

Rapid Development of Real-Time Systems Using RTExpress™

Milissa Benincasa Integrated Sensors, Inc.
 Richard Besler, Integrated Sensors, Inc.
 Diane Brassaw, Integrated Sensors, Inc.
 Ralph L. Kohler, Jr., Air Force Research Laboratory

ABSTRACT

This paper presents the RTExpress™ environment which is a software tool that assists a user in rapidly developing real-time embedded systems. RTExpress™ is a compiler and runtime environment that provides the capability for MATLAB® script files to be directly compiled and then executed on parallel high performance computers (HPC). RTExpress™ provides the capability to employ the power of an HPC on standard MATLAB® without having to recode the MATLAB® in the HPC target language. Its features include support for real-time data and machine performance visualization, multiple parallelization paradigms, multiple homogeneous parallel architectures, utilization of machine specific optimized vector libraries and native compilers, and the ability to change real-time algorithm parameters on-the-fly.

1. Introduction

The current process of developing and debugging application code for HPC's is expensive, time consuming, and limits the use of these systems to application areas where the development costs can be justified. The typical process cycle is depicted in figure 1 below.

Utilizing the approach presented in figure 1 requires the following steps to be performed in order to program a multiprocessing system. The first step is for the engineer or scientist to design the algorithm in MATLAB®. The correctness of the algorithm is verified by running simulations. Once the correctness of the MATLAB® algorithm has been verified, it is rewritten in the appropriate target language for the HPC. Typically the C programming language is utilized to rewrite the MATLAB® algorithm. The C implemented version of the algorithm must then be compared to the MATLAB® implementation to insure that it provides the same functionality. Once it has been verified that the functionality matches, the C implemented algorithm can be executed on the target HPC architecture. The results of the execution on the target HPC must then be compared with the results obtained from running the MATLAB® simulations to verify correctness. If inaccuracies are detected between the two versions, modifications must be made. This results in the reiteration of the whole process.

Utilizing RTExpress™ to program multiprocessor systems significantly reduces the development process. This reduced process cycle is depicted in figure 2 below.

