12th 2008 IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications

RTPROC: A SYSTEM FOR RAPID REAL-TIME PROTOTYPING IN AUDIO SIGNAL PROCESSING

Hauke Krüger and Peter Vary
Institute of Communication Systems and Data Processing
Aachen University, Templergraben 55, D-52056 Aachen, Germany
{krueger,vary}@ind.rwth-aachen.de

Abstract

In this contribution a new system for the rapid development of real-time prototypes for digital audio signal processing algorithms on Windows PCs and a Digital Signal Processor (DSP) platform is presented. The goal of this system is to enable even unexperienced developers to transform the conceptual idea of a new algorithm into a standalone real-time demonstrator as quickly and conveniently as possible. In order to achieve this goal, a software architecture is defined where hardware and algorithm related programming issues are separated to allow the algorithm developer to completely focus on the implementation of the algorithm only.

Compared to the earlier version [7], the new system supports all real-time prototype development phases from first Matlab simulations to the final highly efficient implementation in fixed point arithmetic and covers all relevant aspects such as e.g. control interface generation, function verification, complexity measuring and real-time data tracking. Development of prototypes based on general purpose PCs (`RTProcPC`) and the ADSP-21369 EZKIT [1] embedded DSP target (`RTProcDSP`) are currently supported (Fig. 1). The system has been successfully used to implement various real-time prototypes such as noise reduction, acoustic echo compensation, and digital hearing aid simulation.

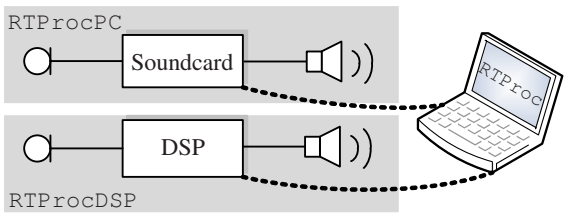


Figure 1. Audio signal processing real-time demonstrator based on RTProc.

1. Introduction

The conventional development of a product applying a new algorithm in audio signal processing commonly can be divided into three development phases as depicted in Fig. 2.

Development Phase I: Algorithm Investigation
First investigations on a new algorithm in general involve the usage of high level programming languages such as Matlab. Existing libraries are employed for algorithm exploration. Since the algorithm is operated with audio samples in offline manner, aspects such as computational complexity do not play a significant role yet.

Development Phase II: Real-Time Evaluation
In phase II, the algorithm found in phase I is tested under real-time conditions[1]. A highly efficient programming language such as C/C++ is chosen since processing efficiency is crucial to reach the required program execution speed. In order to identify a suitable realization of the new algorithm, further aspects have to be taken into consideration such as portability to fixed point arithmetic, algorithmic delay, and computational complexity.

Development Phase III: Product Integration
In the final development phase, the new algorithm is realized on the target product platform, e.g., a low power fixed point DSP or microcontroller (MCU). This phase can be well prepared in phases I and II and, in most cases, is not in the main focus of the algorithm developer.

While phase I is related to the development of high level programming language software only, a realization of the algorithm to be tested and optimized under real-time conditions involves a lot of programming effort to connect the new algorithm to audio hardware and to create a control interface which allows to adapt algorithm parameters during runtime. In praxis, during phases I and II, due to the different programming languages, at least two versions of the

[1] Since the Windows Operating System is not a real-time operating system, the term *real-time* refers to soft real-time conditions in `RTProcPC`.

1550-6525/08 $25.00 © 2008 IEEE
DOI 10.1109/DS-RT.2008.42

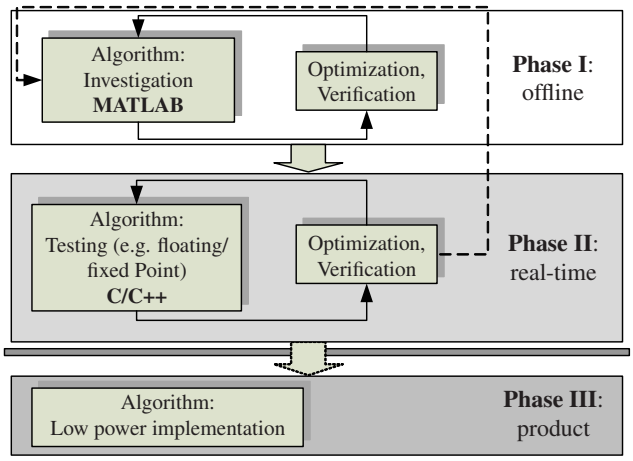


Figure 2. Phases in the conventional development of real-time audio signal processing prototypes.

