Formální metody návrhu software aplikované na embedded systémy

Platformně nezávislý zdrojový kód (3)

Michal Bližňák, Dušan Kolář

Platformě nezávislý kód

Při psaní programů po embedded zařízení se velmi často využívá přímého použití řídících registrů integrovaných periferií. Pomocí těchto registrů lze provádět operace s různými komunikačními rozhraními, vstupně výstupními obvody, časovači apod. Kámen úrazu spočívá především v tom, že u různých typů mikroprocesorů použitých v embedded zařízeních se způsob práce s těmito periferiemi a jejich adresy mění. Za předpokladu, že zdrojový kód generovaný ze stavového automatu využívá některé z těchto periferií, měli bychom být schopni zajistit, aby byl generovaný kód přenosný mezi různými cílovými platformami.

V oblasti klasických stolních počítačů je problém unifikovaného přístupu k různorodému hardware řešen použitím operačního systému, který poskytuje univerzální API (programovací rozhraní) a pomocí ovladačů sám přistupuje k řízeným periferiím. I ve světě embedded zařízení existují operační systémy, které nám mimo jiné umožňují využívat komfortu univerzálního přístupu k řízeným periferiím; tyto systémy však není možné nasadit úplně všude. Důvodem jsou hardwarová omezení cílových platforem, kdy můžeme být silně limitování například velikostí volné paměti, zásobníku, nebo výkonu mikroprocesoru. I v tomto případě však existují cesty, jak vytvořit určitou softwarovou vrstvu hardwarové abstrakce a zajistit tak, aby byl generovaný kód použitelný i na více cílových platformách.

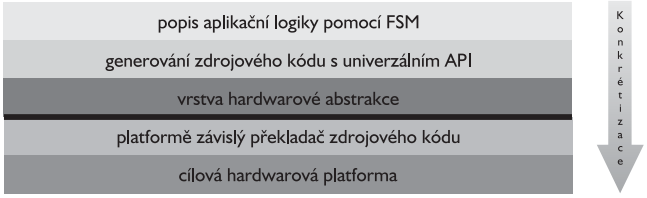
Na výběr máme ze dvou možností:

1. Využití nástrojů, které by poskytovaly univerzální API pro přístup k řízeným periferiím a tyto funkce by byly v průběhu překladu (nebo preprocesingu zdrojového kódu) nahrazovány přímým voláním řídících registrů cílové platformy.
2. Využití multiplatformní knihovny obsahující univerzální API, které by bylo implementováno pro každou cílovou platformu zvlášť.

Výhoda první možností spočívá zejména v menším objemu generovaného kódu. Nevýhodou pak může být poněkud pracnější způsob překladu aplikace. Naproti tomu, při použití multiplatformních knihoven odpadá nutnost použití více vývojových nástrojů, nevýhodou pak bývá méně optimalizovaný a objemnější zdrojový kód.

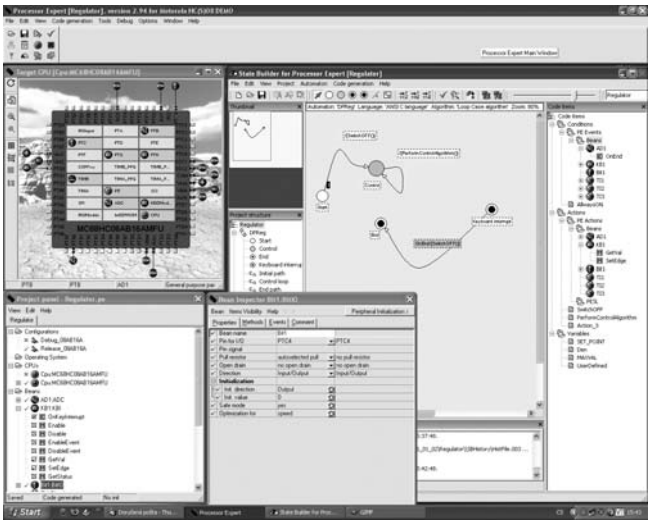
Od slov k činům

Až doposud jsme se v tomto článku pohybovali víceméně v teoretické rovině. Nyní je čas si předvést, že výše uvedené koncepce lze také úspěšně přivést do praxe. Tímto „důkazem" je například existence vývojového prostředí s názvem Processor Expert™ s jeho nástavbou State Builder. Samotné RAD IDE Procesor Expert™ již není třeba mnohým vývojářům softwarového vybavení pro embedded systémy představovat, jedná se o komerční produkt, který je na trhu již nějakou dobu a lze ho pořídit samostatně, nebo například v balíku společně se známým vývojovým prostředím CodeWarrior™. State Builder je však úplnou novinkou.



