Rapid Development of Real-Time Applications Using MATLAB/Simulink on TI C6000-based DSP

JUAN ZAPATA and RAMÓN RUIZ
Universidad Politécnica de Cartagena
Department of Electrónica,
Tecnología de Computadoras y Proyectos
Campus Muralla del Mar
Spain

Abstract: This work discusses a hardware/software platform for educational purpose which integrates MATLAB/Simulink, Texas Instruments (TI) eXpressDSP Tools and C6000 digital signal processing (DSP) target let develop and validate digital signal processing designs from concept through code, in a typical professional vision design-simulation-implementation. This platform automates rapid prototyping on C6000 hardware targets because lets use Simulink to model digital signal processing algorithms from blocks in the Signal Processing Blockset, and then use Real-Time Workshop to generate C code targeted to the TI DSP board by mean Code Composer Studio (CCS IDE). The build process downloads the targeted machine code to the selected harware and runs the executable on the digital signal processor. After downloading the code to the board, our digital signal processing application runs automatically on our target. The library real time data exchange (RTDX) instrumentation that contains RTDX input and output blocks let transfer data to and from memory on any C6000-based target. The educational applications presented in this paper prove the feasibility of this methodology for this platform.

Key–Words: Real-Time Signal Processing, Real-Time Workshop, MATLAB, Simulink

1 Introduction

With the rapid evolution in semiconductor technology in the past several years, digital signal processing systems have a lower overall cost compared to analog systems. DSP applications can be developed, analyzed, and simulated using software tools [1] [2] [3]. There are two types of DSP applications, non-real-time and real-time. Non-real-time signal processing involves manipulating signals that have already been collected and digitized. Real time signal processing places stringent demands on DSP hardware and software design to complete predefined tasks within a certain time frame.

Unfortunely, the development of signal processing applications with real-time algorithms are still difficult and often requires specialized training in a particular assembly language for the specific DSP. Moreover, programming in assembly language gives the engineers full control of processor functions, thus resulting in the most efficent program for maping the algorithm by hand. However, this is very time consuming and laborious task, especially for today's highly paralleled DSP architectures.

Due to the fast-paced nature of the digital signal processing applications and to the limited life span of new products, the time to market (TTM) is a very important figure of merit that is often overlooked. The rapid realization of an implementation from its concept to a product is of utmost importance. The scientific community in general, and the signal processing community in particular, have developed a number of methods for the specification of higher level algorithmic concepts and ideas. Two equivalents alternatives are graphical methods and language-based methods. Graphical method includes block diagrams, state diagrams and schemes for the design of virtual prototypes and language-based method includes hardware description languages (HDLs). Simulink uses graphical block diagrams to create models for real-time implementation of applications and then use Real-Time Workshop to generate C code targeted to the TI DSP board by mean Code Composer Studio (CCS IDE).

This paper presents a platform based on a TI C6000 DSP target and Simulink/MATLAB/CCS can develop digital signal processing applications and as all these new technologies can be integrated in an easy and fast form.

2 Scheme Overview

The key aspect of rapid prototyping is automated code generation. Under our scheme, the algorithm for a given application is initially described with signal-flow block diagrams with Simulink. Simulink is a platform for multidomain simulation and Model-Based Design for dynamic systems. It provides an interactive graphical environment and a customizable set of block libraries, and can be extended for specialized applications. Models built in Simulink can be configured and made ready for code generation. Using Real-Time Workshop, C code can be generated from the model for real-time simulation, rapid prototyping, or embedded system deployment. Fig 1 shows the general scheme for rapid prototyping based on MATLAB/Simulink and Texas Instruments eXpressDSP tools.

