SIMULATION AND RAPID PROTOTYPING ENVIRONMENT FOR AUV NETWORKS

Ethem M. Sözer, Milica Stojanovic,
MIT Sea Grant College Program
Cambridge, MA, 02139, USA
emsozer@mit.edu

***Abstract* – The interest in deployment of multiple autonomous underwater vehicles (AUVs), such as AUV swarms, has been increasing. AUV swarms can reduce mission completion times and increase the system reliability. One of the basic constraints of hese multiple vehicle systems is the limited communication between AUVs through the very challenging underwater acoustic channel. Even though reliable point-to-point communications through the acoustic channel can be successfully established, much research is needed to connect multiple AUVs within a network. In this paper, we present a simulation and prototyping environment for such networks.**

I. INTRODUCTION

Autonomous underwater vehicles (AUVs) have proved to be a valuable asset for conducting scientific and commercial missions with minimum cost and maximum efficiency. A new concept in AUV based operations is to deploy multiple vehicles, such as AUV swarms, to reduce mission completion time and system reliability. In such a system, multiple AUVs have to share information collected through the mission, make decisions based on the information, and rearrange resources if needed. The exchange of information is realized through acoustic communication systems that connect multiple AUVs through a dedicated data network. Even though the current technology provides reliable communications between two vehicles, much research is needed to form acoustic networks, which will enable reliable communications among an AUV swarm.

Unlike the radio channel, the underwater acoustic (UWA) channel does not have a widely accepted mathematical model. Currently, simulations are carried out employing modified radio channel models and experiments are the key process to validate any new algorithms. Experiments are usually performed over a point-to-point link by recording acoustic signals sent through a real channel. The data are then processed off-line using computers. This type of experiments restricts the validation of multi-user network algorithms. In some cases, over-the-counter modems are used to experiment with UWA sensory networks [1]. However, these modems have hard coded parameters that cannot be changed and development of software to implement additional network layers requires extensive engineering work.

We are developing an acoustic modem that will be flexible enough to test different communication algorithms including networking protocols. Due to its highly flexible structure, we call this modem the Reconfigurable Modem or the rModem. The main purpose of the rModem is to bring simulation and rapid prototyping environments together. By this way, algorithms developed by researchers and tested using simulation can be rapidly prototyped and proven in real world scenarios.

The development of the modem is carried out using MathWorks tools, such as Matlab®, Simulink®, and Real-Time Workshop®. Matlab has been the choice of the scientific community for developing new algorithms. We created a common simulation environment using Matlab® and Simulink®. Once the algorithms are tested using the simulation environment, we generate real-time code using Real-Time Workshop®. The generated real-time code can be run on a digital signal processor (DSP). Using a DSP based hardware platform, we can test the algorithms in real channels.

In the Simulink® environment, algorithms are defined using functional blocks. We can exploit this property and design a highly modular acoustic modem. A researcher can only focus on one of the functional blocks, say the equalizer, and develop a new algorithm. By simply changing the equalizer block in the reconfigurable modem, we can test this new algorithm and generate real-time code for experimental validation.

The rModem hardware has four major parts: the DSP board, analog-digital interface, power amplifier, and transducer. The DSP board contains a Texas Instruments TMS320C6713 chip. The analog-digital interface contains four analog-to-digital and digital-analog (AD/DA) channels, which will enable us to develop multi-input-multi-output (MIMO) [2] modems. The power amplifier board is able to drive different acoustic transducers with minimal engineering effort.

This work was performed as part of the project ``New Methods for the Exploration of Deep-Sea Hydrothermal Vents with Multiple Autonomous Underwater Robotic Vehicles," supported by the National Science Foundation (NSF).

In the next section, we will describe the hardware components of the rModem. In Section II, we will discuss the software components of the rModem. Section III and section IV are devoted to examples of simulations and experiments performed with the rModem. We will finish with conclusions and future directions.

