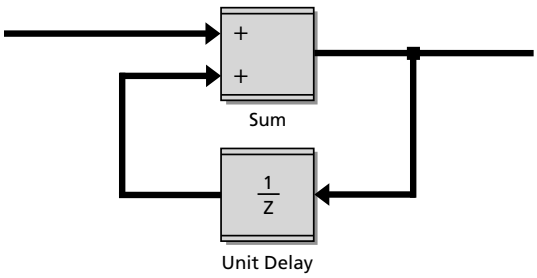
Laufzeitfehlern auf der Spur

- Automatische Laufzeitfehleranalyse von TargetLink-Code
- Navigation vom analysierten Code direkt ins Modell
- Hohe Genauigkeit der Analyse durch Tool-Integration

Bei der Steuergeräte-Software-Entwicklung unterstützt der Seriencode-Generator TargetLink den Anwender nicht nur durch die Generierung des eigentlichen Funktionscodes. Vielmehr steht dem TargetLink-Anwender auch eine umfangreiche Tool-Landschaft für Verifikations- und Validierungsmaßnahmen offen, die teils in TargetLink enthalten ist, teils durch Erweiterungslösungen wie MTest zur Verfügung gestellt wird. Zu dieser Tool-Landschaft gesellt sich nun in Form der TargetLink-PolySpace-Integration ein weiteres innovatives Werkzeug, das der Absicherung gegen Laufzeitfehler dient.

Ursachen von Laufzeitfehlern

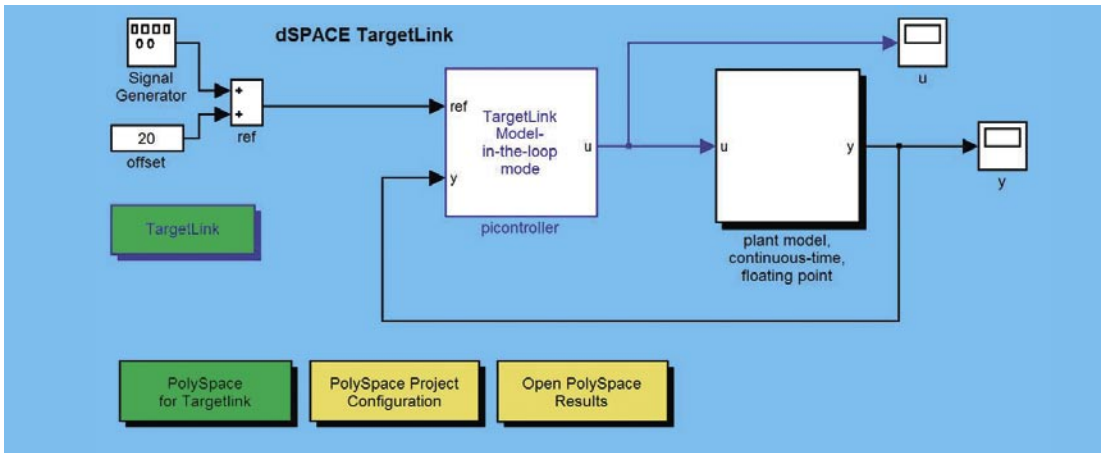
Auch wenn sich kein menschlicher Programmierer in puncto Fehlerfreiheit mit automatischen Seriencode-Generatoren messen lassen kann, ist der generierte Code nicht zwingend frei von Laufzeitfehlern. Die Ursache dafür ist, dass Fehler bereits im Modelldesign, das heißt im Rahmen der Funktionsentwicklung, eingebracht werden können. Bestehen auf Modellebene beispielsweise keine geeigneten Maßnahmen gegen Divisionen durch Null oder potenzielle Über- oder Unterläufe des Wertebereichs, so kann auch der generierte Code Laufzeitfehler enthalten, da die potenziell fehlerhafte Spezifikation 1:1 in Code umgesetzt wird. Die Quellen etwaiger Laufzeitfehler sind somit auf der Modellebene zu suchen und sollten auch bevorzugt dort behoben werden, was durch die TargetLink-PolySpace-Integration maßgeblich vereinfacht wird.



