TeraSoft 鈦思科技
www.terasoft.com.tw

基於目標的快速原型化: 引導生產ECU軟體開發的實用方法

Tom Erkkinen
MathWorks, Inc.
Seshadri Shekar
Automotive Infotronics Private Limited
Mohamed Ziaudeen
Automotive Infotronics Private Limited
Shobhit Shanker
MathWorks, Inc.

摘要

快速控制原型化(Rapid control prototyping, RCP)是一項被廣泛運用在驗證控制器的功能性行為的技術。通常RCP會使用一個具備充足處理能力及記憶體的目標處理器，使得這項技術讓工程師探索新的觀點時更具吸引力。然而，在使用目標處理器進行快速控制原型化以及生產ECU (Electric control unit)間，包含可用的程式碼產生技術、可支援的工具鏈，及I/O (Input/Output) 硬體之間往往存在著極大的差距。因此，把控制器從快速控制原型化移植到生產上需要龐大的工作。除此之外，由於成本上的限制，RCP系統很難被大量的套用到車輛運行測試或產前試驗。

針對RCP所面臨的挑戰，汽車工程師開始使用一項稱為基於目標的快速原型化(on-target rapid prototyping, OTRP)的技術。OTRP可以用來產生程式碼、對程式碼進行交叉編譯，以及載入程式碼到生產用的ECU或具備外加記憶體及儀器支援的開發版本ECU上。OTRP讓工程師可以在開發過程中使用同一套程式碼產生器、支援工具鏈及ECU硬體，因而簡化了移轉到生產的流程。另外，因為開發用ECU的相對較低廉成本，基於目標的快速原型化系統可以被大量的套用。

這篇論文將介紹模型化基礎設計(Model-Based Design)與基於目標的快速原型化(OTRP)、利用一個新的演算法輸出技術來逐步採用OTRP，以及轉移OTRP到生產的考量。我們將用一個應用範例說明如何有效的使用OTRP，用於以生產為目的的ECU程序，包括把一個新的外部模式實現到一個資源有限的定點數(fixed-point)嵌入式系統。

模型化基礎設計(MODEL-BASED DESIGN)介紹

一個模型即代表一個回應了其輸入、狀態、時間函數的動態系統。過去系統工程師使用如圖1所示的區塊圖來建立受控體環境與實體系統模型，以及ECU演算法之設計。

近幾年來，由模塊圖及狀態機(state machines)組成的圖形化建模環境被用來分析、模擬、原型化、定義，以及套用軟體演算法到生產用的ECUs中。模型化基礎設計指的是利用模型及建模環境作為ECU開發的基礎。

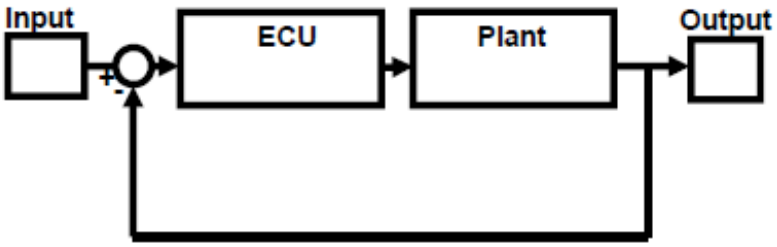

圖1: 回授控制器模型

使用模型化基礎設計開發的汽車系統包括:

- 引擎及變速的ECUs
- 混合動力、電池，及環保車輛系統
- ABS及車身底盤控制系統
- 氣候控制與車用電子
- 儀表板與顯示
- 無線接收器與音頻訊號處理

模型化基礎設計的使用貫穿整個系統開發的生命週期，有助於持續地驗證與規格需求有效性檢測、設計及實現。這個方法可為正式的軟體流程以及任何注重風險管理、失誤預防，或早期錯誤偵測的設計專案帶來顯著的優勢。

模型化基礎設計由下列主要活動組成:

- 建模與模擬
- 快速原型化
- 嵌入式轉檔與佈署
- 迴圈測試(In-the-loop testing)
- 整合活動

建模與模擬(MODELING AND SIMULATION) – 一個動態系統的模塊圖代表了整個系統機制，而模塊集合藉由訊號線條表示了其間的交互關係，這些訊號即為模塊的輸入、輸出，及狀態。

模塊及線條可能是真實的或虛擬的。虛擬的模塊或線條不會影響模擬結果，卻有助於建構或理解模塊圖。不論模塊與子系統是真實或虛擬，都可以被儲存在自訂的函式庫中有助於之後重複使用及摘錄。大型模型可以使用模型一為參考來建立，尤其是最高層級的模型調用較低層級的模型的方式可以降低記憶體的消耗又能加快模擬及模型更新的時間。