II. RECONFIGURABLE MODEM HARDWARE

The reconfigurable modem hardware has four major parts: power supply carrier board, the DSP board, analog-digital interface, power amplifier, and transducer(s). All boards are compatible with the micro-line interface defined by Orsys Orth Systems, gmbh [3]. In the following we describe these hardware modules.

A. Power Supply Carrier Board

The first layer of the hardware boards is the power supply carrier board. This board functions as a base for a micro-line stack of DSP, analog-digital interface and the power amplifier boards. It delivers power to the micro-line stack and provides additional services such as a UART communications connector and a hard reset button.

B. DSP Board

The DSP platform is an of-the-shelf Orsys micro-line embedded development board. The micro-line board features a Texas Instruments TMS320C6713 DSP chip and a Xilinx Virtex-II FPGA. This board provides an open micro-line bus interface for integrating peripheral hardware directly with DSP or FPGA resources.

The TMS320C6713 is a 225 MHz floating-point DSP processor with a theoretical maximum performance of 1350 MFLOPS. We decided to utilize a floating-point processor to minimize the time required to convert simulation software into real-time code. The processing power of this DSP is enough to minimize the hand optimization effort for rapid prototyping.

The price we pay for high performance with floating point functionality is high power consumption as compared to the C5000 series low power DSP chips. As we intend to employ the rModem as a research-based rapid prototyping environment, we decided to choose ease of programming over low power consumption. We assume that these modems will not be employed for extended periods without maintenance or will be deployed within a system which does not have strict power consumption requirements for its peripherals, such as an AUV.

The micro-line C6713 compact board also features 64 Mbyte on board SDRAM as nonvolatile memory space together with a Resident Flash File system for easy software downloading and handling. The Resident Flash File system enables us to store multiple modem definitions in the on board memory and select the required definition at the boot time. Therefore, we can test multiple communication algorithms within one deployment without the need for multiple downloads.

Since one of the deployment platforms of the rModem is AUV, we paid special attention to the size of the system. The micro-line board dimensions are 120 x 67 mm (4.72" x 2.64"). The peripheral boards stack on the DSP board. The final size of the system depends on the number of peripherals.

C. Analog-Digital Interface Board

The analog-digital interface board features four analog-to-digital and digital-to-analog (AD/DA) channels. By employing multiple input and output channels, we will be able to develop and test multi-input-multi-output (MIMO) modems.

Each channel on this board can sample at 250 kHz and has a built in anti-aliasing filter with cut-off frequency at 100 kHz. However, we can program the board to provide us with a lower sampling rate, by decimating the signals in the on board FPGA. We can program the sampling rate of the A/D converter, the decimation rate, and the decimation filter coefficients. The same coefficients are used to interpolate the signals going to the D/A channels. The coefficients can be programmed during start up.

D. Power Amplifier Board

Power amplifier board will be placed on top of the analog-digital interface and drive the acoustic transducers. This board will utilize high-efficiency linear amplifiers. The amplifier gain can be controlled by the DSP, enabling us to experiment with power control algorithms for network optimization. The power amplifier board also hosts an automatic gain controller, which adjusts the gain of the receiver amplifier based on the commands sent by the DSP. The board can be reconfigured for different transducers by changing a couple of components without changing the basic design.

II. RECONFIGURABLE MODEM SOFTWARE

We are developing the rModem software using the Simulink® platform. Simulink® provides an environment where the rModem can be modeled in a block diagram fashion. Each block defines a separate task of the rModem, such as filtering, synchronization, or equalization. The rModem functional blocks can be tested by running simulations. In addition to simulations, using the Real-Time Workshop tool, we can convert the Simulink® block diagram into real-time C code. This generated code can be compiled and downloaded to our hardware using Code Composer Studio (Texas Instruments Development Environment).

