Per Knopfdruck vom Modell zum Code mit automatischer Generierung von Seriencode

Model to Code via Push-Button Production Code Generation

Michael Fröstl
Pilot Engineering, Production Code Generation
MathWorks

http://www.mathworks.de/embedded-code-generation/
switch(braindump) {

case ‘Applications’:
Pendolino tilting train.

The MEDUMAT Transport ventilator. Image © Weinmann Medical Technology.
case 'Programming':
case ‘Hardware’:
case ‘Operating Systems’:
Green Hills® Integrity RTOS

Texas Instruments™ DSP/BIOS™

OSEK-OS

Embedded Linux®

QNX® Neutrino® RTOS

Android™

Wind River® VxWorks®

Microsoft® Windows Embedded
case ‘Standards’:
default:
    printf(“Need a break?”);
}
MBD_Overview();
switch(topics)
{

case ‘Fixed-Point’:
void diffEq( void )
{
    /* Implements a fixed point
     first order difference equation */

    int Prod;
    long Accum;
    static short lastVal=0;
    short a=0x7eb8; /* 0.99 in s16,15 */
    short oneminusa=0x0148; /* .01 in s16,15 */
    short temp;

    Prod = gAlg_in1 * gAlg_in1;
    temp = Prod >> 15;
    Accum = a*lastVal + oneminusa*temp;

    gAlg_out1 = (short)(Accum >> 15);
    lastVal = gAlg_out1;
}
Fixed-Point Design Workflow

**Proof of Concept**
Design and simulate floating-point algorithms
Iterate on algorithm trade-offs

**Model Hardware Constraints**
Convert algorithm to fixed-point and simulate
Iterate on implementation trade-offs

**Verifying Fixed-Point Algorithms**
Verify fixed-point results against floating-point reference
Verify results against original requirements
case ‘Code Generation’:
\[
\begin{align*}
    &\text{int16} \\
    &1 \quad \varepsilon_a \\
    &\text{Add} \\
    &\text{int16} \\
    &+ \\
    &\text{int16} \\
    &+ \\
    &\varepsilon_c \\
    &1 \\
\end{align*}
\]
Algorithm Export (MAAB\textsuperscript{1} Model, MISRA C\textsuperscript{2})

\textbf{Input Drivers} \rightarrow \textbf{Generated Algorithm Code} \rightarrow \textbf{Output Drivers}

\textbf{Communication Interfaces} \rightarrow \textbf{Comm Drivers} \rightarrow \textbf{Generated Algorithm Code} \rightarrow \textbf{Special Device Drivers}

\textbf{Sensors} \rightarrow \textbf{Input Drivers}

\textbf{Tuning} \rightarrow \textbf{Scheduler/Operating System and Support Utilities}

1\textsuperscript{MathWorks Automotive Advisory Board Style Guidelines, v3.0}
2\textsuperscript{MISRA AC AGC: Guidelines for the Application of MISRA-C:2004 in the Context of Automatic Code Generation}
19 /* Model step function */
20 void very_simple_step(void)
21 {
22    /* Sum: '<Root>/Add' incorporates:
23     * Inport: '<Root>/a'
24     * Inport: '<Root>/b'
25     */
26    *c = a + b;
27   }
/* Model step function */

void Run_Cyclic(void)
{
    /* SignalConversion: '<Root>/TmpSignal ConversionAtcInport2' in
    * Inport: '<Root>/a'
    * Inport: '<Root>/b'
    * SignalConversion: '<Root>/TmpSignal ConversionAtaOutport2'
    * SignalConversion: '<Root>/TmpSignal ConversionAtbOutport2'
    * Sum: '<Root>/Add'
    */

    Rte_IWrite_Run_Cyclic_SPC_c(Rte_IRead_Run_Cyclic_RPa_a() +
                                Rte_IRead_Run_Cyclic_RFb_b());
}
Saturation: on
19  /* Model step function */
20  void very_simple_sat_step(void)
21  {
22      int32_T tmp;
23
24      /* Sum: '<Root>/Add' incorporates:
25         * Inport: '<Root>/a'
26         * Inport: '<Root>/b'
27      */
28      tmp = (int32_T)a + b;
29      if (tmp > 32767L) {
30          tmp = 32767L;
31      } else {
32          if (tmp < -32768L) {
33              tmp = -32768L;
34          }
35      }
36
37      *c = (int16_T)tmp;
38
39      /* End of Sum: '<Root>/Add' */
40  }
Saturation: on
Algorithm Export (MAAB Model, Target Code)
19 /* Model step function */
20 void easy_integrate_step(int16_T a, int16_T b, int16_T *c)
21 {
22     /* Output: '<Root>/<c>' incorporates:
23        * Input: '<Root>/<a>'
24        * Input: '<Root>/<b>'
25        * Sum: '<Root>/Add'
26     */
27     *c = c28x_add_s16_s16_s16_sat(a, b);
28 }
Full Executable (Target Model, Target Code)
/* Model step function */
void very_simple_io_wrapper_step(void)
{
    /* S-Function (c2802xadc): '<Root>/ADC' */
    {
        /* Internal Reference Voltage: Fixed scale 0 to 3.3 V range. */
        /* External Reference Voltage: Allowable ranges of VREFHI(ADCINA0) = 3.3
        AdcRegs.ADCSOCR1.bit.SOC0 = 1;
        asm(" RPT #22 || NOP");
        rtDW.a = (AdcResult.ADCRESULT0);
    }