模擬可以經由兩種方式完成。其一為使用模型內建的代表型式，用解譯模式來執行這個模擬。另一種方式為先從模型中產生程式碼，再將程式碼透過產生程式碼中的模擬技術來執行。解譯模擬可讓使用者對於執行環境能有更多的掌握並提供更佳的互動功能，但對於大型模型而言在速度上會比較慢。利用程式碼產生方式進行的模擬則互動性較少，但可以提升速度。基於這個理由，從程式碼產生的模擬又被稱為模擬加速。使用參考模型來建模的模型可以用正常模式(解譯式)或加速模式來模擬。

快速原型化(RAPID PROTOTYPING) – 所謂簡略快速原型化(bypass rapid prototyping)，程式碼是由控制器或演算法模型產生。這個程式碼接著被交叉編譯及下載到一個高速(通常是定點)快速原型化電腦中即時執行。I/O 通常由記憶夾或連接到快速原型化電腦及現有 ECU 的模擬裝置來管理，或許它們仍被裝置在一台車輛中。其他的 I/O 選項包括匯流排(bus)之間的溝通，例如 CAN，可能需要客製的訊號處理及電力電子。該控制器參數在試駕或在實機(如引擎)實驗時被微調，且繞過現有 ECU 程式碼允許嵌入新的程式碼。當一組參數值被定義可以幫助達成規格要求，新的演算法就會被認定是可實行的(見圖 2)。

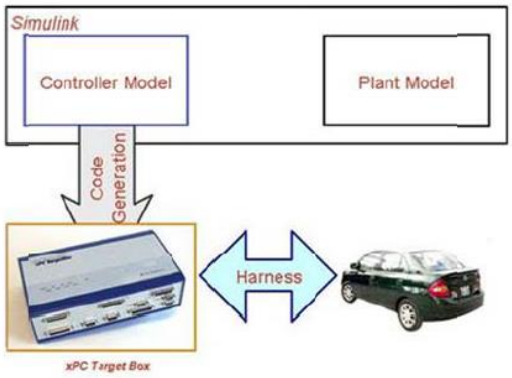

圖 2: 簡略快速原型化(Bypass rapid prototyping)

在 OTRP 中，藉由簡略快速原型化，僅產生了模型控制器部分的程式碼。然而，在 OTRP 交叉編譯器的程式碼不會被轉檔佈署到快速原型化的電腦，而是到嵌入式微處理器或生產用 ECU，或者可能是一個裝配有多一點存儲空間及 I/O 的程式碼的近似處理器上。OTRP 通常使用一個整

數處理器，因此相較於浮點運算處理器及供簡略的快速原型化模型，需要一個更詳細的定點模型。I/O 則是藉由標準 ECU 裝置來管理。

接著，主機選擇性地直接跟 ECU 硬體連接(也許還存在車體內)，透過外部模式或一個校準工具進行參數調校；控制器參數接著被轉成"運作中"。當性能達到要規格要求時，新的演算法會被顯示為可行和實用；也就是說，新演算法會被使用在生產及資源受限的環境中(見圖 3)。

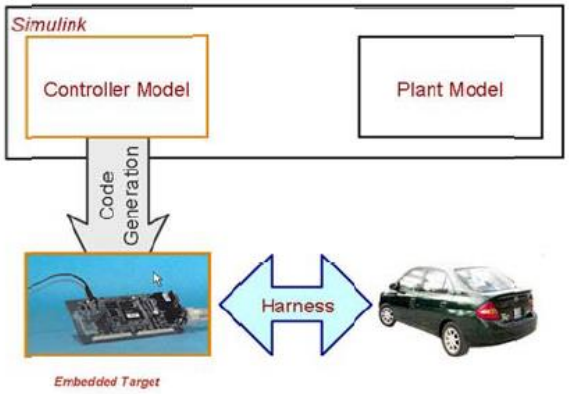

圖 3: 基於目標的快速原型化(On-target rapid prototyping)