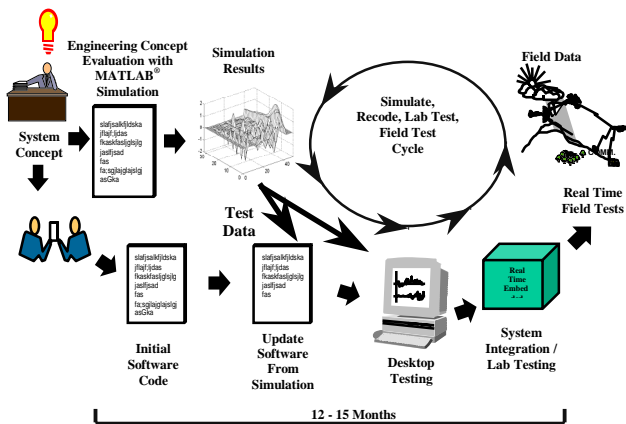


Figure 1- Current Development Process of Real-Time Systems

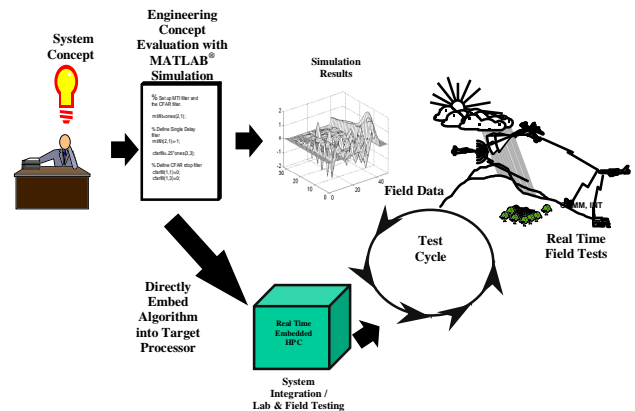


Figure 2 - Reduced Real-Time Development Process

The process is now shortened to include the following steps. The engineer or scientist still needs to design and develop the algorithm in MATLAB®, and then test/verify the accuracy of the algorithm in MATLAB®. But it is no longer necessary to rewrite the MATLAB® algorithm in the target language for the selected HPC architecture or verify that the rewritten algorithm corresponds to the MATLAB® algorithm. The scientist/engineer can then focus more on the functionality and requirements of the algorithm and not be concerned on how the algorithm will have to be translated to execute on the target HPC. The next step in this process is to utilize RTEXpress™ for specifying the algorithm parallelization, and selecting the target HPC architecture for execution. The RTEXpress™ user does not need to be knowledgeable in parallel programming, but rather can focus on the specifics of the functional decomposition of the algorithm. RTEXpress™ will generate the appropriate target code for the selected multiprocessing architecture. Once RTEXpress™ has generated the code for the selected HPC, it can be executed on the HPC platform. The results can be visualized graphically, and can then be compared with the results obtained by execution in the MATLAB® environment. If inaccuracies are found the algorithm can be modified in MATLAB®, not in the target language, and rerun through RTEXpress™.

2. The RTEXpress™ Environment

Figure 3 provides a functional block diagram representation of the RTEXpress™ environment.

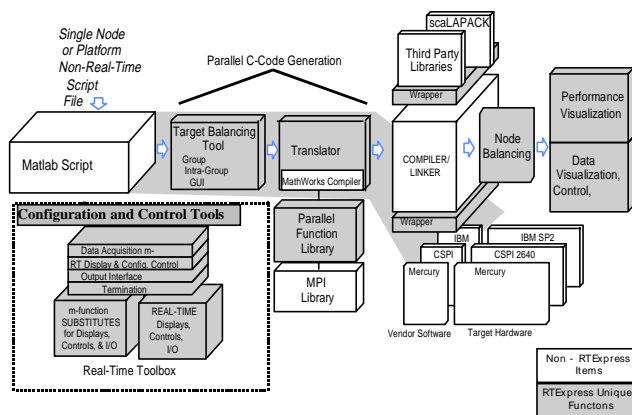


Figure 3 - RTEXpress™ Environment

This block diagram representation shows the flow of a MATLAB® m-file through RTEXpress™ and onto the supported target architectures. It depicts in gray the unique functions and features included in RTEXpress™.

The typical RTEXpress™ environment consists of a workstation referred to as the HOST PC which is utilized to interface with the target HPC. The HOST PC is used to

create, compile, and link the parallelized C-code generated by RTEXpress™. The parallelized C-code is then executed on the selected target HPC.

Referring to figure 3, the input to the RTEXpress™ environment is a MATLAB® m-file. This m-file is usually written to execute first on a single node or workstation platform. Before using RTEXpress™, the user must make sure their m-file runs correctly and error-free in MATLAB®. The Target Balancing Tool is then utilized to modify the source m-file, partition the m-file into groups and instances of groups, and specify the compile and runtime parameters for the selected HPC. The term group, for the purposes of RTEXpress™, is defined as referring to a set of tasks (comprised of a collection of MATLAB® scripts and functions) which are processed sequentially by each node of a user-specified set, or refers to the set of nodes dedicated to the tasks of the group. By design, all the nodes within a group perform the same tasks in unison on different portions of the data. The output from the Target Balancing Tool is the modified m-file or m-files.

The m-file is then sent through the RTEXpress™ Translator. The translation process consists of three steps: preprocessing, MATHWORKS Compilation, and post processing. The preprocessing step on the m-file is necessary in order to prevent the MATHWORKS Compiler from generating C code that does not parallelize well. For example, in many cases the MATHWORKS compiler converts functions to “for loops”. The preprocessing step detects these cases and modifies the statements so that the loops are not generated. Once the preprocessing of the m-file is complete, the m-file can be run through the MATHWORKS compiler. The output generated from the compiler is a C file. This C file is then post processed. The post processing step is required to provide support for either single or double precision data types. MATLAB® only supports double precision data types. The user specifies whether they want single or double precision data in the Target Balancing Tool.

