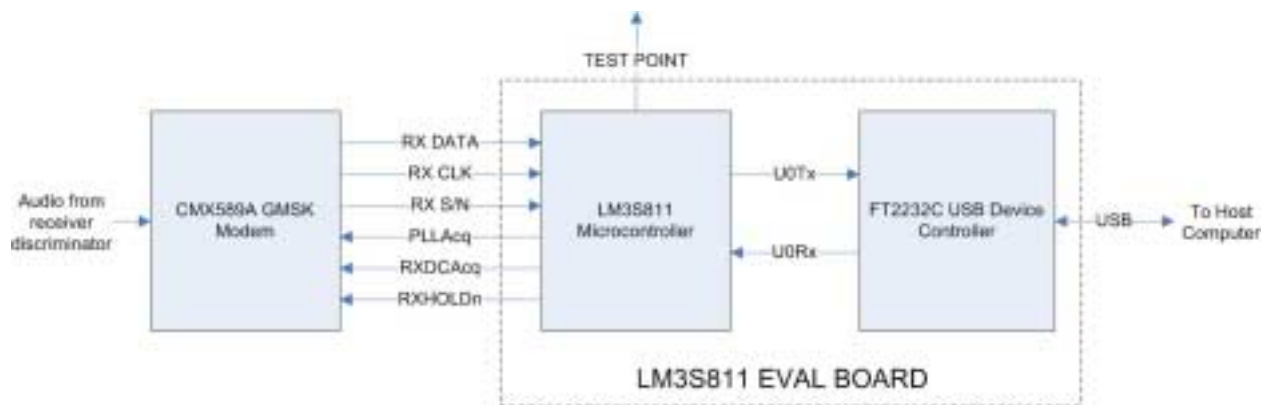


Figure 1 - System Block Diagram

The block diagram of my decoder is shown in Fig 1. As can be seen the hardware is very simple – A CMX589A GMSK modem IC is interfaced to the LM3S811 evaluation card via some GPIO pins. UART0 provides the interface with the host computer, via the USB virtual COM port.

The Stellaris microcontroller proved ideal for this task. I have demonstrated that the receiving of the data packets takes a small part of the available bandwidth of an LM3S811 running at 20Mhz, leaving plenty of time for further processing of responsive applications.

Photographs

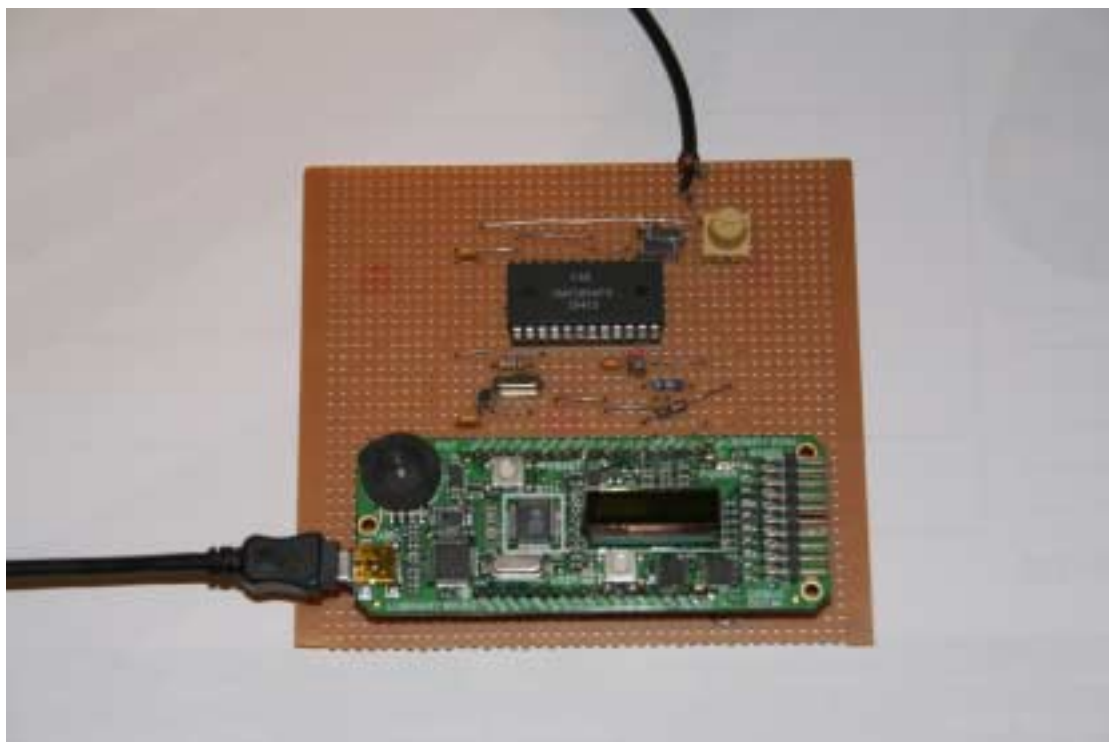


Figure 3 - The Decoder Board



Figure 4 - Icom IC-R7000 and Decoder Board

Code Sample

```
//
// This function checks if there is a packet waiting in the FIFO.
// If there is up to bufsize bytes of the next packet are transferred
// to the nominated buffer.
// Returns the number of bytes copied, zero if no packet
//
unsigned short FIFO_Get_Packet(unsigned char *buf, int16u bufsize)
{
    int16u pkt_start, pkt_end, pkt_size, bytes_copied;

    // Check if there's a packet available
    if (in_index == out_index)
        return 0;

    // Find the start and end of the packet
    pkt_start = pkt_index[out_index];
    pkt_end = pkt_index[(out_index + 1) & INDEX_MASK];

    // Calculate the size of the packet
    if (pkt_end > pkt_start)
        pkt_size = pkt_end - pkt_start;
    else
        pkt_size = (BUFFER_SIZE - pkt_start) + pkt_end;

    // Copy the packet out into the target buffer
    bytes_copied = 0;
    while ((bytes_copied < pkt_size) && (bytes_copied < bufsize))
    {
        *buf++ = buffer[pkt_start];
        pkt_start = (++pkt_start) & BUFFER_MASK;
        ++bytes_copied;
    }

    // Advance the out pointer
    out_index = (++out_index) & INDEX_MASK;

    // Return the number of bytes copied
    return bytes_copied;
} // FIFO_Get_Packet
```