We created custom blocks to model the rModem functionality. Each block is made up of three files: the s-function file, the task file, and the target language compiler (TLC) file. The s-function file defines the block properties such as the number and type of inputs and outputs, state information, and parameters. We define the relationship between the inputs and the outputs, or the behavior of the block, in a separate function called the task function. The task function defines the underlying algorithm of the block. In this way, we can wrap previously developed C code with s-functions and employ them in our rModem models. The TLC file is used to convert the block definition into C code.

MathWorks also provides the Stateflow® toolbox.

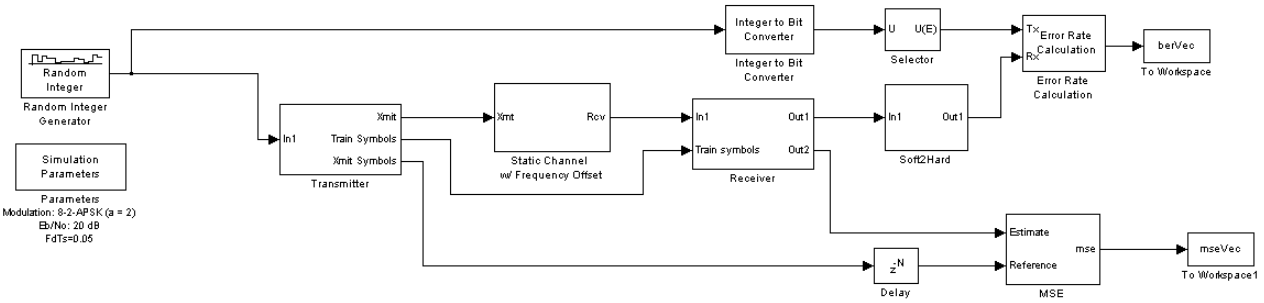


Fig. 1. We created a simulation model to investigate the performance of a linear equalizer with PLL under frequency offset with several APSK modulation schemes.

Stateflow® charts define state machines that can be used in the Simulink® environment. We use these charts to define the logical behavior of the rModem. State machines are especially useful in modeling network layers of a communication system.

Since the hardware development of the rModem is still in progress, we started the software development and testing using the TI C6713 DSP Starter Kit (DSK). This DSK combines the power supply board, DSP board, and the analog-digital interface by providing a voice codec. We will be able to use the same system model on the original hardware by only changing the hardware dependent driver blocks in the system.

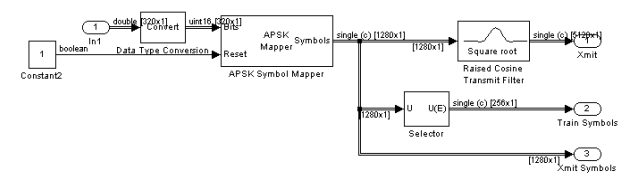
III. SIMULATIONS WITH RECONFIGURABLE MODEM

A complete communication system consists of many blocks such as synchronization, channel estimation, equalization, and coding. Simulating the behavior of the whole system and obtaining performance measures may become too complex and may be time consuming. The rModem simulator enables the researchers to focus on specific block of a system at a time due to its block based design. Fig. 1 shows the simulation model for an APSK [5] modulation system. The receiver employs a linear equalizer with LMS adaptations [6]. For this particular system, we chose a static multipath channel with frequency offset. With this model, we were able to test the sensitivity of the adaptive equalizer to frequency offset, which may be a result of Doppler shift or clock mismatch.

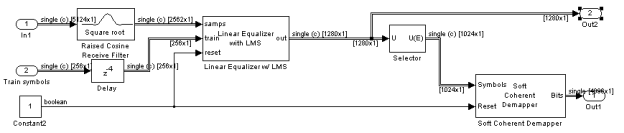
We present the details of the transmitter and the receiver in Fig. 2. The transmitter consists of an APSK modulator and a square-root raised-cosine interpolation filter. A number of symbols are selected as training symbols to be used at the receiver. The transmitter employs the same filter shape for decimation. The output of the adaptive filter is converted into soft bit estimates. We also utilize the output of the adaptive filter to estimate the mean square error performance. We employed the same model to test a decision feedback equalizer (DFE).