Once the translation process is complete, the generated C code must be compiled and linked for the selected HPC architecture. It is at this point that the RTEXpress™ Parallel Function Library, MPI Library, Third Party Libraries, and the vendor specific libraries are linked in. The RTEXpress™ Parallel Function Library contains parallel implementations of many of the MATLAB® functions. For example, the Parallel Function Library provides parallel implementations of functions such as FFT, IFFT, and SORT, just to name a few. The MPI Library is linked in because RTEXpress™ uses the Message Passing Interface (MPI) for providing communication between nodes on the multiprocessing architectures. MPI was selected because it made RTEXpress™ very portable across the supported parallel architectures. Third party libraries, for example scaLAPACK, are utilized for the linear algebra parallel functions like eigenvalue, matrix

inverse, QR reduction, and matrix divide. The vendor specific libraries contain hand-coded machine specific vector functions that RTEExpress uses to increase performance.

The final step is to execute the C code on the selected HPC platform. RTEExpress™ provides the user with performance visualization and data visualization/control of their program. These visualization capabilities help to assist the user in determining how well their algorithm is performing on the selected HPC platform. Figure 4 presents an example of one of the visualization capabilities provided by RTEExpress™. It is the Parallel Performance Statistics Display.

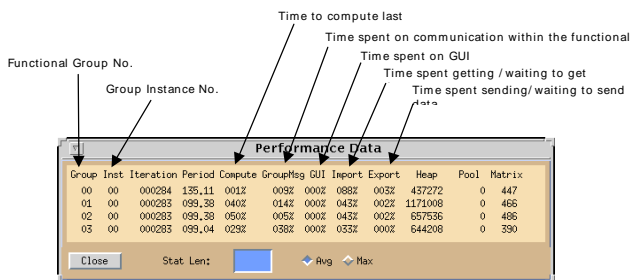


Figure 4 - Parallel Performance Statistics Display

The Parallel Performance Statistics Display is a dynamic display, which is continually updated while the program is executing on the HPC architecture. The Parallel Performance Statistics Display, in figure 4 contains four separate entries. Each entry corresponds to the performance statistics for a single group in an application. In this particular example, the application program was partitioned into four separate parallel groups. The statistics provided are obtained by utilizing two special purpose Real-Time Toolbox functions. These two functions are “itertop” and “iterbot”. By using these two functions, a user can specify which portions of their MATLAB® code are to be evaluated during program execution. The user specifies the beginning of the section of code to be evaluated by placing the “itertop” function in front of the first line of code. The “iterbot” function is placed at the end of the section of code that is to be evaluated. In order to make the performance calculations consistent between groups, the iteration processing of all nodes needs to be synchronized. This synchronization can be accomplished by placing the function “iterstart” just before the “itertop” function. The “iterstart” function upon being encountered will cause each node to halt processing and wait until all nodes are ready to begin iteration. The Parallelization Performance Statistics counters are reset when the “itertop” function is encountered, thus defining the beginning of iteration.

When “iterbot” is encountered, the CPU timers are read and the performance statistics are calculated.

The Parallel Performance Statistics Display provides the following information on a user program while it is executing: Iteration, Period, Compute, GroupMsg, Graphical User Interface (GUI), Import/Export, Heap, Pool, and Matrix. The Iteration column provides a running total of the number of times that an “itertop/iterbot” pair is processed for each group. The Period column is the average (or max) amount of CPU time spent between “itertop/iterbot” for the last number of iterations. The Compute column provides statistics on the fraction of time spent performing the computational processing tasks for the group. The GroupMsg column provides information on the percentage of time spent redistributing and passing data among the nodes for that particular group. The GUI column displays information on the percentage of communication time that is taking place with the master GUI node, which is receiving control information and run-time parameters, and sending performance and display information. The Import/Export column defines the percentage of time spent waiting to input or output data from one group to another group. The Heap column specifies the size of the heap that is currently being used. The Pool column contains the number of buffers being used from the static buffer pool. Finally, the Matrix column defines the number of matrices that are currently active.