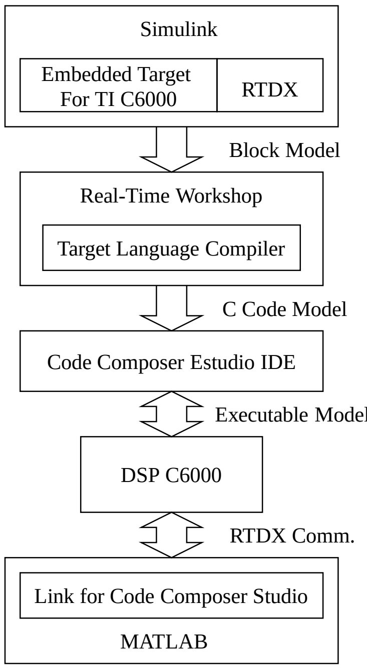
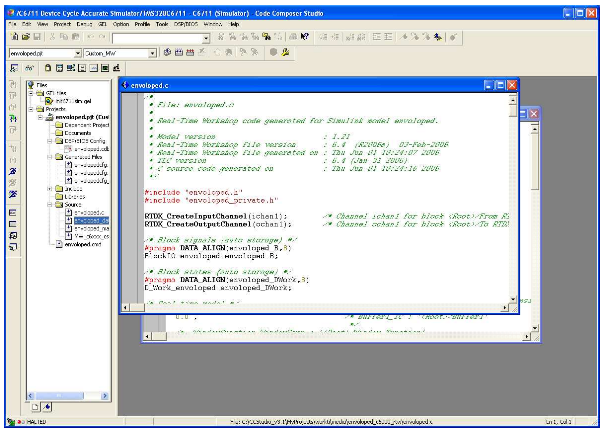


Figure 1: Development scheme of MathWorks and Texas Instruments eXpressDSP tools

Real-Time Workshop generates and executes stand-alone C code for developing and testing algorithms modeled in Simulink. The resulting code can be used for many real-time and non-real-time applications, including simulation acceleration, rapid prototyping, and hardware-in-the-loop testing. Real-Time Workshop uses target template files to translate Simulink models into ANSI/ISO C code [4] [5]. The target templates specify the environment on which this generated code will run. Own custom targets can be develop or use the ready-to-run configurations and third-party targets supported by Real-Time Workshop. Fortunelly, the Embedded Target for TI TMS320C6000 DSP [6] consists of TI C6000 target (C6000lib blockset) that automates rapid prototyping on a C6000 hardware targets.

Embedded Target for TI TMS320C6000 DSP integrates Simulink and MATLAB with Texas Instruments eXpressDSP tools. TI development tools pick the C code generated with Real-Time Workshop [7] for a customized hardware target supported for Embedded Target for TI TMS320C6000 DSP and build an executable file for this target-specific processor. Additionally, one of Real-Time Workshop build options builds a Code Composer Studio project from the C code generated and, therefore, all features provided by Code Composer Studio work to help develop the algorithm or application.


/C6711 Device Cycle Accurate Simulator/TMS320C6711 - C6711 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
enveloped.c

```
/*
 * File: envoloped.c
 *
 * Real-Time Workshop code generated for Simulink model envoloped.
 *
 * Model version                        : 1.21
 * Real-Time Workshop file version      : 6.4  (R2006a)  03-Feb-2006
 * Real-Time Workshop file generated on : Thu Jun 01 18:24:07 2006
 * TLC version                          : 6.4 (Jan 31 2006)
 * C source code generated on           : Thu Jun 01 18:24:16 2006
 */

#include "envoloped.h"
#include "envoloped_private.h"

RTDX_CreateInputChannel(ichan1);         /* Channel ichan1 for block <Root>/From R
RTDX_CreateOutputChannel(ochan1);        /* Channel ochan1 for block <Root>/To RTD

/* Block signals (auto storage) */
#pragma DATA_ALIGN(envoloped_B,8)
BlockIO_envoloped envoloped_B;

/* Block states (auto storage) */
#pragma DATA_ALIGN(envoloped_DWork,8)
D_Work_envoloped envoloped_DWork;
```