As an example case, we investigated the performance of a linear adaptive equalizer with PLL under frequency offset for several modulation schemes. A sample simulation result obtained using this model for BPSK, 4-1-APSK (equivalent to QPSK), 8-1-APSK (equivalent to 8PSK), 4-2-APSK, and 8-2-APSK is given in Fig. 3. The interpretation of the simulation results is out of this manuscripts scope.

Once we are pleased with the performance of the modulator and/or the equalizer, we can use the same blocks in the real-time system that we describe in the following section.



a) Transmitter



b) Receiver

Fig. 2. The transmitter consists of an APSK modulator and a square-root raised-cosine interpolation filter. A number of symbols are selected as training symbols. The receiver uses the same filter, an adaptive equalizer, and a symbol demapper.

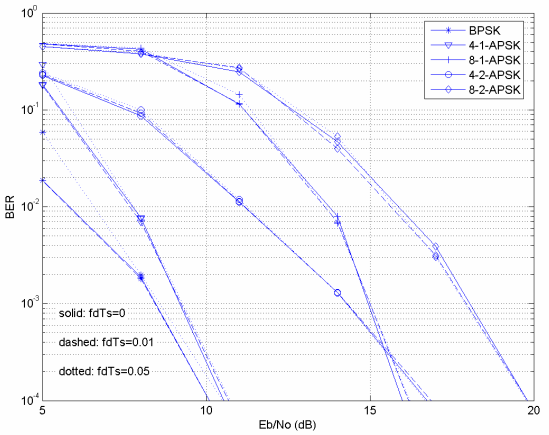


Fig. 3. Simulation results obtained using the rModem model presented in Fig. 8, where we investigated the performance of the linear equalizer with PLL under frequency offset.

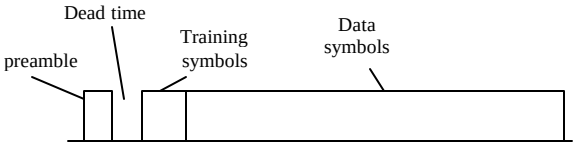


Fig. 4. The packet structure of rModem consists of a preamble, dead time, training symbols, and data symbols. Packets are divided into frames. In this version, we use a frame size of 256 symbols.

IV. RECONFIGURABLE MODEM REAL-TIME SYSTEM

In this section, we describe the modem structure that can be used in the real-time reconfigurable modem system. In a simulation environment, we can process a whole communication packet in one simulation time instance regardless of its size. Also, we can make sure that the transmitter and receiver are synchronized in time. However, in the real-time system, we have to process the communication signals in frames rather then packets due to the limited memory of DSP systems and stringent real-time constraints. At each clock tick, we process a frame worth of data.

The basic building blocks of a communication system, as we used them in the simulation environment, are simple algorithms that generate an output based on the input. To utilize these same blocks in a real-time system, we need to introduce some intelligence into the system. Most of the blocks in the real-time system are used to control the data flow through the modem based on this framed structure. With the real-time modem control blocks in place, a researcher insert the block that was designed for simple simulations into the appropriate section of the real-time system and obtain a prototype to test the performance of the new algorithm in real environments.

We defined a communication packet as a collection of a preamble, dead time, training symbols, and data symbols, as shown in Fig. 4. Packets are divided into frames. The frame size can be selected by the user. Currently, we use frames of 256 symbols. The physical layer sends one frame worth of data to the analog-to-digital converter at each clock tick. At the receiving side, the rModem processes one frame of symbols at each clock tick. If we select QPSK modulation, this means the demodulator block will output two times the frame size (or 512) bits.