The final piece in RTEExpress™ is the Real-Time Toolbox. The Real-Time Toolbox is a collection of functions designed to help the user better control the RTEExpress™ environment. To take advantage of the Real-Time Toolbox functions, the user must incorporate them into their MATLAB® m-file. The Real-Time Toolbox consists of both MATLAB® built-in functions and RTEExpress™ unique functions. Some of the MATLAB® based functions have been altered to better facilitate their use within RTEExpress™. To provide backward compatibility to non-real-time platform execution, all Real-Time Toolbox functions have been emulated using MATLAB® script files and the syntax has been kept identical with any MATLAB® based counterparts.

The Real-Time Toolbox functions consist of the following: data input/output functions, group communication functions, real-time displays/controls, standard MATLAB® graphic functions, and special RTEExpress™ displays. A user can obtain definitions of the calling parameters required by the Real-Time Toolbox functions, by executing MATLAB® and requesting help on a function.

RTEExpress™ supports MATLAB® based file input/output functions. Data generated at any point during RTEExpress™ execution can be stored to a file or loaded from any file type that MATLAB® supports. RTEExpress™ also provides the capability to interface data from other

devices. For example, the Real-Time Toolbox provides input/output streams for file input/output and for the Mercury Raceway RINT-T input/output stream.

The Real-Time Toolbox contains the following group communication functions: `groupexport`, `groupimport`, `groupid`, `instanceid`, and `numinstances`. The most important of all these functions are the `groupexport` and `groupimport` functions. These two functions allow data to be passed between RTExpress™ defined groups.

RTExpress™ provides an elaborate real-time GUI. Continuous data and algorithm performance metric visualization, real-time control of application parameters, and display of parallel architecture performance statistics are all available with only minor modifications to the original MATLAB® m-file. A user can select GUI support as a configuration parameter in RTExpress™. By selecting GUI support, upon execution of the program in RTExpress™, a server window is displayed containing three buttons. These buttons are Performance, Figures, and Quit. Selecting the Performance button will display the Parallelization Performance Statistics window, which has already been discussed previously. By selecting the Figures button, another window is displayed, it will contain buttons for each of the figures generated by the user MATLAB® code. The user must define in their MATLAB® code what figures are to be displayed from their application. The Quit button terminates RTExpress™ execution.

Finally, the Real-Time Toolbox provides support for standard MATLAB® graphics and special RTExpress™ displays. The standard MATLAB® graphical functions that are supported are the following: `figure`, `plot`, `image`, `imagesc`, `clear`, `set`, `get`, `uicontrol`. The special displays that RTExpress™ provides are: an enhanced MATLAB® plot capability which provides support for real-time operation, a real-time radar PPI display, and a modified version of `uicontrol` which provides the capability to display and modify `uicontrol` objects.

3. The Target Balancing Tool

The Target Balancing Tool is the main component of the RTExpress™ environment that the user interfaces with in order to execute their MATLAB® program on a HPC. Refer to Figure 5 for an example of the Target Balancing Tool.

It is in this tool that the user modifies the MATLAB® m-file, partitions the m-file into groups or instances of groups, and specifies what parallel architecture the m-file will execute on. The Target Balancing Tool provides the user with the following capabilities: file manipulation functions, editing functions, a search function, font and color selections, group specification functions, and build run-time functions.