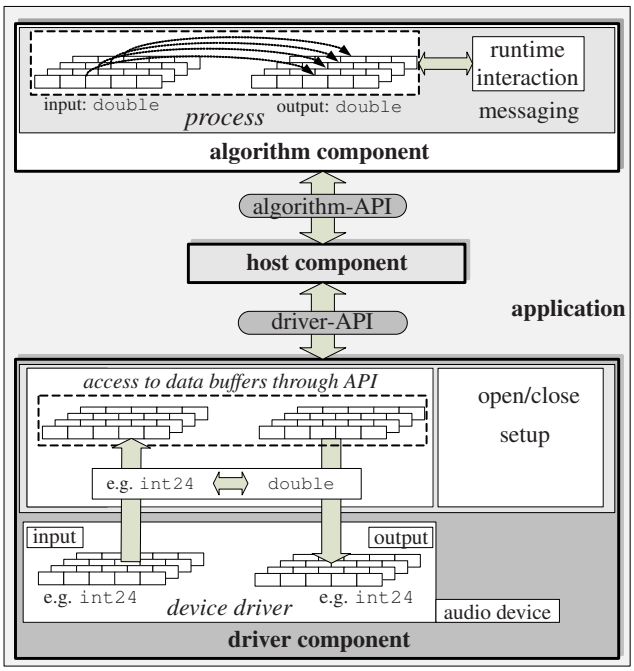


Figure 3. Basic principle of RTProc.

algorithm exist. Each version is verified and optimized in each of the phases independently. The synchronization of both versions (indicated by the dotted arrow in Fig. 2) is difficult and time consuming. In summary, the conventional development is inefficient and prohibits short development intervals.

The goal of RTProc is to simplify this process to reduce development time. Due to a clear separation of functional components in RTProc, all hardware related programming aspects are hidden from the developer, and almost no additional effort is required to switch from offline to real-time processing. In addition, an easy-to-use mechanism enables to attach control interfaces to the prototypes in a very convenient way. Due to an integration of the real-time functionality in Matlab, in RTProc aided development, phases I and II from Fig. 2 are merged to overcome the problems related to different versions for offline and real-time processing. Moreover, tools are provided to measure and optimize the timing behavior during real-time processing and algorithm efficiency in terms of computational complexity. At the end of the development with RTProc, the resulting real-time prototype is a demonstrator that shows all benefits of a new algorithm in a realistic application scenario and, henceforth, is the most convincing argument to market new solutions.

2 Principle of RTProc

RTProc is based on a software architecture realized in C/C++ that composes audio signal processing applications into three functional units, the **driver**, the **algorithm** and the **host** component, as depicted in Figure 3.

The Driver Component
The driver component has access to the audio input/output device. In praxis, the mechanism to interact with the hardware differs depending on the target platform and, on the PC, the chosen software library. In order to provide one unique interface for all applications, the driver component maps all basic hardware and system specific functionality to a simple interface, the RTProc driver application programmable interface (driver-API). During real-time processing, the driver component converts all hardware specific into hardware independent data types and grants access to the audio samples for further processing through the driver-API.

The Algorithm Component
The algorithm component has access to the audio buffers prepared by the driver component. Its main purpose is to process the input audio samples according to the algorithm to be realized. The resulting output samples are written to the reserved output buffers. The algorithm component is completely independent from the rest of the audio processing application and the hardware. The *runtime interaction* block in Fig. 3 enables to modify processing parameters and therefore the algorithmic behavior during audio processing.