Obr.6 Platformě nezávislý vývojový proces využívající FSM pro popis aplikační logiky

Jak již napovídá samotný název tohoto nástroje, jedná se o rozšíření RAD IDE Processor Expert™ o možnost návrhu aplikační logiky (struktury programu) pomocí metody stavových automatů. Tento nástroj využívá všech poznatků uvedených v tomto článku a spolu s Procesor Expertem umožňuje vytvářet platformě nezávislé aplikace určené pro širokou škálu embedded zařízení.



Obr.7 State Builder pro Processor Expert™

Processor Expert společně se zásuvným modulem State Builder využívají první zmiňovaný způsob tvorby platformě nezávislých aplikací. Processor Expert poskytuje sadu univerzálních funkcí určených pro ovládání interních i externích periferií cílového MCU. Tyto funkce jsou nabízeny aplikaci State Builder, kde mohou být použity pro tvorbu těl akcí a podmínek stavového automatu popisujícího aplikační logiku vyvíjeného programu. Po vytvoření návrhu aplikace pak State Builder umožňuje provést verifikaci a optimalizaci struktury zdrojového stavového automatu a vygenerování odpovídajícího zdrojového kódu (tento programový kód lze generovat pomocí několika různých algoritmů).

Výstupní zdrojový kód aplikace je pak překládán prostřednictvím aplikace Processor Expert, který v průběhu překladu zaměňuje volání univerzálních funkcí na použití nativních registrů cílového MCU. Budeme-li pak v budoucnu chtít danou aplikaci přeložit pro jinou cílovou platformu, stačí tak pouze změnit typ použitého mikroprocesoru a aplikaci znovu z jejího formálního popisu vygenerovat a přeložit.

Co říci na závěr?

Smyslem toho článku bylo seznámení čtenářů s možnostmi a úskalími využití formálních metod návrhu aplikací při vývoji programového vybavení určeného pro embedded zařízení.

Jak jsme ukázali, i v tak specifické oblasti, jakou vývoj software pro embedded systémy bezesporu je, lze úspěšně využívat nejnovějších metod návrhu aplikací. Použití stavových automatů pro návrh embedded aplikací není samozřejmě možné ve všech případech, určitě je to však jedna z cest, které nabízí jednodušší, přehlednější a hlavně rychlejší vývoj tohoto typu software.

Výzkum a vývoj prostředků a postupů uvedených v tomto článku byl podporován granty Ministerstva školství České republiky (č.: MSM 7088352102) a České akademie věd (č.: 1ET400750406).

Literatura

[1] QING, L.: Real-Time Concepts for Embedded Systems, CMP-Books, 2003, ISBN 1-57820-124-1

[2] BARR, M.: Programming Embedded Systems, O'Reilly, 1999, ISBN: 1-56592-354-5

[3] CARROLL, J., LONG, D.: Theory of Finite Automata with an Introduction to Formal Languages, Prentice Hall, Englewood Cliffs, 1989.

[4] HOPCROFT, J. E., ULLMAN, J. D.: Introduction to Automata Theory, Languages and Computation, Addison -Wesley, 1979.

[5] ČERNÝ, S., KOLÁŘ, D., STRUŽKA, P.: Processor Expert, Component Application Builder for Embedded Systems, In: Proceedings of 5th IEEE Design and Diagnostics of Electronics Circuits and Systems Workshop, Brno, CZ, FIT VUT, 2002, s. 393-397, ISBN 80-214-2094-4

Ing. Michal Bližňák
doc. Dr. Ing. Dušan Kolář

Univerzita Tomáše Bati ve Zlíně, ČR
Fakulta aplikované informatiky
Ústav Aplikované Informatiky
e-mail.: bliznak@fai.utb.cz