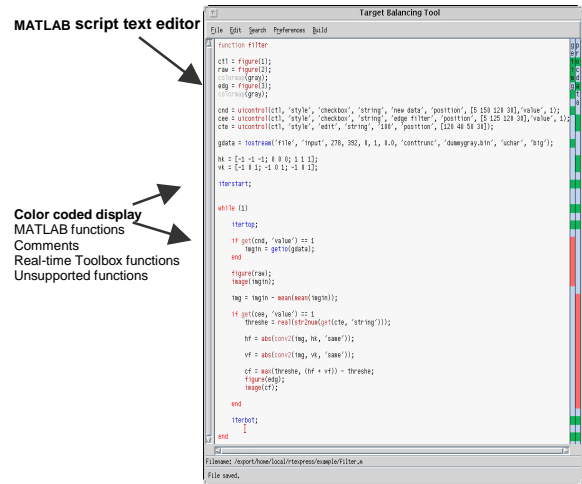


Figure 5 - Target Balancing Tool

The two most important functions provided by the Target Balancing Tool are the group specification functions and the build run-time functions. The group specification functions provide the user with the capability to define the parallel partitioning of their MATLAB® m-file. The partitioning is defined by dividing the m-file into functional groups. This allows the user to control the top-level parallelization. While tasks performed by different groups are processed concurrently and asynchronously, tasks within a group are performed sequentially as they occur within the MATLAB® m-file. Each group specified has associated with it a main m-file, which is referred to as a group file. The group file can contain any RTExpress™ supported MATLAB® command. It can also contain calls to other MATLAB® scripts or functions that are required by the specified group of tasks. The group file can also contain Real-Time Toolbox functions which can control the data I/O, real-time displays, iteration structure, initiation, and termination. The user can specify the group and group name using the Target Balancing Tool. This capability provides the user with a great deal of flexibility in RTExpress™. In defining the group partitioning of an m-file, the user has access to a number of different parallelization paradigms that are supported by RTExpress™. This provides the user with the capability to define a variety of architectures limited only by the number of nodes available on the targeted architecture, and the number of ways that the MATLAB® m-file can be partitioned.

The following parallelization paradigms can be specified in the Target Balancing tool: task parallel, pipelined, data parallel, round robin, and mixed mode parallelization. Task parallel is defined as the ability to execute multiple tasks concurrently on the same data by separating the tasks into different functional groups.

Pipeline processing is defined as the concurrent asynchronous operation of groups connected in a series. Data parallel processing is defined as the allocation of separate portions of a data segment among a set of nodes each which performs the same tasks. Round robin parallelization consists of sending distinct data segments to different groups that are operating in parallel. Finally, in mixed mode parallelization the user can combine any of the supported parallelization paradigms together in a single application.

The user specifies a group by highlighting the lines of code in the m-file that will constitute the group. The add group dialog is then selected. It is also in this group dialog, where the user will specify how many physical nodes will be assigned to a group. Once the user has partitioned the main portions of the m-file into functional groups, the Target Balancing Tool Auto Assign Determine selection is utilized. This function saves the user from having to assign every single line of the m-file to a group. The Auto Assign Determine selection will parse the MATLAB® m-file and determine how the remaining lines of the m-file should be assigned. In some cases, lines of code may get assigned to multiple groups. This is usually the case with variables. Once the user has functionally partitioned the MATLAB® m-file, they are ready to utilize the Build Run-Time functions capability of the Target Balancing Tool.

The Build Run-Time function capability, is where the HPC architecture is specified. The compile, link, and run-time parameters are also specified here. Figure 6 depicts the Target Balancing Tool Configuration Dialog.

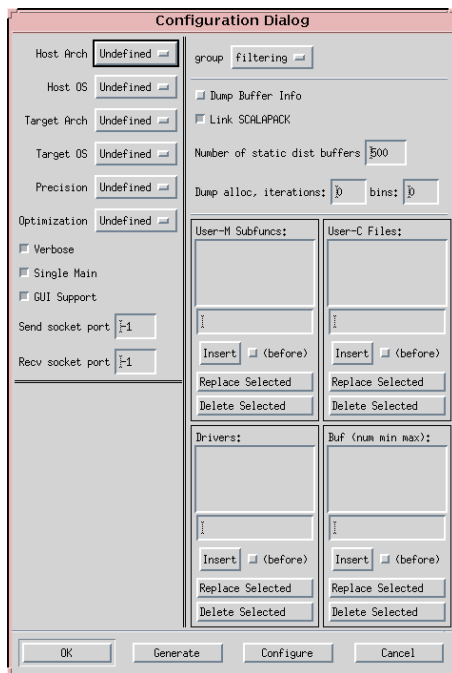


Figure 6 - Configuration Dialog

The Configuration Dialog contains the following parameters: host architecture, host operating system, target architecture, target operating system, precision, optimization, verbose, single main, GUI support, send socket port, receive socket port, dump buffer info, link scaLAPACK, number of static dist buffers, dump alloc iterations, user-m files, user-C files, drivers, and buf. Besides specifying what the target HPC architecture will be for execution, and its run-time parameters, the configuration dialog also provides the user with the ability to specify additional m-files, link in user C files, and hardware drivers. If there are any user specified function m-files called by the user's main m-file program, the m-files must be listed so that they can be translated and linked in. Also if the user wishes to link in C MEX files that are called from the main m-file program, they must be listed to, so they can be linked in as well.