表 1 為傳統簡略快速原型化與 OTRP 比較

	傳統快速原型化	**基於目標的快速原型化**
目的	利於測試新構想及 green-field 環保領域的研究	利於開發階段設計的精進及校準
執行硬體	使用個人電腦或非目標硬體	使用 ECU 或接近量產的硬體
編碼效率, I/O 等待時間	較不注重編碼效率及 I/O 等待時間	較注重編碼效率及 I/O 等待時間
適用之專案	適用於新車專案	適用於針對現有專案的改善
工程師	通常在研發或生產階段由系統工程師完成	通常在生產階段由系統及軟體工程師完成
成本及便利性	可能需要客製即時模擬及硬體，或者使用較便宜的現成 PC 硬體及 I/O 板	可以使用現有硬體，所以比較方便、便宜，尤其是大量套用的時候

表 1: 傳統與基於目標的快速原型化比較

嵌入式轉檔及佈署(EMBEDDED DEPLOYMENT) – 快速原型化之後，控制器模型常常會被轉換成詳細的可執行軟體規格。在這裡需要考量一些因素，像是功能及檔案劃分、開機與關機程序、診斷，及內建的例行測試程序。這個模型已被收斂及調整到能夠妥善地在嵌入式系統硬體上執行。

接下來，要產生更精細控制器模型的嵌入式程式碼，並將程式碼作為生產軟體建構的一部份下載到實際的嵌入式微處理器或 ECU 上。在這個步驟沒有相關的模擬行為，重要的是要確保最終建

構已完全地整合，而且能自動經由現有的程式碼、I/O 驅動程序、即時作業系統(real-time operating system, RTOS)軟體產生程式碼。

迴圈測試(IN-THE-LOOP TESTING) – 模型模擬是第一次驗證與有效性檢測其中的一個步驟。測試模型需要用一個比特定模擬更嚴格的方法，且通常用在早期演算法開發階段來執行。模型的測試需要一個有系統的途徑來創建及執行測試範例。特殊的模塊，比如訊號的產生及確認，促進了這種類型的測試程序。模型及程式碼的結構覆蓋率分析能幫助達成測試的完整度。模型經過測試之後，有幾種方法可以測試這些產生的程式碼。

軟體迴圈(Software-in-the-loop, SIL)測試是指在非即時建模環境下連同受控體模型執行控制器的生產程式碼。程式碼的執行與建模環境是在同一個主機上。與程式碼一起產生的編碼封裝則提供模擬與程式碼產生之間的介面(見圖 4)。

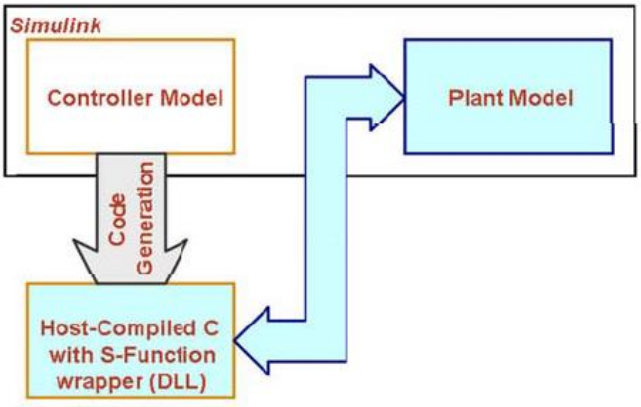

圖 4: 軟體迴圈測試(Software-in-the-loop testing)

處理器迴圈(Processor-in-the-loop, PIL)測試與 SIL 類似，也是去執行控制器的生產程式碼。然而這個程式碼是執行在實際的嵌入式處理器或指令集的模擬器，用這樣的方式在實際目標處理器上驗證程式碼。一台控制器區域網路(CAN bus) 或一連串的裝置被用來傳遞在處理器上執行的生產碼以及建模環境下的受控體模型資料。配合 SIL，PIL 可以用來進行非即時的生產程式碼測試(見圖 5)。

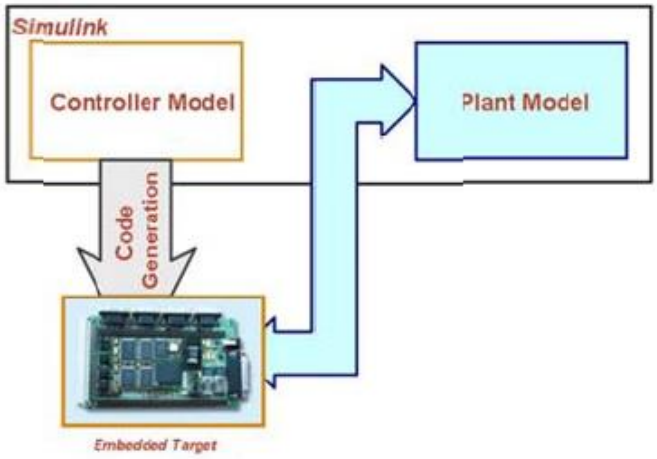

圖 5: 處理器迴圈測試(Processor-in-the-loop testing)