The Host Component
The host component controls the driver and the algorithm component through the algorithm- and the driver-API. It is the key element of all audio processing applications and

routes all user interaction to the other two components. During audio signal processing, it transfers all audio data between driver and algorithm component.

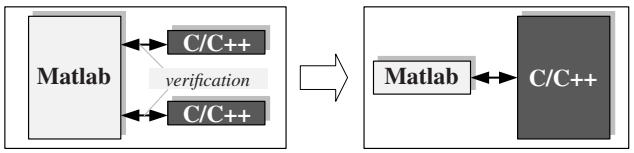
In RTProc aided development, the algorithm developer realizes the new algorithm as an algorithm component independently from any hardware aspects. Since driver and host components have been implemented as part of RTProc already, it is sufficient to focus on the algorithmic functionality only.
On the PC platform (RTProcPC), supported driver components are based on the **ASIO** [11], the **directSound** [9] and the **directKS** [10] technology to enable multichannel audio processing with system latencies of less than 5 ms. Two different host components are available, one to integrate real-time processing in Matlab (the **Matlab host**), and a generic library to offer the opportunity to add RTProc to customized applications (the **generic host**). The **GUI host** application makes use of the generic host component and provides a graphical user interface (GUI) to control real-time audio processing.
The embedded DSP based version of RTProc, RTProcDSP, targets applications which require even lower system latency and hard real-time. The driver and the host component are both part of a proprietary operating system in this case. A serial communication link (RS-232, [6]) enables runtime user control of parameters and algorithm verification from within Matlab in analogy to RTProcPC.

3 RTProc Aided Algorithm Development

RTProc aided algorithm development benefits from the merge of the development phases I and II. The developer starts algorithm exploration in Matlab. Targeting real-time audio processing with low latency, a Matlab program is not fast enough. In order to use RTProc and at the same time benefit from Matlab, the RTProc Matlab host component offers an offline mode in which audio signals from the Matlab workspace are fed into the algorithm component and Matlab and C/C++ functions can be mixed arbitrarily.

In general, at the beginning of the development, most functionality is realized in Matlab. In order to approach a real-time version of the algorithm, all functionality parts are ported to C/C++ step-by-step as demonstrated by the examples in Fig. 4 for an early and an advanced development stage, respectively. Implementation failures, which are often very time consuming in conventional development, are avoided since each single function can be easily verified against its Matlab counterpart.

Once all functionality is implemented in C/C++ and verified, the algorithm can be operated in real-time. The Matlab host is simply switched from offline into real-time mode to use audio samples from the sound device rather than the



a) Early development stage b) Advanced development stage

Figure 4. Porting from Matlab to C/C++ in RTProc, examples for an early (a) and an advanced (b) development stage.

samples from the Matlab workspace. Processing parameters such as filter coefficients can be calculated in Matlab offline and directly used in real-time processing.
Adding a runtime user control (GUI) can be done in a very convenient way: Instead of dealing with GUI libraries such as, e.g., QT [2], in RTProc, parameters to be adapted during runtime are described in a software construct, in Fig. 3 shown as the *runtime interaction* block. Based on this description, the GUI is generated dynamically by the host. Consequently, the specification of the runtime control is independent from the host and guarantees that the same processing parameters can be controlled, no matter which type of host is currently used. Given a simple description of the runtime control parameters in a configuration file, the RTProc Generic Runtime Compiler generates all required code automatically. Once a first version of the algorithm in real-time is available, the following tools are provided with RTProcPC to profile and optimize the code:

- The RTProc Complexity Instrumentation Tool helps to transform standard C/C++ code into an instrumented version based on which the complexity of the algorithm can be measured in terms of number of weighted operations per second. The weight of each processing instruction can be easily adapted to simulate different types of DSP instruction sets.
- The RTProc Full Speed Data Logger enables to store huge amounts of data produced during real-time processing on hard drive without interfering with real-time processing. Matlab tools are provided to read the stored data for timing accurate back tracing purposes.