Fig. 5 shows the highest level block diagram of the rModem. During the development of the rModem, we loosely followed the OSI layering structure [4]. This version of the rModem software defines the Physical Layer (or Layer 1), the Transport Layer (Layer 4), and the UART interface for serial communications with the modem. In the future versions, we will include the Network Layer (Layer 3) and the Data Link Control Layer (Layer 2). Each layer is connected to its higher level through two queues (or FIFO buffers), one for downstream communications and one for upstream communications.

A. Physical Layer

The physical layer consists of the transmitter, the receiver, controllers, and AD/DA converters, as shown in Fig. 6. The controller blocks are state machines that define the sequence of events during transmission and reception of acoustic packets. The AD/DA converter blocks are device drivers for the analog-to-digital and digital-to-analog converter hardware. During simulations they do nothing but pass the signals through, probably to a channel simulation block. The transmitter and receiver blocks enclose the basic

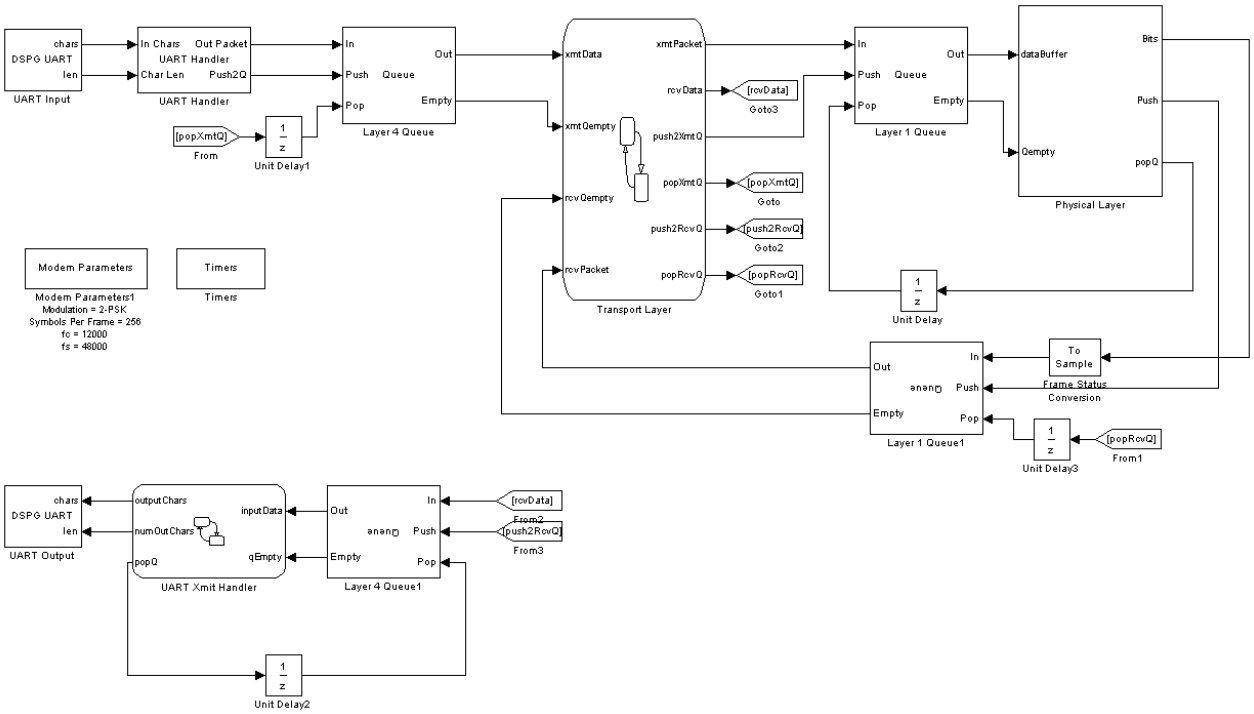


Fig. 5. Highest level block diagram for rModem defined in Simulink®.