對硬體迴圈(hardware-in-the-loop)測試而言，所產生的程式碼是為了受控體模型而產生。測試在一個高決定性的即時電腦上執行。先進的訊號調節及電力電子被用來適當的刺激 ECU 輸入(感應器)及接收 ECU 輸出(執行器指令)。相較於快速原型化通常是一個開發或設計活動，HIL 被視為開始最終系統整合及現場測試前的一個實驗室測試階段(見圖 6)。

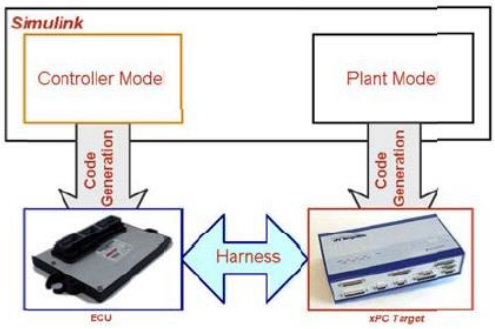

圖 6: 硬體迴圈測試(Hardware-in-the-loop testing)

整合活動 – 在模型化基礎設計中，有幾個整合活動跨越了整個開發週期，像是文件集。文件集，例如程式碼，可以自動地從模型中產生。文件集可透過範本來產生，工程師只要把每一個部分的文件集內容加入範本即可。若要追溯設計需求則可藉由模型中的模塊及現有設計需求管理來源之間的介面來完成。從模型產生的程式碼也可以被追溯回該模塊，幫助稽核員追蹤從高層級的設計需求一直到程式碼，以及再從程式碼追溯回到設計需求。有了需求管理，模型的來源控制(source control)可使用現有的來源控制產品從建模環境外部完成。模型化基礎設計也有建模環境及來源控制的介面可以幫助開發人員自動地記錄及檢查模型以及變更文件集。

漸進式 OTRP 途徑

針對嵌入式轉檔佈署的程式碼產生方式有兩種。其一為演算法輸出，產生函數程式碼接著把他們彙整到整個手寫的應用程式中。第二種為產生完整執行檔，使用模型來完整地產生整個應用程式。

演算法輸出方法較適合用於需要把模型導入到多個處理器，需要更多彈性的供應商。完整執行檔模型需要有裝置驅動模塊包含在模型中，即使驅動程式未經過模擬。完整執行檔的方法適用於不需要可移植演算法的 OEM 廠商，因為演算法會被鎖定在一個處理器架構中數年的時間。圖 7 為演算法輸出方式的說明。