To switch from the development to a deployable version of an algorithm is very easy since all components developed with the Matlab RTProc host can be employed without modification also in combination with the RTProc GUI host application which can be distributed to, e.g., customers as a stand-alone real-time demonstrator.
Porting a prototype developed with RTProcPC to RTProcDSP does not involve much programming effort since both versions of RTProc are mostly source code compatible.

4 Example Applications

RTProc was successfully used to develop various audio signal processing prototypes. The following are example projects which will be presented at the DS-RT 2008:

- **Acoustic Echo Control**: Acoustic echo in hands-free communication systems is due to the echo path from the loudspeaker to the microphone [3]. An echo canceler reconstructs and removes the echo signal from the microphone signal before transmission. The robustness of the adaptation following [4] in realistic acoustic environments is demonstrated by a real-time prototype developed with RTProc.

- **Speech Enhancement**: An approach to perform speech enhancement in noisy acoustic environments based on the filter-bank equalizer proposed in [8] was realized with RTProc. This new approach has an algorithmic delay lower than that achieved by common frequency-domain filtering approaches known from the literature. The key technique of the new approach is a non-uniform frequency resolution achieved by frequency warping based on an allpass transformation.

- **Voice Over Internet Protocol (VoIP)**: Based on RTProc, a simple VoIP application was realized to connect participants of a wideband voice call via internet. The involved digital speech transmission is based on the ITU-T G.722 wideband audiocodec [5].

5 Conclusion

In this contribution RTProc has been presented as a new system for the rapid development of audio signal processing prototypes on different platforms. Compared to the conventional way, RTProc simplifies the development process and shortens the development time. Key element of RTProc is a software architecture according to which each application is decomposed into different functional blocks. This architecture enables to liberate the developer of a new algorithm from all programming aspects which are not related to the actual algorithmic functionality.
Besides this, tools are provided with RTProc to support the developer to create runtime user controls, to verify the proper functionality even under real-time conditions, and to optimize the new algorithm. Currently, with RTProcPC and RTProcDSP two platforms are supported based on Microsoft Windows PCs and the ADSP-21369 EZKIT from Analog Devices respectively. Example real-time prototypes developed with RTProc will be presented in the demo at the 12th IEEE International Symposium on Distributed Simulation and Real Time Applications.

References

[1] Analog Devices, Inc. EZ-KIT Lite for Analog Devices ADSP-21369 SHARC Processor. http://www.analog.com, 2008.

[2] J. Blanchette and M. Summerfield. *C++ GUI Programming with Qt 4*. Prentice Hall, 2006.

[3] C. Breining and E. Hänsler. Acoustic echo control, an application of very-high-order adaptive filters. *IEEE Signal Processing Magazine*, Sept. 1999.

[4] G. Enzner and P. Vary. Robust and elegant, purely statistical adaptation of acoustic echo canceler and postfilter. In *Proceedings of International Workshop on Acoustic Echo and Noise Control (IWAENC)*, Kyoto, Japan, Sept. 2003.

[5] ITU-T, Rec. G.722. 7 kHz Audio Coding within 64 kbit/s, 1988.

[6] B. Kainka. *Messen, Steuern, Regeln über die RS 232-Schnittstelle*. Franzis Verlag, 1997.

[7] H. Krüger, T.Lotter, G. Enzner, and P.Vary. A PC based Platform for Multichannel Real-time Audio Processing. In *Proceedings of International Workshop on Acoustic Echo and Noise Control (IWAENC)*, Kyoto, Japan, Sept. 2003.

[8] H. Löllmann and P. Vary. Generalized Filter-Bank Equalizer for Noise Reduction with Reduced Signal Delay. In *Proceedings of European Conference on Speech Communication and Technology (INTERSPEECH)*, Lisbon, Portugal, Sept. 2005.

[9] Microsoft Corporation. DirectX Software Development Kit. http://msdn.microsoft.com.

[10] Microsoft Corporation. Windows Driver Development Kit. http://msdn.microsoft.com.

[11] Steinberg Soft- und Hardware GmbH. ASIO Interface Specification, v2.0, 1997-1999. http://www.steinberg.de, 1999.