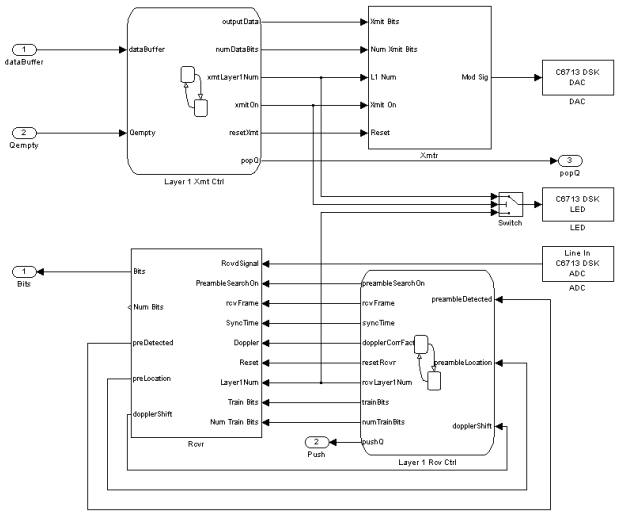


Fig. 6. The physical layer (Layer 1) of rModem consists of the transmitter, the receiver, controllers, and AD/DA converters.

blocks of the physical layer.

i. Transmitter

The transmitter handles the actual conversion of the bits into acoustic signals. Fig. 7 shows the block diagrams for the transmitter. The data bits are first passed through the convolutional encoder and encoded according to the coding scheme determined by the transmitter controller. The controller may indicate no coding, in which case the bits pass through the encoder block without any modification. The encoded bits are then interleaved and passed to a circular buffer. The circular buffer outputs one frame duration of bits every clock tick. These bits are passed through an interpolation filter. Finally, the signals are carrier modulated and sent to the D/A converter. Each block in the transmitter can be replaced with custom designed blocks to change the coding scheme, interleaver matrix, or symbol mapping.

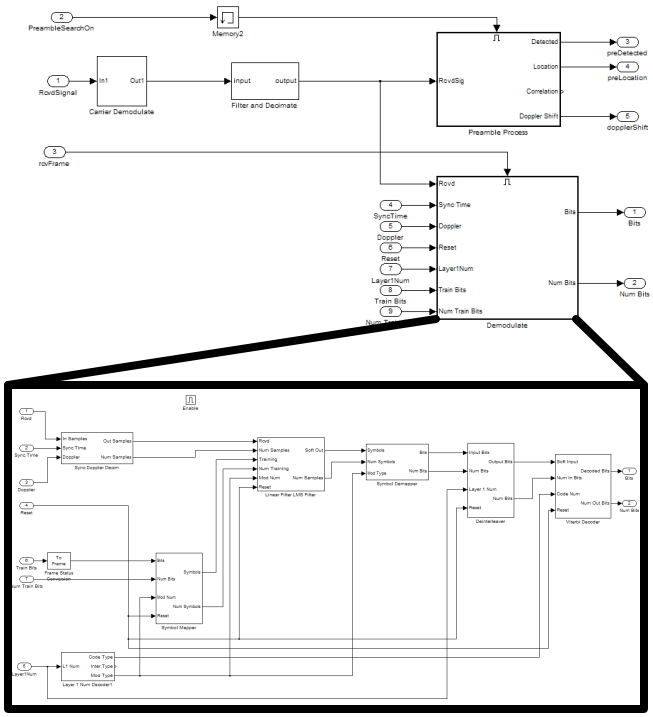


Fig. 7. The Xmit block converts data bits into symbols. The symbols are then are passed through an interpolation filter and carrier modulated.

ii. Receiver

The acoustic receiver consists of two major parts: the *Preamble Process* block and the *Demodulator* block (see Fig. 8). The samples received from the A/D converter are first down converted to baseband. The baseband samples are then passed through a decimator filter. The output of the decimator filter is fed to both the *Preamble Process* block and the *Demodulator* block.