File: C:\CCStudio_v3.1\MyProjects\workti\medici\envoloped_c6000_rtw\envoloped.c

Figure 2: CCS IDE environment

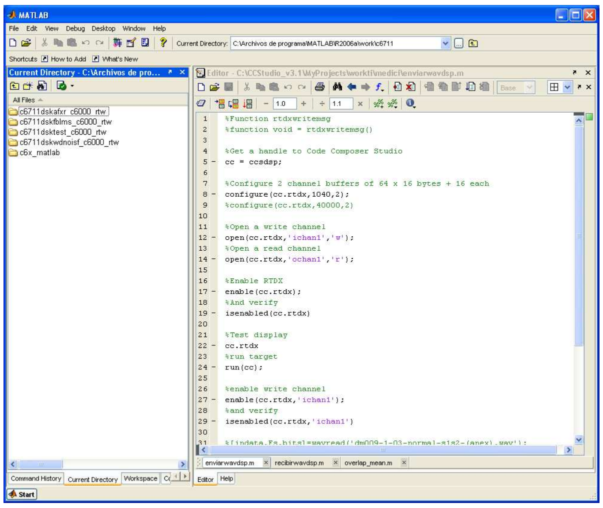
Once target-specific executable is donwloaded to the hardware and run it, the code runs wholly on the target and the running process can be accessed only from Code Composer Studio or from MATLAB with two powerful tools: Link for Code Composer Studio [8] and Real-Time Data Exchange (RTDX).

Link for Code Composer Studio lets use MATLAB functions to communicate with Code Composer Studio and with the information stored in memory and registers on a target. Figure 3 illustrates the main window of MATLAB. With the links, information can be transfered to and from Code Composer Studio and with the embedded objects, information about data and functions stored on the signal processor can be retrieved. Within the collection of hardware that Link for Code Composer Studio supports, some features of the link cannot be applied. This features or components are four:

- Link for Code Composer Studio IDE — Lets use objects to create links between CCS IDE and

MATLAB.

- Link for Real-Time Data Exchange Interface — Provides a communication pathway between MATLAB and the signal processor.
- Embedded Objects — Provides Object methods and properties that let access and manipulate information stored in memory and register on signal processor, or in Code Composer Studio project.
- Hardware-in-the-Loop — Enables to write functions in MATLAB that exercise functions from project on target processor. From MATLAB, data can be generated, can be send to target, and a function C can be used to manipulate the data in the hardware.


MATLAB
File Edit View Debug Desktop Window Help
Current Directory: C:\Archivos de programa\MATLAB\R2006a\work\c6711
Current Directory - C:\Archivos de pro...
c6711dskafxr_c6000_rtw
c6711dskfblms_c6000_rtw
c6711dsktest_c6000_rtw
c6711dskwdnoisf_c6000_rtw
c6x_matlab
Editor - C:\CCStudio_v3.1\MyProjects\workti\medici\enviarwavdsp.m
enviarwavdsp.m
recibirwavdsp.m
overlap_mean.m
Command History
Current Directory
Workspace
Editor
Help
Start

```
%Function rtdxwritemsg
%function void = rtdxwritemsg()

%Get a handle to Code Composer Studio
cc = ccsdsp;

%Configure 2 channel buffers of 64 x 16 bytes + 16 each
configure(cc.rtdx,1040,2);
%configure(cc.rtdx,40000,2)

%Open a write channel
open(cc.rtdx,'ichan1','w');
%Open a read channel
open(cc.rtdx,'ochan1','r');

%Enable RTDX
enable(cc.rtdx);
%And verify
isenabled(cc.rtdx)

%Test display
cc.rtdx
%run target
run(cc);

%enable write channel
enable(cc.rtdx,'ichan1');
%and verify
isenabled(cc.rtdx,'ichan1')

%[indata,Fs,bits]=wavread('dm009-1-03-normal-s1s2-(apex).wav');
```

Figure 3: The main MATLAB window

Real-Time Data Exchange provide a communication pathway to manipulate data and processing programs on the target digital signal processor. RTDX offers real-time data exchange in two directions between MATLAB and the digital processor. The general task flow for developing digital signal processing programs through RTDX include:

- Create an RTDX link to desired target and load the program to processor.
- Configure channels to communicate with the target.
- Run the application on the target and use MATLAB to investigate the results of running process.
- Close the links to the target and clean up the links and associated debris left over from your work.