Once the configuration dialog has been filled in, the Generate command must be selected at the bottom of the configuration dialog. The Generate command will execute the RTEExpress™ genmake tool which will create a make file for compiling and linking, and a “go” launch script for runtime execution. The Genmake command will also create a Main.c file that is linked to the translated C code. Upon exiting the Target Balancing Tool, the user will then need to execute the “make” command which will create the executable for the selected HPC target architecture. Once the executable has been created, the “go” command is used to execute the application on the architecture.

4. The Benefits of RTEExpress™

One of the powerful features of RTEExpress™ is that it allows an algorithm to be specified in MATLAB®. MATLAB® has become a widely accepted choice by scientists and engineers for expressing ideas in their science and engineering development projects. Also for this community of users, the MATLAB® language interface is far easier to work with than having to use C or Fortran to allow an algorithm to execute on an HPC. The RTEExpress™ environment provides the user with the capability to express their algorithm in a language that is appropriate, thus making the algorithm the central focus, rather than having to worry about how to express the algorithm, therefore increasing the quality of the developed algorithm.

Another important feature of the RTEExpress™ environment is that it isolates the user from having to deal with implementation details. Programming a parallel architecture is not an easy task, and often requires an experienced parallel programmer. In particular, one of the difficult tasks in programming a problem on an HPC, is partitioning the problem over multiple nodes, and handling the details of inter-processor communication efficiently. The scientist or engineer who is developing the algorithm

does not want to be concerned with implementation details. RTEExpress™ frees the user from having to focus on these types of implementation details. This significantly reduces the time to develop the algorithm, and therefore reduces the development costs. Also because the algorithm does not contain the implementation details, it is easier to adapt and maintain the algorithm as it changes. This results, due to the fact that the level of expression has now been raised to a point where it states what is to be done, not how to do it.

5. Future Directions

Under DARPA funding, research is currently being conducted to extend the capabilities of RTEExpress™. In particular the goals of the new research are to: provide heterogeneous architecture mapping and physical constraint specification, development methods for seamless translation of abstract real-time constraints, and provide multidimensional parallelism combining user specified and automated graphical methods. Under this research program, heterogeneity is defined as providing the capability for an application to be partitioned to execute across different node types within a physical hardware box. Currently, RTEExpress™ only supports homogeneous nodes within a physical hardware box.

Another important aspect of this program is to develop methods for seamless translation of user specified real-time constraints to MPI Real-Time (MPI/RT) quality of service, and MPI/RT real-time paradigms. RTEExpress™ does not currently support real-time paradigms such as constraint driven, priority driven, event driven, and time based. RTEExpress™ will be extended to incorporate at least two of these paradigms. RTEExpress™ will also be modified to support MPI/RT.

The results of this research will be to provide an integrated environment which allows the engineer or scientist to directly target a heterogeneous embedded HPC based on systems algorithms specified in MATLAB®, system timing requirements, heterogeneous architecture specifications, and manual/automatic parallelization methods. This environment will be referred to as H-RTEExpress™.

In addition to the DARPA funded research, ISI is further refining the RTEExpress™ product. ISI is adding support for additional hardware platforms, and providing a more robust Target Balancing Tool. RTEExpress™ V1.0 will be released commercially in February 1998.

6. Acknowledgements

The initial concept for this research was funded under an Air Force Research Laboratory contract. We would

like to thank Dr. Russell Brown for his efforts under this program.

The development of the RTEExpress™ environment was funded under DARPA/ITO BAA 95-19. We wish to thank Dr. Jose' Munoz at DARPA/ITO for his continued support and direction.

References

- [1] William Gropp, Ewing Lusk, and Anthony Skjellum, *USING MPI: Portable Parallel Programming with the Message Passing Interface*, The MIT Press, Cambridge, MA, 1994.
- [2] *MATLAB® Reference Guide*, The MathWorks, Inc., Natick, MA, 1992.