When the rModem is not transmitting, the receiver controller enables the *Preamble Process* block. The received samples are further down sampled before correlating with the known preamble to reduce the computational cost. This block is also responsible for providing an estimate of the Doppler shift present in the received signal. If the correlation value exceeds a threshold, this block issues a detection signal together with the position of the preamble and the Doppler shift estimate.

Following the detection of a preamble, the controller disables the *Preamble Processor* block and enables the *Demodulator* block. The received samples are first passed through a Doppler compensator, synchronizer, and decimator. The output of the *Sync Doppler Decim* block is fed into the equalizer. The equalized symbols are demapped into soft bit values. The *Deinterleaver* block has an internal buffer where the soft bits of a data packet are buffered until the whole packet is received. Then the soft bits are deinterleaved and decoded in the *Viterbi Decoder* block.

B. Networking Layers

We define the networking layers as the layers of the OSI structure those are above the physical layer. We designed rModem to include the first four layers of the OSI structure. Layers above the transport layer, such as the application layer, are assumed to be handled by the host computer. The current version of the rModem does not include the data link control layer and the network layer. However, these layers can be easily implemented and included into the rModem structure by connecting them to the neighboring layers through queues.

We implemented a very simple transport layer. The transport layer divides the data received from the host computer into appropriate sized packets at the transmitting side. The receiving transport layer reassembles these packets and sends them to the host computer.

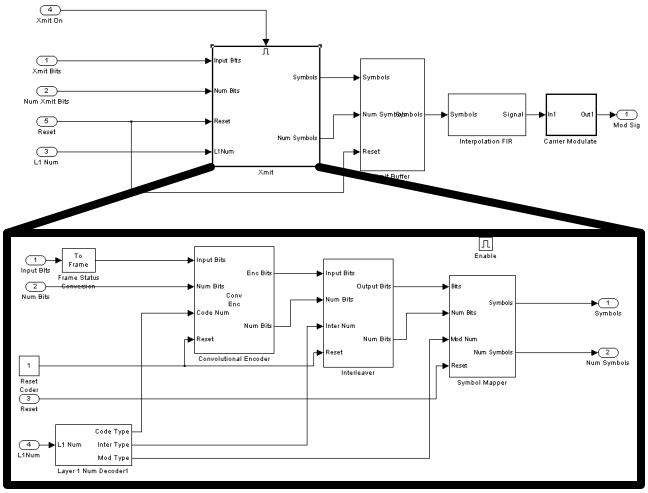


Fig. 8. The receiver first determines the packet presence based on the pre-amble. The receiver controller activates the receiver block, where the symbols are converted into data bits.

IV. EXPERIMENTS WITH RECONFIGURABLE MODEM

We used the APSK modulator, demodulator, and the equalizer blocks presented in the previous section to create a real-time modem. We processed data recorded during an experiment in the Aegean see in the summer of 2004 with this system. We generated real-time DSP code and downloaded the code to a TI 6713 DSK board. We used the sound card of a laptop computer to feed the experimental data into the rModem. After we finished the processing of a data packet, we stopped the DSP and examined its memory. Fig. 9 shows the screen capture of the DSP development program. With this program, we were able to obtain the scattering plot at the output of the DFE, the adaptive filter coefficients at the end of the packet, and the signal at the

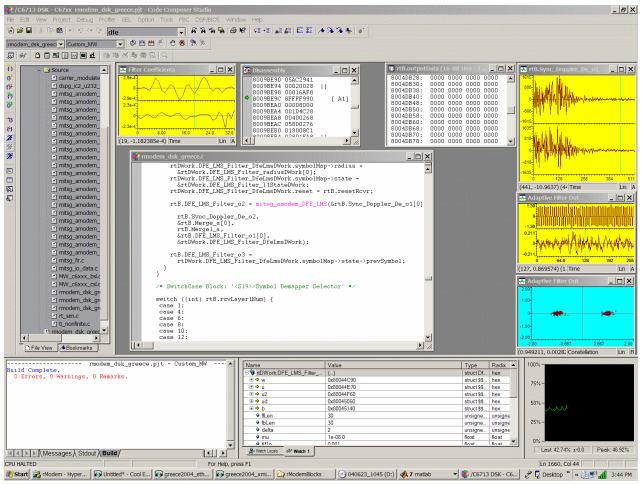


Fig. 9. This screen shot of the DSP development software is captured after the reception of a packet.