    /* S-Function (c2802xadc): '<Root>/ADC1' */
    {
        /* Internal Reference Voltage: Fixed scale 0 to 3.3 V range. */
        /* External Reference Voltage: Allowable ranges of VREFHI(ADCINA0) = 3.3
        AdcRegs.ADCSOCR1.bit.SOC1 = 1;
        asm(" RPT #22 || NOP");
        rtDW.b = (AdcResult.ADCRESULT1);
    }

    /* ModelReference: '<Root>/Algorithm' */
    simple_integrate_step(rtDW.a, rtDW.b, &rtDW.c);

    /* S-Function (c28xsci_tx): '<Root>/SCI_Transmit' */
    {
}
void simple_integrate_step(int16_T a, int16_T b, int16_T *c) {
    /* Sum: '<Root>/Add' */
    *c = c28x_add_s16_s16_s16_sat(a, b);
}
case 'Targets':
Hardware Support

Connect MATLAB and Simulink to Hardware

Connect MATLAB® and Simulink® to hardware platforms for project-based learning, signal processing, computer vision, communications, data acquisition, instrument control, embedded systems, and more. Use low-cost hardware platforms like Arduino® and Raspberry Pi™, embedded systems like ARM®- and Zynq®-based architectures, or high-end systems for real-time testing. Find the right solution to help you prototype and implement your applications with MATLAB and Simulink.

Search Hardware Support

Explore supported hardware by:

**Vendors**
Select vendor

**Applications**
Select application

**Product Families & Products**
Select product or family

Or enter a keyword:
Training Services

Developing Embedded Targets
Advisory Service
case ‘Getting ARMed’:
Create Model
Field-Oriented Control of Permanent Magnet Synchronous Machine
System Test Bench

System Inputs

Processor

Inverter and Motor

Discrete

Embedded Processor

System Analysis

Copyright 2010-2012 The MathWorks, Inc.

Controller Mode Scheduler

Motor Control
- position_valid

Startup_Open_Loop_Control
controller_mode = Startup;
torque_command = ctiParams.StartupCurrent;

Velocity_Control
controller_mode = VelocityControl;
en, du:
velocity_command = command;

Position_Control
controller_mode = PositionControl;
en, du:
position_command = command;

Torque_Control
controller_mode = TorqueControl;
en, du:
torque_command = command;

[error]

[command_type == Velocity]

[error]

[command_type == Position]

[error]

[command_type == Torque]

[abs(velocity) < __]
ctiParams.RampToStopVelocity

Ramp_To_Stop
controller_mode = VelocityControl;
en, du:
velocity_command = 0;
torque_command = 0;

Simulink

Stateflow
Simulate and Test (on Host)
Field-Oriented Control of Permanent Magnet Synchronous Machine System Test Bench

Model Description: Field-Oriented Control of Permanent Magnet Synchronous Machine

Demonstrates a Field-Oriented Control algorithm with Space Vector Modulation for a Permanent Magnet Synchronous Machine (PMSM). The test bench can be used to evaluate the system performance. Examples include turning the motor on, calibrating for a valid rotor position, transitioning to closed loop operation, and changing speed and torque during closed loop control. The Embedded Processor subsystem contains the controller algorithm (which supports C code generation) as well as simulation models of peripherals.
Generate ARM Optimized Code
/* Trigonometry: '<S14>/sine_cosine1' */
cos_coefficient = arm_cos_f32(sin_coefficient);
Processor-in-the-Loop Execution Profiling
Field-Oriented Control of Permanent Magnet Synchronous Machine System Test Bench

Copyright 2010-2012 The MathWorks, Inc.

Host

Cable

Target

STM32F
Add I/O Blocks, Generate Code, and Deploy!
Field-Oriented Control of Permanent Magnet Synchronous Machine System Test Bench

STM32F4xx Blockset
default:
    printf(“Time for a break!”);
}
INTEGRATION
IMPLEMENTATION
TEST & VERIFICATION
RESEARCH
REQUIREMENTS
DESIGN

Environment Models
- Mechanical
- Electrical

Supervisory Logic
Control Algorithms

IMPLEMENTATION

C, C++, VHDL, Verilog, Structured Text

MCU, DSP, FPGA, ASIC, PLC