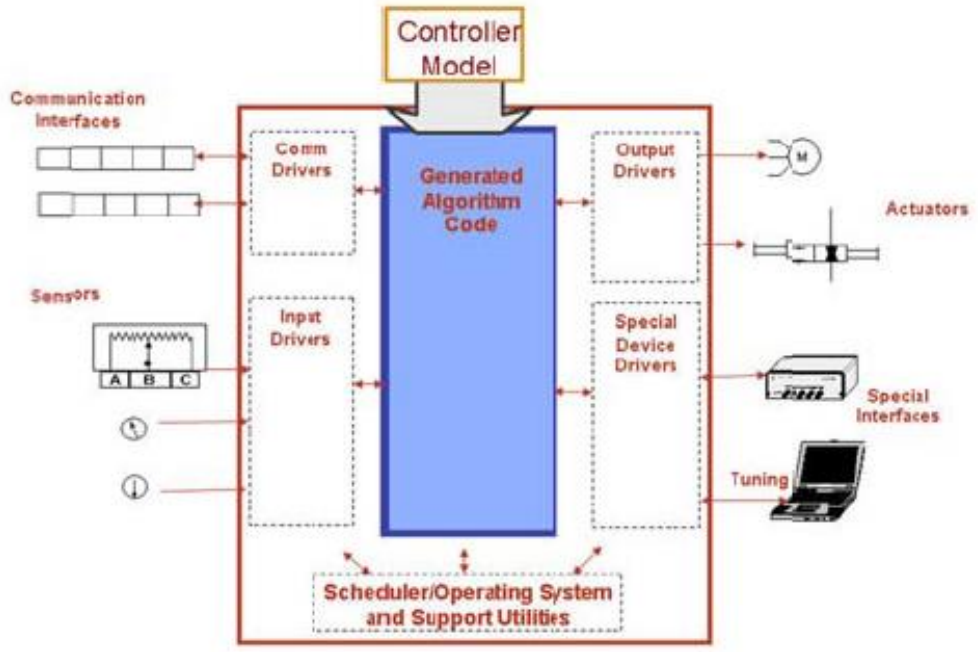

圖 7: 演算法輸出程式碼產生

以下描述了逐步執行 OTRP 的基本步驟。第一步為決定是否使用完整執行檔或演算法輸出技術。

不論選擇哪一種方式，建立模型準則及程式碼產生的設定都非常重要，這能讓設計更清楚、程式碼產生更有效率。Mathworks 公司的 Simulink Model Advisor 及 Code Generation Advisor 對這部分很有幫助。建立基底模型及進行浮點運算模擬，或必要時轉換為定點模擬也很重要。工具自動化對這個轉換過程將會非常有幫助。

簡短來說，將先前討論的兩種 OTRP 方法應用到初步的模型化基礎設計步驟如下：

1. 使用 Simulink Model Advisor 建立模型準則。
2. 使用 Code Generation Advisor 完成程式碼產生的設定。
3. 使用建模工具 Simulink 及事件導向系統模擬軟體(Stateflow)建立浮點演算法設計。
4. 轉換為定點資料，必要時使用定點轉換的自動化工具(Fixed-Point Designer)。
5. 比較定點及浮點演算的結果。

完整執行檔轉碼法 – 使用完整執行檔整合，特定目標處理器的模塊被加入到前一個步驟建立的通用演算法模型中。這個特定目標的模塊可以代表裝置的驅動程式或目標最佳化的函數。因特定目標的程式碼不能在主機處理器上執行，所以這些模塊通常會定出一個用來做模擬的默認值或傳遞行為。為了改善模型的可攜帶性，開發人員應該要把演算法的構成要素保留在一個與特定目標的架構模塊獨立的子系統或模型中。

在 Simulink，系統函數(system-functions 或 S-functions)被用來建立裝置驅動模塊。S-functions 可以經由 Simulink 的 Legacy Code Tool 手動或自動產生。這些 S-functions 必須一致才能把產生的程式碼最佳化，接著在無程式碼封包的狀態下叫出特定目標的程式碼。

程式碼替換函式庫(Code replacement library)被用來產生處理器最佳化的程式碼。一般來說，開發人員使用一個程式碼替換函式庫來繪製更優化版本的的基礎操作及數學函數。舉例來說，有了 TFL 便很容易可以利用編譯控制定向或硬體指令產生程式碼幫助在溢位保護下自動飽和。替換的程式碼也可以用來建立高度優化的三角函數。

除了S-functions及程式碼替換，開發人員需要建立一個客製的主要文件集以及可把建立過程自動化以調用交叉編譯的工具鏈。編譯、下載、執行都可以被完全的自動化。客製化的建立可以使用資源檔案範本、利用範本製作檔案，然後連結到APIs應用程式介面以幫助和控制建構程序。

最後，可以建立一個系統目標檔案來達成先前提到的完全客製化的建構解決方案。想了解更多與這些主題相關的細節可以參考Simulink Coder documentation [1]。

主要的步驟概述如下:

1. 建立一個系統目標文件(STF, System Target File)來基準化無驅動程式的目標。
2. 使用S-functions加入裝置的驅動模塊。
3. 使用目標函數函式庫優化程式碼。
4. 選擇STF來產生及編譯程式碼。
5. 下載程式碼並在目標處理器上執行
6. 使用外部模式或第三方校準工具來調整參數(選擇性)。

使用外部模式來調整參數需要一個主機與目標之間的溝通介面。當介面建立完成，外部模式的應

用程式介面(APIs)可以用來建立一個用進行互動式溝通的程式。Simulink提供了使用TCP/IP基本的範例及其他可用的選擇。

舉例來說，如要使用CAN，開發人員會需要主機CAN的支援產品，比如使用車載網路工具箱(Vehicle Network Toolbox)來與目標的CAN支援作溝通。一個CAN校準協議(CAN Calibration Protocol，CCP)模塊可以從草稿開發或從現有範例移植過來。在程式碼產生的同時，開發人員可以選擇建立ASAP2檔案這個選項來定義資料及記憶體位置。
見[2]，一個使用內建即時操作系統完整執行OTRP方法的範例。

