

Best Practices für TargetLink-Modelle

Um Regelungsalgorithmen optimal in effizienten C-Code umzusetzen, wurden in Zusammenarbeit mit einem deutschen OEM Modellierungsrichtlinien für TargetLink entwickelt. Die frisch aufgelegte Version 1.0 steht allen TargetLink-Anwendern zur Verfügung. Überarbeitungen des Modells, wie sie an der Schnittstelle zwischen Funktions- und Software-Entwicklung in der Vergangenheit oftmals erforderlich waren, werden deutlich reduziert und die Produktivität im Entwicklungsprozess gesteigert.

- Mehr Produktivität im Entwicklungsprozess
- Nahtloser Übergang von Funktions- zu Software-Entwicklung
- Tipps für effizienten Code, MISRA-Konformität, transparente Modelle usw.

Wozu werden Modellierungsrichtlinien benötigt?

Bei der Umsetzung von Reglermodellen tritt oftmals das Problem auf, dass in der Funktionsentwicklung Modellierungsstile bzw. Modellierungselemente verwendet werden, die sich nicht in effizienten C-Code umsetzen lassen. Die Folge sind zeitaufwendige und fehleranfällige Überarbeitungen der Modelle, die im Zuge der Software-Entwicklung mit TargetLink vorgenommen werden müssen. Die TargetLink-Modellierungsrichtlinien unterstützen nun Funktionsentwickler bei der Auswahl einer geeigneten Sprachuntermenge in Simulink®/Stateflow® und Software-Entwickler bei der optimalen Umsetzung der Modelle in hocheffizienten C-Code.

Inhalt der Modellierungsrichtlinien

Die TargetLink-Modellierungsrichtlinien umfassen rund 150 Regeln zu folgenden Aspekten:

- Transparentes Reglerlayout
Analog zu Codierrichtlinien werden auch auf Modellebene Regeln postuliert, die Transparenz und Lesbarkeit von Modellen erhöhen.
- Geeignete Sprachuntermenge
Es wird eine Untergruppe der in MATLAB®/Simulink®/Stateflow® verfügbaren Sprachelemente definiert, die eine optimale Umsetzung der entwickelten Modelle mit Hilfe von TargetLink ermöglicht.
- Optimaler Festkomma-Code
Neben den in TargetLink bereits integrierten Hilfsmitteln wie Autoskalierung dienen die Regeln als Leitfaden zur Umsetzung von Modellen in hocheffizienten Festkomma-Code.

- Code-Generator-Optionen
Es werden Optimierungseinstellungen für das Handling von Variablen und Funktionen beschrieben, um effizienten Code zu erzeugen.
- MISRA-Konformität
Es werden Regeln formuliert, die eine maximale Konformität des generierten Codes mit MISRA C gewährleisten.

Variable Handling

11.6 Moving of Variables

The MOVABLE optimization attribute for variable classes signals to the code generator that the code for variables of that particular class can be moved into dependent branches whenever possible.

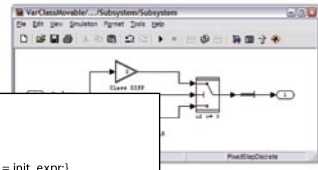
Purpose
Generation of efficient code.

Remark
When code is moved into dependent branches, intermediate results are calculated only if they are really required. This reduces execution time.

References

- TargetLink Advanced Practices Guide [6], Optimizing the Production Code > Optimizing an Entire TargetLink System > Optimizing Logging > The Variable Class Attribute 'Movable'

Example
Fig. 75 shows a subsystem which contains branches that do not need to be calculated. The MOVABLE optimization attribute is set for the DSP variable class, see fig. 76.



Loops/Chart

```

(loop_var = init_expr;
while ([loop_condition])
{
loop_action;
loop_var = modify_expr;
}
                
```

◀ Jede Regel enthält Namen, Definitionen, Zweck, Hintergrundinformationen sowie optionale Verweise und Beispiele.

▲ Die Verwendung von Design Patterns in Stateflow (hier zur Modellierung einer Schleife) ermöglicht die direkte Übersetzung in effiziente C-Konstrukte.

TargetLink-Anwender können die Modellierungsrichtlinien (PDF-Dokument) kostenlos direkt bei unserem Vertrieb unter info@dSPACE.com anfordern.