3 System Configuration

In our scheme, the building process is initiated from Simulink, with a model or algorithm. Next, the Target language Compiler from Real-Time Workshop builds a program automatically for real-time application in C6000 environment. Using the make utility, Real-Time Workshop controls how it compiles and links the generated source code. Data can be sent, or received to the application through the RTDX channels.

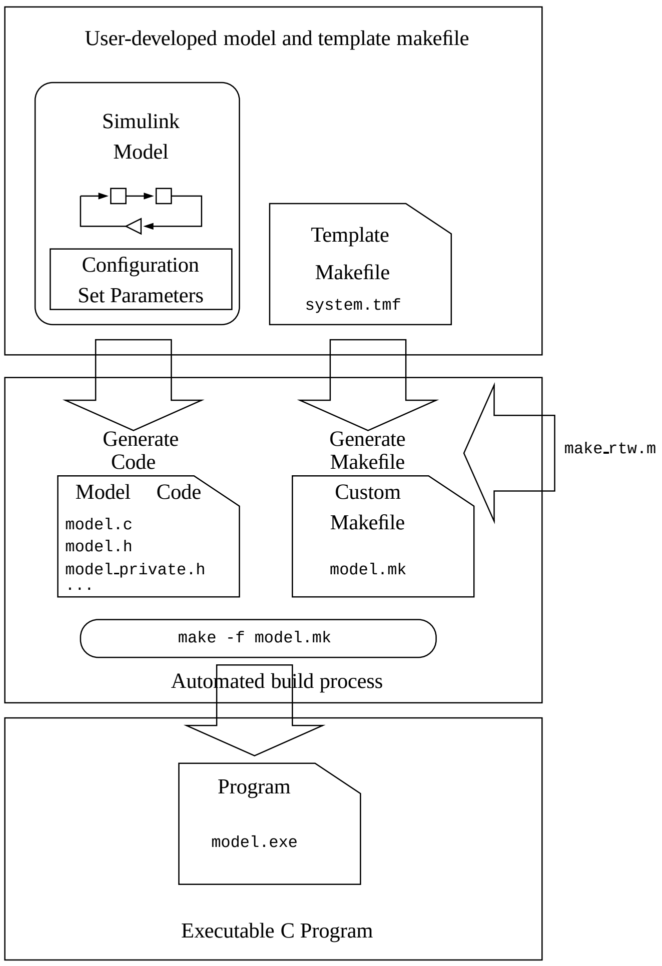


Figure 4: Automatic build process for Real-Time Workshop

3.1 Real-Time Workshop Configuration

Real-Time Workshop analyzes the block diagram and compiles an intermediate hierarchical representation in a file called `model.rtw`. The Target Language Compiler reads `model.rtw`, translates it to C code. Real-Time Workshop constructs a makefile from the appropriate target makefile template. The make utility reads the makefile to compile source code, link object files and libraries, and generate an executable image, called model (UNIX) or model.exe (Windows). Figure 4 illustrates the complete process. The box labeled ”Automated build process” highlights portions of the process that Real-Time Workshop executes.

After generating the code, Real-Time Workshop generates a customized makefile, `model.mk`. The generated makefile instructs the make system utility to compile and link source code generated from the model, as well as any required harness program, libraries, or user-provided modules.

