Simulation & Code Generation for EKC Thermostat model

Kim Larsen Arne Skou Brian Nielsen Ulrik Larsen
Andrzej Wąsowski

Phase 0 Working Meeting

14 January 2004

Outline

- Simulation in IAR VisualSTATE Validator [demo]
- Code Generation [demo]
- Workstation Code execution [demo]
- Executable Size [Control Algorithm]
- Compilation Conditions
- The Interface Code
- The Generated Code

Executable Size [Control Algorithm]

All numbers for h8300 microcontroller:

model	**IAR VS 5.1** [bytes]	**SCOPE** [bytes]
minimal	1 748	1 312
aircond	2 426	1 668
ekc	4 134	est. 2 600

- The aircond model is a small mockup of the airconditioner shown at the last meeting.
- First two lines are the same as shown at the last meeting (marginally better).
- SCOPE result for EKC is estimated by proportional scaling of aircond1 size after substraction of the minimal model size.
- See the exact compilation conditions on the next slide.

Synthesis Conditions

- Generated with IAR VS Coder v 5.1
  - No merging of guards and transitions.
  - Data initialized using initializers.
  - Function pointer arrays for dispatch of guards and actions.
- SCOPE (snapshot release):
  - scope --release -cCF -cCstubs -cCdrv

CAUTION: Options are very dependent on the compiler, platform and the actual project.

Compilation Conditions

- Cross compiler: gcc-3.3.2
  - --target=h8300-hms --with-newlib   Thread model: single
  - newlib-1.11.0
  - binutils-2.14
- Compiler options: h8300-hms-gcc -Os -static -DNDEBUG -fomit-frame-pointer -foptimize-sibling-calls -Xlinker --relax
- No debug information
- Executables are stripped
- About 180 bytes exit code
  (not needed, but **included** here to avoid custom linker scripts).
- No drivers, no timer implementation, no RTOS, only standard entry code of gcc, etc
  - bare control-code + internal data + interface variables.

Interface Code (visualSTATE API)

```
int main( void ) {
 SEM_ACTION_EXPRESSION_TYPE ActionExpressNo;
 SEM_EVENT_TYPE EventNo = 0;
 SEM_Init();            /* initialize kernel */
 SEM_InitSignalQueue(); /* initialize signal queue */

while(1) {
  /* Fire event */
  if ( SEM_Deduct( EventNo ) != SES_OKAY ) break;
  /* Compute System Reaction */
  while (SEM_GetOutput(&ActionExpressNo) == SES_FOUND)
     SEM_Action( ActionExpressNo );
  /* Advance system's state */
  if (SEM_NextState() != SES_OKAY) break;
  /* Sense next event from the environment */
  EventNo = (SEM_EVENT_TYPE)Sample;
} }
```

Integration

- Use concurrent threads/hardware to write and read interface variables.
- or insert some sensor/actuator code into the main loop.
- or change the interface to more event-driven: translate environment events to model events and call actuator layer instead of changing output variables.

- visualSTATE types are macros or typedefs, but they correspond to C99 types
- Pointers are not allowed, but variables may be defined externally.
- Use compiler pragmas/linker sccripts to force allocation at specific addresses.
- or use a C preprocessor to modify the generated code.

Generated Code [Project Structure]

```
|--api /* static VS libraries */
|  |-- SEMLibB.c /* API implementation (‘‘kernel’’) */
|  |-- SEMLibB.h /* VS API prototypes */
|  ‘-- VSTypes.h /* definitons of VS types */
|
|--code /* files generated with VS coder */
|  |-- EKCThermostat.c       /* transition tables */
|  |-- EKCThermostat.h
|  |-- EKCThermostatAction.h/* action functions types */
|  |-- EKCThermostatData.c /* model code&internal data */
|  |-- EKCThermostatData.h
|  |-- SEMBDef.h
|  ‘-- SEMTypes.h
|
|--driver.c /* hand-made: main loop,actions, drivers,... */
‘--Makefile
```

Thank you for Your attention.