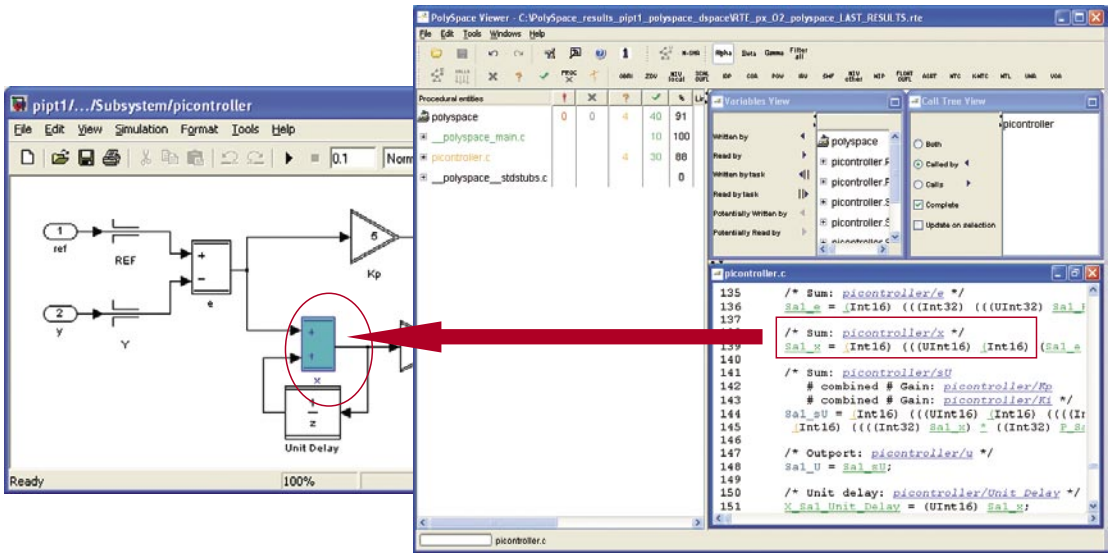
▲ *Modellfragment mit potenziellem Über-/Unterlauf aufgrund der enthaltenen Rückkopplung.*

Was leistet die TargetLink-PolySpace-Integration?

Die Integration ermöglicht die direkte Anbindung von TargetLink an den PolySpace Verifier, der auf Basis einer statischen Analyse mit Hilfe der so genannten abstrakten Interpretation den generierten Code analysiert. Diese Technik gestattet es, Aussagen über Laufzeitfehler mit einer Exaktheit vorzunehmen, die mit der mathematischer Beweise vergleichbar ist. Einzelne Code-Fragmente werden entweder als hundertprozentig laufzeitfehlerfrei, hundertprozentig laufzeitfehlerbehaftet, als „toter Code" oder als potenziell unsicher eingestuft. Lediglich die letztere Klasse bedarf dann noch der genaueren Analyse durch den Anwender, da aufgrund von Näherungen (Abstraktionen) nicht endgültig determiniert



▶ *Starten und Konfigurieren der Code-Analyse auf Modellebene mit Hilfe zusätzlicher TargetLink-PolySpace-Blöcke.*



▲ *Farbliche Hervorhebung von potenziellen Laufzeitfehlern im Code durch den PolySpace Viewer und direkte Navigation ins Modell.*

werden kann, ob tatsächlich ein Laufzeitfehler auftreten kann oder nicht.

Vorteile der TargetLink-PolySpace-Integration

Durch die TargetLink-PolySpace-Integration ergeben sich für den Anwender beider Werkzeuge wesentliche Vorteile:

- Die Analyse des generierten Codes wird in gewohnter Manier direkt von der Modellebene mit wenigen Mausklicks gestartet. Auch die Konfigurationsparameter des PolySpace Verifiers und das TargetLink-Subsystem, dem der zu inspizierende Code zugeordnet ist, werden direkt im Modell spezifiziert.
- Im PolySpace Viewer, der den generierten Code darstellt und die einzelnen Operationen entsprechend ihrer jeweiligen Klassifikation farblich hervorhebt (z. B. grün für hundertprozentige Laufzeitfehlerfreiheit, orange für potenziell unsichere Stellen mit nicht endgültig geklärtem Laufzeitfehlerverhalten), kann vom Code direkt an die korrespondierenden Stellen im Modell navigiert werden. Hierdurch lassen sich die kritischen Stellen besonders komfortabel ins Modell zurückverfolgen, inspizieren und gegebenenfalls korrigieren.
- Durch die Spezifikation von Zusatzinformationen auf Modellebene wie Wertebereichsgrenzen für applizierbare Parameter oder Eingangsgrößen kann die Genauigkeit der Analyse substanziell gesteigert werden. Der PolySpace Verifier liest diese Informationen aus dem dSPACE Data Dictionary aus und berücksichtigt sie in der Analyse, wodurch die Zahl der Code-Zeilen mit nicht endgültig geklärtem Laufzeitfehlerverhalten reduziert wird.
- TargetLinks optimierte Code-Generierung durch die Technik „Compute through Overflow“ wird vom PolySpace Verifier explizit als solche erkannt, wodurch wiederum die Analyse des generierten Codes vereinfacht wird.

Die von PolySpace und dSPACE vorgenommene Tool-Integration führt nicht nur zu einer Beschleunigung des Entwicklungsprozesses, sondern sie ebnet auch einen komfortablen Weg für die Absicherung des generierten Seriencodes.

Glossar

Toter Code – Programmfragment, das grundsätzlich nicht erreicht bzw. ausgeführt werden kann.

Compute through Overflow – Berechnungsverfahren für arithmetische Operationen, bei dem, wenn das Endergebnis korrekt darstellbar ist, Überläufe in Zwischenresultaten auftreten dürfen.

Abstrakte Interpretation – Verfahren zur Analyse der Semantik eines Programms, wobei Abstraktionen zur Reduzierung des Berechnungsaufwandes eingesetzt werden.

Weitere Informationen zur TargetLink-PolySpace-Integration (PolySpace for Model Based Design) sind bei der Firma PolySpace unter contact@polyspace.com erhältlich.