Real-Time Workshop creates `model.mk` from a system template file, `system.tmf` and where system stands for the selected target name. The system template makefile is designed for a specific target environment. Exits the option of modifying the template makefile to specify compilers, compiler options, and additional information used during the creation of the executable.

Real-Time Workshop creates the `model.mk` file by copying the contents of `system.tmf` and expanding lexical tokens (symbolic names) that describe a model’s configuration. Real-Time Workshop provides many system template makefiles, configured for specific target environments and development systems.

During the final stage of processing, Real-Time Workshop invokes the generated makefile, `model.mk`, which in turn compiles and links the generated code. On PC platforms, a batch file is created to invoke the generated makefile. The batch file sets up the proper environment for invoking the make utility and related compiler tools. To avoid unnecessary recompilation of C files, the make utility performs date checking on the dependencies between the object and C files; only out-of-date source files are compiled. Optionally, the makefile can download the resulting executable image to target hardware.

To generate embedded C code, there are some constraints on model blocks. It requiers that all blocks in the model are either discrete time block or continous time block but can be sampled at discrete time. Moreover, if multiple sample rates are used in a system, it requires that the lowest sample will be chosen as the base rate and other higher sample must be multiple time of the base rate. The purpose of these constraints is obtain a C code with real-time scheduling support. This C code is in legacy main program and subroutine style, an interrupt service routine (ISR) to implement the real-time algorithm.

3.2 RTDX Communication Channels

Once time the code is downloaded and runned on a specific hardware, Real-Time Data Exchange (RTDX) can enable real-time bi-directional communication between C6000 hardware and a host application using MATLAB. Fig 5 shows the general communication scheme based on RTDX library.

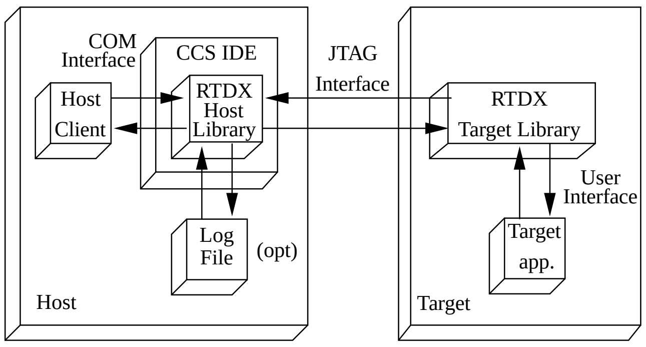


Figure 5: RTDX communication scheme

This host application open and enable communication channels through the Link for Code Composer Studio and the Code Composer Studio IDE. Openings channels consists of opening and configuring each channel for reading or writing, and enabling the channel. The following steps describe the basic instructions that enable MATLAB to read data messages from a target application [9]:

1. Declare the function.
   `function void = h_readmsg;`

2. Get a handle to Code Composer Studio IDE.
   `cc = ccsdsp;`

3. Get a handle to RTDX channel “ochan”.
   `rtdx_ochan = cc.rtdx;`

4. Enable RTDX.
   `rtdx_ochan.enable;`

5. Open the RTDX channel for reading.
   ```
   open ( rtdx_ochan, ’ochan’,
   ’r’);
   ```

6. Read data within a loop until no more data is available.
   ```
   tout_msg = ’Timeout’;
   NOMOREDATAMSG = ’No more data is
   available’;
   errmsg = NaN;
   while(isempty(findstr(tout_msg,
   errmsg)))
   try
   data = (readmsgrtdx_ochan,’ochan’,
   ’int32’);
   disp( data );
   catch
   ```

```
errmsg = lasterr
disp( NOMOREDATAMSG);
break;
end
end
```

7. Close the RTDX channel.
   `close( rtdx_ochan, ’ochan’ );`

8. End Function.
   `return;`

In the same manner, the following steps describe the basic instructions that enable to write data messages to a target application:

1. Get a handle to Code Composer Studio IDE.
   `cc = ccsdsp;`

2. Get a handle to RTDX channel “ichan”.
   `rtdx_ichan = cc.rtdx;`

3. Enable RTDX.
   `rtdx_ichan.enable;`

4. Open the RTDX channel for writting.
   ```
   open ( rtdx_ochan, ’ichan’,
   ’w’);
   ```

5. Write data to the target application.
   ```
   writemsg( rtdx_ichan,
   ’ichan’,int32(data) );
   ```

3.3 TMS320C6000 DSK

The choice of DSP-based hardware permits the development of a powerful and highly flexible real-time system. Due to its full programmability and high performance, the TMS320C67xx processor mounted on the TMS320C6000 DSK is a suitable device for researching and testing DSP algorithms [10]. In the work described in this paper, a personal computer was equipped with the Code Composer Studio development environment which helps to construct and debug embedded real-time DSP applications. It provides tools for configuring, building, debugging, tracing and analyzing programs. Texas Instruments DSP’s provide on-chip emulation support that enables Code Composer Studio to control program execution and monitor real-time program activity. Communication with this on-chip emulation support occurs via an enhanced JTAG link. This link is a low-intrusion way of connecting into any DSP system. Emulator interface provides the host side of the JTAG connection.

The heart of the TMS320C6000 DSK is the Texas Instruments TMS320C6711 processor. The C6711 is based on a VLIW-like architecture which allows it to execute up to eight RISC-like instructions per clock cycle.

The two data paths of the C6711 extend the functionality of the data paths of the C6201 with support for 64-bit data and IEEE-754 32-bit single-precision and 64-bit double-precision floating-point arithmetic. Each data path includes a set of four execution units, a general-purpose register file, and paths for moving data between memory and registers. The four execution units in each data path comprise two ALUs, a multiplier and an adder/subtractor which is used for address generation. The ALUs support both integer and floating point operations, and the multipliers can perform both 16x16-bit and 32x32-bit integer multiplies and 32-bit and 64-bit floating point multiplies. The two register files each contain sixteen 32-bit general-purpose registers. To support 64-bit floating point arithmetic, pairs of adjacent registers can be used to hold 64-bit data. In addition to the operations supported by the C6201, the C6711 offers support for floating-point reciprocal and reciprocal square root estimation, and for converting data between fixed- and floating-point formats.

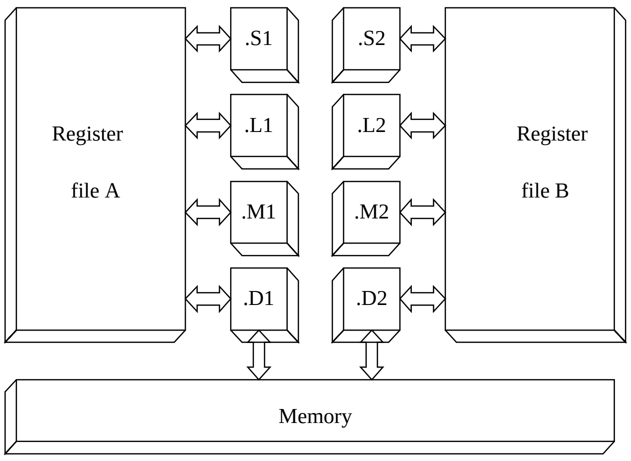


Figure 6: TMS320C67xx’s core

4 Development of Real-Time Applications

The rapid prototyping platform has been assessed at our laboratory and found to be well suited to develop real-time signal processing algorithms. In this section we describe the development of two applications for real-time processing using the platform described in this paper: a median filter and the envelop of a signal. In both cases, the process of design and implementation is the same. First the Simulink model is created with two RTDX channel for communication. Then, a project is built and compiled in the CCS IDE through Real-Time Workshop. This executable file is downloaded in the C6000-based target. Finally, data can be transfered in real-time from MATLAB using two programs based on RTDX instrumentation.

