Utilization of TargetLink for Delphi Control Applications

As a part of the company's effort to increase the quality of the products and minimize its development cycle, Delphi is rapidly adopting algorithm modeling and code generation techniques. The production code generator TargetLink from dSPACE is a major component of the tool chain that Delphi has been using in a new development environment.

Dr. Lev Vitkin, Delphi Delco Electronics Systems, U.S.A.

We had the chance to interview Dr. Lev Vitkin, who is the Autocode Technology Leader of Delphi Delco Electronics Systems.

dSPACE NEWS: Lev, thank you for giving us the opportunity to ask you more about the usage of TargetLink for automotive control application development within Delphi's projects. Why is Delphi using TargetLink as a production code generator?

Lev Vitkin: Using TargetLink is demonstrably advantageous for Delphi's software developers. There are really no syntax errors, and logical errors are more quickly identified and traced to their root cause. We have found out that the speed of coding a model and testing of the generated software is greatly increased; therefore the engineers have more time to find the "optimum solution" rather than the quickest or easiest one. Model changes midstream – of which there are usually many – do not force the developer to go back to step one. Another advantage is that TargetLink's code consistency is unmatched, allowing better handoffs between several engineers working on the same project.

dSPACE NEWS: What were the criteria TargetLink had to meet in order to fulfill its role as a production code generator ?

Lev Vitkin: One of the main criteria for an autocode tool for automotive applications is the efficiency of generated code. TargetLink did not cause any significant overhead on the project budget with regard to ROM, RAM, and throughput. Generated code featured the same quality as hand-written code for similar applications. In terms of consistency, I can state that the same type of model-to-C-code implementation is repeated, which makes work a lot easier. Also, the model can be set to generate code that is itself modular, we call that a self-contained function call, and it allows global or local access to the model variables. So TargetLink supports and delivers modularity. TargetLink is very flexible. You can easily "tune" automatically generated code to comply with Delphi software coding standards. And finally, the quality of the technical support of a new complex tool, like a code generator, is another main criterion in choosing a tool's supplier. We are very pleased with the level of technical expertise and responsiveness of the TargetLink support team.

dSPACE NEWS: We have heard that you are optimizing the usage of TargetLink with the help of a special Delphi User Guide. Could you tell us more about that?

Lev Vitkin: Yes, we have developed a set of procedures and guidelines for algorithm modeling in a MATLAB®/Simulink®/Stateflow® environment and within TargetLink code generation so that changes to the model are more readily noticed and logical or implementation errors are less likely to occur.

For our engineers we recommend: The model and its components should follow a specified Delphi structure (Fig. 2). This ensures that minimum modifications are involved, if any, in a stage of generation of highly structured and efficient code. Moreover, we advise our engineers to use Delphi's Simulink and Stateflow library blocks (Fig. 1). These blocks were created firstly, to implement in Simulink/Stateflow some reusable routines that are common in Delphi software applications, and secondly, to ensure the efficient TargetLink code generation representing these blocks. To minimize the project development cycle and promote the consistency in algorithm development, Delphi's TargetLink User Guide specifies how to apply TargetLink for Delphi applications. This guide covers topics like variable initialization, accommodation of legacy code in autocode, integration of autocode into handwritten applications, code verification procedures, and tips on setting TargetLink properties.

dSPACE NEWS: Which application did you apply TargetLink to?

Lev Vitkin: We used TargetLink for several Motorola 16-bit and 32-bit processor based applications, like EGR Control, Rollover Detection, and Adaptive Cruise Control. These algorithms were modeled with a mixture of Simulink blocks and Stateflow charts.

dSPACE NEWS: What did you personally experience as the greatest challenge of Delphi's application projects with TargetLink so far ?

Lev Vitkin: That was to incorporate legacy code and structures into autocode as well as assign specific memory allocation for variables and constants. But our engineers overcame it without any handwritten modifications of generated code by using the full power of TargetLink's flexibility in code generation.
Finally, the usage of a code generator requires strong discipline to follow the process of building the software application. So, we have to be sure that our Common Systems and Software Development Process properly accommodates new autocode technology, and our engineers thoroughly follow that process.

dSPACE NEWS: Thanks a lot for your time, Lev!

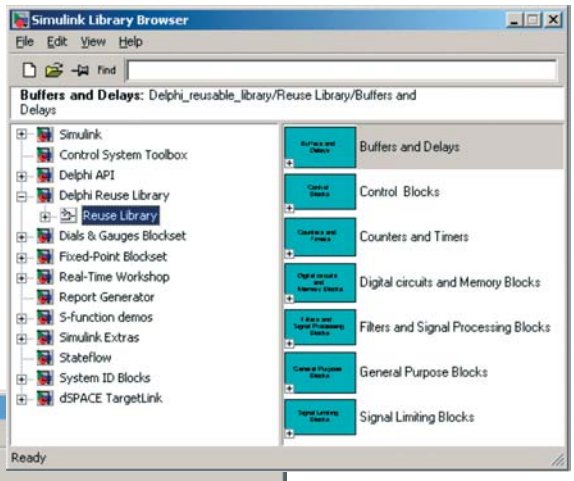

Fig. 1: Delphi's reuse library – created, among other things, for efficient use of TargetLink code generation.

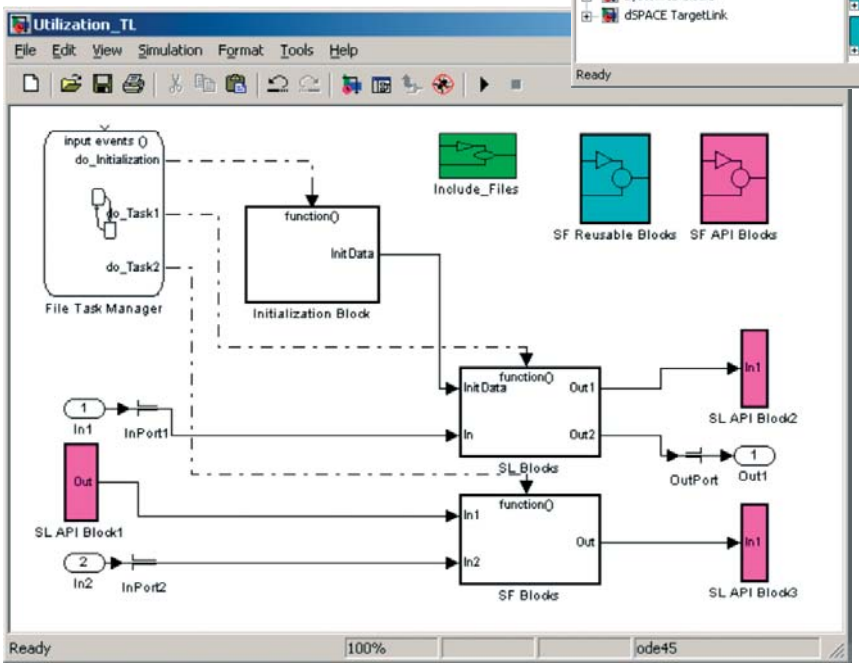

Fig. 2: A specified Delphi structure for models and model components.