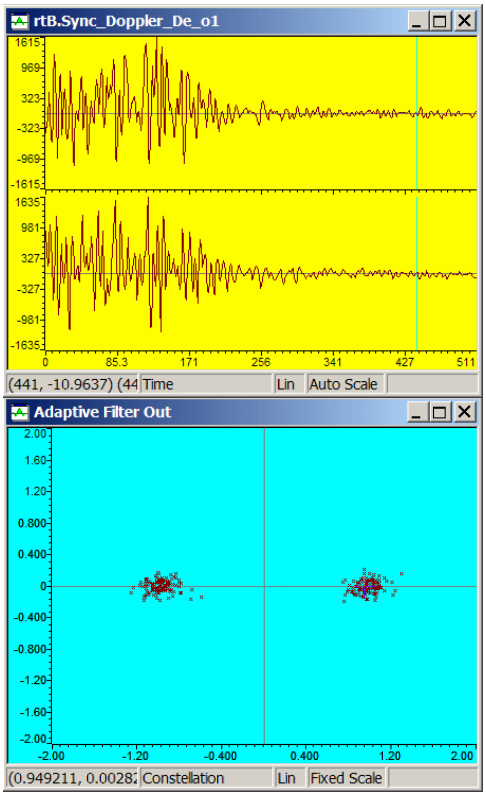


Fig. 10. The signal at the input of the DFE is shown in the upper plot, while the lower plot is the scattering plot for the last frame of the packet.

input of the DFE. Note that, these snapshots correspond to the last frame of the packet.

A closer look to the input received signal and the DFE output is given in Fig. 10. We can see that the DFE converged to a stable state where the scattering plot presented distinguishable symbol estimates.

V. CONCLUSIONS AND FUTURE DIRECTIONS

We are developing the initial rModem software as a basic communication system where information bits are encoded, interleaved and modulated at the transmitter and demodulated, deinterleaved, and decoded at the receiver. The underlying algorithms that we chose to carry out these tasks can be changed by replacing the corresponding block. At the time we started the development of the rModem, the communications blocks provided by Simulink® were general purpose blocks that either were not able to generate real-time code, or the generated code was inefficient. However, through our communications with the developers of Simulink®, we learned that they are working on providing blocks that will generate specific, efficient real-time code. These ready to use blocks will decrease the time to prototype communication algorithm even more.

We will continue to develop network layers for the rModem. We will customize the target files to enable one-click code generation, compilation, and download. We will conduct real channel experiments to test point-to-point links and networking algorithm.

REFERENCES

[1] R.K. Creber, J.A. Rice, P.A. Boxley, C.L. Fletcher, "Performance of undersea acoustic networking using RTS/CTS handshaking and ARQ retransmission," *Proc. MTS/IEEE Oceans Conf.*, pp. 2083-2086, Nov. 2001.

[2] Gesbert D., *et.al*, "From theory to practice: an overview of MIMO space-time coded wireless systems," IEEE JSAC, Vol. 21, pp.281-302, April 2003

[3] Orsys Orth Systems, gmbh, URL http://www.orsys.de/

[4] Bertsekas D. and Gallager R., *Data Networks*, 2nd Edition, Prentice-Hall Inc., 1992

[5] Chow, Y.C., Nix, A.R. and McGeehan, J.P., "Analysis of 16-APSK modulation in AWGN and Rayleigh fading channel," Electronics Letters, Vol. 28, pp. 1608-1610 Aug. 1992

[6] Haykin S., *Adaptive Filter Theory*, Fourth Edition, Prentice Hall, 2002