4.1 Median Filter

Fig. 7 shows a median filter. This model is created using elementary blocksets from Simulink libraries. The median filter is a non-linear filter that is often used in digital signal processing to smooth signals while at the same time preserving edges. To get the result of the filter a window is slided over the signal to filter. Data are ordered in ascending order. The Median Filter block replaces the central value of window with its median value. If the neighborhood has a center element, the block places the median value there.

Once the median filter model was simulated by passing a glitched signal and its performance to filter was checked then the real-time C code for a specific target was generated through Real Time Workshop. Before two channels for RTDX communication were added. In this process, the executable application was built and donwloaded automaticaly by CCS IDE in the DSP target. At last, real-time digital data was send for digital processing through two programs based on RTDX instrumentation. Processed data were received for the other channel in the same way for visualitation with differents tools of MATLAB.

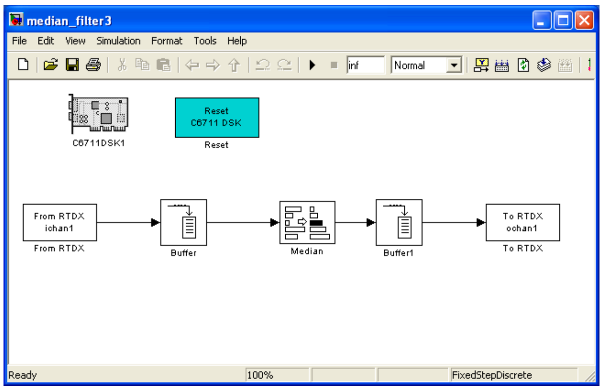


Figure 7: Median filter Simulink Model

This is not an impediment to effective use of tools provided for Code Composer Studio IDE. Moreover, these tools may sometimes be necessary to use, following the tradional practice of development of real-time applications using Code Composer Studio IDE exclusively. CodeComposer Studio provides tools for configuring, building, debugging, tracing and analyzing programs. Texas Instruments DSP's provide on-chip emulation support that enables Code Composer Studio to control program execution and monitor real-time program activity.

4.2 Envelope of a signal

Fig. 8 shows the envelope of a signal. There are different methods of calculating the envelope of the normalized signal. In this case, we have used a moving average filter, with a window length of 60 ms and with a overlapping between windows of 57 ms.

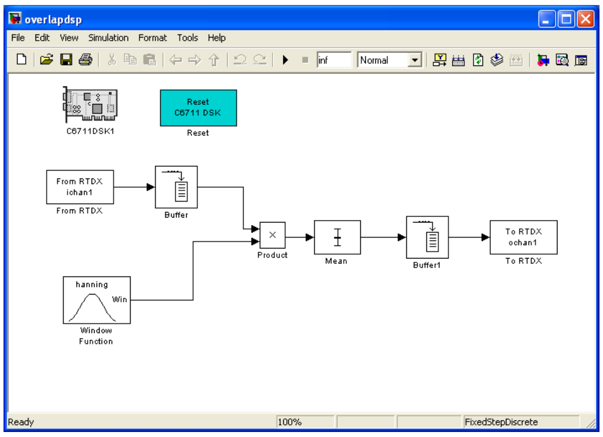


Figure 8: Signal envelop Simulink model

A sample of envelope of the normalized signal is calculated as

$$G(m) = \frac{1}{nP_1} \cdot \sum_{i=1}^{nP_1} y(i) \cdot w(i)$$

where $m = 1 \ldots, M$ and M is the sample number of the envelope of signal. nP_1 is the sample number of the window (60 ms in our case), and w is a Hanning window.

Once this model was tested and verified, then this algorithm was implemented and built it in DSP target. Now real-time analysis can be performed on the application runninga and we can use graphical tool provided in the MATLAB environment. A real heart sound wav file was sent out from host towards DSK and the calculated envelope was sent in from DSK to host through of two channels RTDX.