演算法輸出法 – 利用完整執行檔的方法，開發人員應已建立含有程式碼產生設定的模型指南[3]以及浮點或定點設計。然而，裝置的驅動模塊並不會被建立因為這屬於手動編碼架構軟體。這個架構也會調用到產生的演算法程式碼。成功調用或整合的關鍵，在於建立適當的調用介面及參考而不是重新定義介面資料。

嵌入式程式碼轉碼器(Embedded Coder)提供許多選擇，可用來控制所產生程式碼的函數特徵。透過基本設定，轉碼器會像void-void函數使用全域資料產生初始化、階躍和終止函數。Simulink模型也會依據每個類別資料分別建立資料結構。在Simulink中匯入、匯出、參數，及說明建立的每一個變數都被分別放置在資料結構裡。每個結構的存放空間則由產生的程式碼分配。

開發人員可以變更這個基本設定的行為並且把指針轉向前先所提到的初始化、階躍，及終止函數的每一架構中。在這個案例，產生的程式碼不再依靠全域資料且可以再利用。開發人員負責申報每個資料結構的存儲。開發人員也可以明確地控制這個初始化、躍階、終止函數的原型。除了函數名稱，開發人員也可以指定參數名稱、傳遞機制(依參照或值)、及限定。用於控制函數原型的對話框如圖8所示。此範例為一個帶有四組匯入及一組匯出的Simulink模型。基本設定的void-void原型被改變為以多種方式傳遞匯入值，而Simulink匯出則被設定為一個函數回傳值。

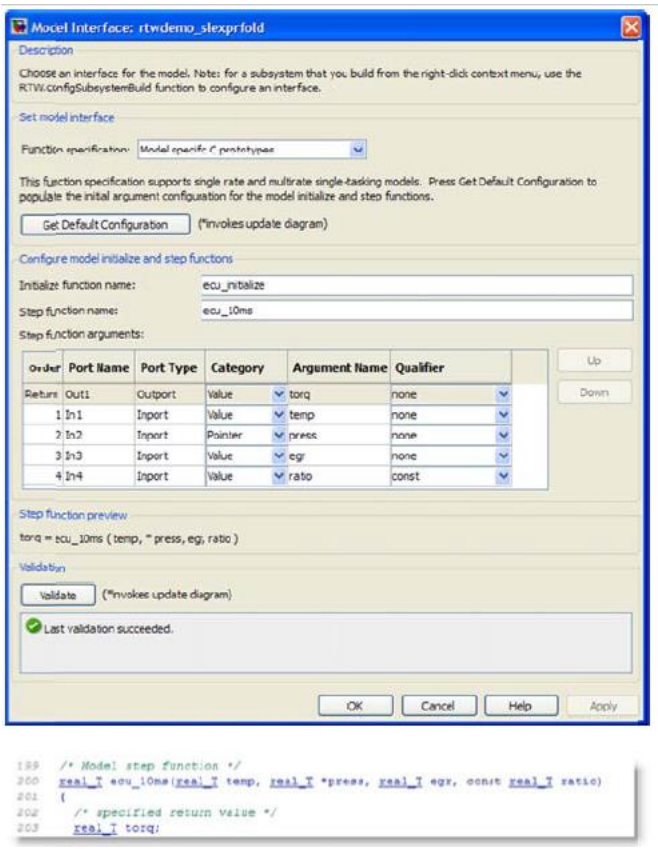

圖 8: 函數原型化控制及產生的程式碼

模型資料可以使用 Simulink 資料物件(Simulink Data Objects)、MPT 資料物件(MPT Data Objects)、及客製儲存等級(Custom Storage Classes，CSCs)來控制。一旦模型訊號或參數與訊號物件有關聯，它很肯定的會被分配到一個 CSC，例如一個全局變數或 get/set 存取法。輸入的外部 CSC 很有用，因為它可以參照現有資料項目(像是從模型外部一個單獨的數據字典所定義的校準參數)，在模型中使用這些資料項目，不需重新定義即可產生程式碼。

用於嵌入式系統轉檔佈署的演算法匯出主要步驟如下：

1. 產生具有適當調用介面的程式碼。
2. 建立匯入及匯出資料介面。
3. 利用外部框架程式碼來建立整合所需要的文件及函數。
4. 使用現有定序程式及生產建立程序以調用所產生的演算法程式碼。
5. 下載程式碼並在目標處理器上執行。
6. 利用第三方校準工具調整參數 (選擇性)。

從 OTRP 轉移到生產程式碼的考量

當 OTRP 框架建立完成，下一步自然是要考慮是否套用到生產上。有些活動可能需要重新定義，包含程式碼優化、程式碼驗證、認證、及標準編譯。

如同稍早所提到的，程式碼替換函式庫在 OTRP 為選擇性選項。但它對於生產卻很重要，因為程式碼需要高度優化來降低每單位的成本。這能夠幫助直接映射通常由程式碼產生器輸出的 ANSC 程式碼與特定 ECU 處理器支援的特定目標程式碼。

程式碼產生之後，驗證其行為是非常重要的。PIL 測試，如稍早所述，是一個有效的驗證法。PIL 在使用優化目標程式碼時特別重要，因為不可能在模擬期間於主機上做目標處理器程式碼測試。MathWorks 提供多種 PIL 解決方案，例如 PIL AIPs 可以幫助開發人員快速地建立他們自己的基於目標處理器的測試平台。

見[4]，TFL 與 PIL 測試的描述

汽車開放系統架構(AUTomotive Open System Architecture, AUTOSAR)代表了一個外部框架程式碼的特別案例。Simulink 對於這個新興的汽車標準有很好的支援。採用 AUTOSAR 時，開發人員可以省略很多先前提到的步驟，可選擇 Embedded Coder 中的 AUTOSAR 系統目標檔案取代之。他們接著可以使用一個執行環境(runtime environment, RTE)產生器為目標整合來導入程式碼產生過程中產生的 XML 描述。

另一個高信度生產 ECU 的新興趨勢為 ISO 26262 認證。參考[5]，其中描述了 Simulink 如何協助開發符合 IEC 61058 及 ISO 26262 的應用。

案例分析— AIPL

Automotive Infotronics Private Limited (AIPL)是 Ashok Leyland Limited 與 Continental AG 合資成立的公司。它的經營項目包含為運輸部門設計、開發、及改造電子產品與服務。該組織的宗旨是提供開發中市場的 OEM 廠商們，具有比市場上現有產品具備更佳價格/性能的產品。該公司開發產品涵蓋商用車輛應用相關的儀器應用程式、車身控制電子及其他多種控制單元[6]的電子零件與軟體。

車身控制單元/數據多工器及儀表板應用開發

AIPL 的前兩項旗艦商品是車身控制單元(BCU)/數據多工器(MUX)及儀表板應用(IC)，主要供應給印度一家重要的 OEM。這些產品的軟體從頭到尾都由內部自行開發。應用軟體完全利用 MathWorks 產品(MATLAB、Simulink、Stateflow 及 Embedded Coder)開發，裝置驅動層級的 BCU 部分則是使用 Freescale®的 CodeWarrior® IDE 開發，儀表板類部分則是 Fujitsu®的 SOFTUNE® IDE。這些模型的演算法的正確性透過模型迴圈(model-in-the-loop)模擬來做驗證。模型藉由使用 Simulink 定點產品被轉換為一個定點模型表示，SIL 測試被用來驗證目標字長限制及測試溢位條件。為了節省時間及減少建置錯誤的可能性，程式被寫成 MATLAB 語法便於生產程式碼從應用模型自動生成，以及編譯裝置驅動檔與應用程式檔。這些以 MATLAB 語法寫成的腳本還負責連接目標碼，並將其反映到目標處理器。

模型與建模架構

BCU(圖 9)、MUX(圖 10)、及 IC(圖 11)在一個整合的模式下一起運作。BCU 處理類比、數位及頻率輸入來導出驅動儀表板的值及直接驅動車輛附載的數位訊號。

這些模組互相交換他們運算的結果，並通過利用 Simulink 子系統的模組化方式來查看指令。使用非虛擬子系統來建立整個系統應用的模型，AIPL 裝配了代碼產生器(嵌入式編碼器)來產生流暢地連接手動編碼的定序程式代碼。

在這個過程，為了 CAN 校準協議正常運作，校準參數必須設為群組並放置在記憶體預定的區域。這是利用目標編譯器的#pragma 指令來達成。為了讓參數在產生程式碼時自動編組，AIPL 工程師把須要校準的參數列為一個客製儲存等級(CSC)。如此可使程式碼產生器把這個儲存等級類型的所有參數在一個單一指令下聚集。該指令的名稱則依照目標訂定。

圖 9: 車身控制單元 (BCU)

圖 10: 數據多工器 (MUX)

圖 11: 儀表板 (IC)