Finally to point out that the C code generated by this rapid prototyping platform is not the most efficient. To obtain a better code and increase the performance, there are techniques to improve and modify the C code generated but these techniques are not rapid pricessly. Finally to point out that the C code generated by this rapid prototyping platform is not the most efficient. To obtain a better code and increase the performance, there are techniques to improve and modify the C code generated. However, these techniques are not rapid and easy pricessly because designers must manually optimize the generated code in the code Composer Studio IDE. Alternatively, the code can be optimized by modifying the corresponding blocks of Simulink model and using others blocks from the preoptimized C62x and C64x libraries. When the code is generated, the Embedded Target for TI C6000 DSP produces function calls to preoptimized assembler implementations of the blocks, increasing the efficency and performance of critical zones of real-time application.

5 Conclusion

We have discussed in this paper a design methodology of real-time algorithms and applications based on cutting edge like MATLAB/Simulink, Code Composer studio IDE, and EVM/DSK hardware based on C6000 DSP of Texas Instruments. The purpose of this work is to provide an easy and efficient method for rapid prototyping and development for real-time applications. The paper described and illustrated the most important point to consider during the application of this technology. This technology lets develop and validate digital signal processing designs from concept through code, in a typical professional vision design simulation implementation. Moreover, the above discussion illustrated the use of the rapid prototyping system is a fully automated program building process, where the system is tested prior to the generation of the executable file. This saves tremendous amount of time, besides reducing the hardware cost.

In the continued effort to train more DSP engineer, this type of technology incorporates an added profit to the formation of new experts in this knowledge area and can help to speed up the learning curve and implementation of real-time DSP applications.

Another important benefit is that it avoids low level hardware work that can be tedious and very time consuming and therefore designers can focus their efforts in another importants aspects of design of real-time applications. This work describes the step sneeded to write and RTDX host application using MATLAB and the Developer's Kit for Texas Instruments DSP. Finally we illustred this process with some applications presented in this paper and its feasibility is proved.

Acknowledgements: This work has been supported by Fundación Séneca of Región de Murcia and Ministerio de Ciencia y Tecnología of Spain, under grants PB/63/FS/02 and TIC2003-09400-C04-02, respectively.

References:

[1] M. Chacon and I. Valenzuela, "Fast image processing application development scheme for the dsk c6711 using matlab and simulink," in *Digital Signal Processing Workshop, 2004 and the 3rd IEEE Signal Processing Education Workshop. 2004 IEEE 11th*, 1-4 Aug. 2004, pp. 79–83.

[2] F. Assis de Melo, M.A.; Leonardi and A. La Neve, "Digital signal processing with matlab and dsp kits," in *Digital Signal Processing Workshop, 2004 and the 3rd IEEE Signal Processing Education Workshop. 2004 IEEE 11th*, 1-4 Aug. 2004, pp. 15–18.

[3] W.-S. Gan and S. M. Kuo, "Teaching dsp software development: from design to fixed-point implementations," *IEEE Transactions on Education*, vol. 49, no. 1, pp. 122–131, Feb. 2006.

[4] W. Gan, "Teaching and learning the hows and whys of real-time digital signal processing," *IEEE Transactions on Education*, vol. 45, no. 4, pp. 336–343, Nov 2002.

[5] R. Chassaing, *DSP Applications Using C and the TMS320C6x DSK*. New York: John Wiley & Sons, 2002.

[6] *Embedded Target for TI TMS320C6000*, The MathWorks, Inc, March 2006.

[7] *Real-Time Workshop*, The MathWorks, Inc, March 2006.

[8] *Link for Code Composer Studio Development Tools*, The MathWorks, Inc, April 2006.

[9] D. Allensworth, "How to write an rtdx host application using matlab," Texas Instruments, Tech. Rep., May 2002.

[10] *TMS320C6000 Peripherals Reference Guide*, Texas Instruments, February 2001.