這個結構有一個很重要的特點，就是可以對目標進行線上校準。基於這個目的，CAN 校準協議(CCP)會透過 Simulink 外部模式及車載網路工具箱(Vehicle Network Toolbox)實行。就 BCU 而言，可轉換的參數包括了模組的定限、時間點、車輛參數、及感應器容許度。而就儀表板的運行關鍵-特定車輛的演算法，因為許多特定車輛的參數都需要經過調整，這個線上調整功能被證實對 OEM 特別有用。AIPL 提供一個基於 MATLAB 的圖形使用者介面(GUI)(圖 12)來幫助線上參數校準工作。

客製處理使用者輸入的 Simulink 模型環境的能力提供許多原本很難實現的設計功能。簡單來說，MathWorks 工具提供 AIPL 一個可進行演算法建模、迴圈模擬、驗證及確認、程式碼產生、建立及套用、以及校準 GUI 開發的單一建模環境。原本需要多個不同的工具，且帶有各自的相關成本及整合問題的與執行目標相連結的功能就此被整合。

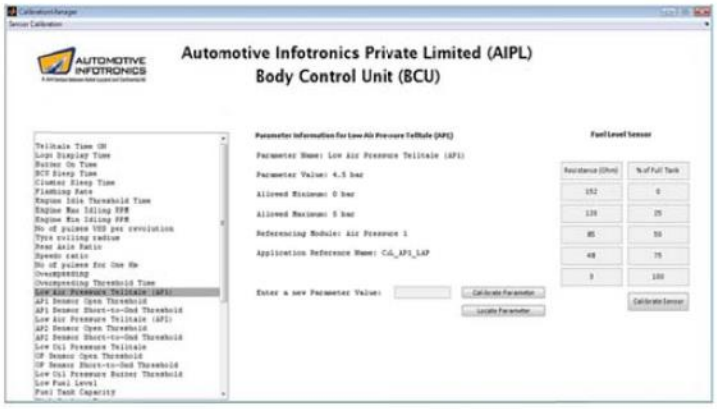

圖 12: 基於 MATLAB 依據 CCP 校準 BCU 的 GUI

測試及確認

開發完成之後，AIPL 產品會通過一個全面性的測試流程來確認是否符合客戶要求。在這些測試之後，產品還會在真實車輛上進行現場試驗。

項目結果及結論

模型化基礎設計節省了 BCU/MUX 及 IC 的應用軟體開發 40%的時間。而且與手動編碼相比，這個以模型為基礎的方式讓處理要求的變更及解決瑕疵變得更容易。AIPL 工程師使用 MathWorks 工具，以模型的形式指定系統要求，並藉由程式碼產生來實現。這在市場進入時間非常關鍵的情況下大幅縮短了產品開發時間。

結論

利用模型化基礎設計自動產生程式碼提供嵌入式系統開發人員多種建立原型、套用及驗證產品軟體的選擇。了解編碼產生可能的應用非常重要，因為僅僅應用這項技術並不能改善生產流程。嵌入式系統的開發人員必須建立一個充分發揮程式碼產生技術卻又符合完善軟體工程原則的生產流程，像是降低複雜性及建立合適的驗證確認及校準流程。

這個案例分析描述了 AIPL 如何利用模型化基礎設計來開發一個車身控制單元/數據多工儀表板並減少 40%的開發時間。

參考資料

[1] Developing Custom Target s, Simulink Coder User Guide, MathWorks, 2011, www.mathworks.com/access/helpdesk/help/toolbox/rtw.

[2] Automatic Code Generation – Technology Adoption Lessons Learned from Commercial Vehicle Case Studies, T. Erkkinen MathWorks, S. Breiner John Deere, SAEE Commercial Vehicle paper 08AAE-22, 20077, www.mathworks.com/mason/tag/proxy.html?dataid=9939&fileid=44540.

[3] Control Algorithm Modeling Guidelines Using MATLAB, Simulink, and Stateflow -Version 2.1, MathWorks Auto motive Advisory Board (MMAAB), 20099, www.mathworks.com/automotive/standards/mab.html.

[4] Fixed-Point ECU Code Optimization and Verification with Model-Based Design, T. Erkkinen MathWorks, SAE Congress paper 2009-01-0269, 2009.

[5] Qualifying Software Tools According to ISO 26262, M. Conrad and P. Munier MathWorks, F . Rauch TÜV SÜD, Model-Based Development of Embedded Systems, 2010.

[6] Ashok Leyland and Siemens VDO in Infotronics JV, Press Release, July 2007. www.siemens.co.in/en/news_press/index/news_archive/jul_16_2007.htm.