Target Support Package Development for In house Prototyping ECU

By
Tata Motors
Engg Research Centre
Advanced Engineering Department

Team Members:
Sarkar Prasanta, Dr Nitin Bhiwapurkar, Geetanjali Khot, Chaitali Nilawar
Agenda

- Usage of Matlab/Simulink tool chain
- Need for TSP
- Aim
- Benefits of TSP
- In House TSP development
- TSP Development Steps
- Break Through
- Hurdles crossed: Interrupt Implementation
- Implementation of Single Step Code generation process
- Milestones
- Results
Tata Motors is extensively using Matlab Simulink for software development for Hybrid and EVs.
Benefits of MATLAB Simulink

- Less Development Time
- Early error detection and correction
- Auto code generation feature
- Systematic approach
- Changes can be made fast and easily
- Reusable Code
- Ensures quality and Timeliness
Problems faced

- Simulink Auto Code generation feature only applicable to control strategy
- Requires manual integration of device driver c code with control strategy
- Code build procedure is complex and requires several steps
- Complete code cannot be generated on same platform i.e. Simulink

**Major Hurdle**

Unavailability of device driver on model level.
Code build process without TSP

MANUAL STEP 1
Copy auto generated C code files to Project in code warrior IDE
Integration of device driver c code with auto generated control strategy
Invoke compiler
Hex file Map File A2I File

MANUAL STEP 2

MANUAL STEP 3

MANUAL STEP 4
Invoke matlab command to copy the addresses from the map file to A2I file

Non simulink environment
Code warrior IDE
Freescale Device drivers
Device driver c code (non auto generated code)
Control strategy c code
Control Strategy Simulink Model
Matlab Simulink Environment
Code Warrior IDE environment

Matlab Simulink Environment
Aim

- To enable model based design approach for in house prototyping ECUs.
- To develop single step code generation process.
- Hide the complexity of the build procedure for domain engineers.

Hence TSP and automation script was needed.
How TSP and M scripts help

**TSP: Target Support Package** are Simulink blocks that incorporates the device drivers for the target controller.

- With TSP device drivers are brought on the model level
- It help in auto code generation of complete code i.e. control strategy + device driver code

**M Scripts**

- Helps in automating the build process and generate hex and A2l file in single step
- M script runs at the background to generate code in single step
Code Build process with TSP

Simulink Model

- Inputs
  - CAN RX block
  - LIN RX
  - Digital in
  - Analog channel

- Outputs
  - PWM out
  - Digital out
  - CAN TX block

TSP blocksets (Device drivers on model level)

Click Build button

- Model.c
- Model.h

Processor expert project template

RTW folder
- Model.c
- Model.h
- Model_private.h

BCM device drivers folder

- Model.c Calls the device driver functions

Compile the project
Linking the project

Project.map file
Project.s19 file
Project.a2l file

Automated Build Process Using .m scripts

Executable and a2l files are generated
Code Build process with TSP

Simulink Model

Inputs
- Analog config block
- PIT block
- CAN Config Block

Control Strategy
- CAN RX block
- LIN RX
- Digital in
- Analog channel

Outputs
- PWM out
- Digital out
- CAN TX block

TSP blocksets (Device drivers on model level)

Simulink Environment

Click Build button

Executable and a2l files are generated

Project.s19 file

Project.a2l file
Benefits of TSP

- Easy to make changes
- Single step code generation process
- Reduces development time
- Design reuse
- Enables MBD design approach use of in house prototyping ECUs
- Provides common platform for device driver and control strategy development
- Hides complexity of device driver code
- Rapid prototype development
Why In house development?

- Unavailability of TSP for specific controller used in the in house prototyping ECU
- Cost advantage
- Readymade TSPs are Black box
- Readymade TSPs cannot be used for in house prototyping ECUs
- Core competency development
- Availability of embedded team and infrastructure
TSP Development Steps

1. C code development
2. Modification to the C code function
   - sfnc_generate .m script
   - Module.c
   - Module.h
3. Legacy code tool (MATLAB)
   - Generates blockset and its related files
4. Simulink block (sfunction)
   - S function wrapper file (.c file)
   - S function target language compiler file (.tlc file)
   - S function simulation file .mex file
5. TSP block set generated
Break through

Use of Matlab legacy code tool was a major break through

Advantages of Legacy code tool

- Easy to use
- Faster than the traditional approaches - sfunction builder or writing s function methods
- Use the existing C code for generation of TSP blocks.
- The s-function developed using legacy code tool generates optimized code during auto code generation when used with embedded coder.

Use of Freescale processor expert

- Help auto generate device driver c code
- Eliminated tedious hand written c code.
Hurdles crossed: Interrupt Implementation

- How to develop a Simulink block that takes user defined routine and places it in a ISR.

- The legacy code tool generated block *.tlc files only calls the device drive functions after every time step.

- So interrupt s-function and tlc files generated needed to be modified
  - To generate function from the user defined routine
  - To call this function on interrupt
Interrupt Implementation

- Interrupt Priority (Decimal value)
  7 - Highest Priority
  1 - Lowest Priority

- ENABLE INTERRUPT
  - Enable PIT Timer 0 Timeout Interrupt
  - Enable PIT Timer 1 Timeout Interrupt
  - Enable PIT Timer 2 Timeout Interrupt
  - Enable PIT Timer 3 Timeout Interrupt
  - Enable LIN Receive Buffer Full Interrupt
  - CAN4 Receive Buffer Full Interrupt

- INTERRUPT PRIORITY (1-7)
  - PIT Timer 0 Timeout Interrupt Priority 0
  - PIT Timer 1 Timeout Interrupt Priority 0
  - PIT Timer 2 Timeout Interrupt Priority 0
  - PIT Timer 3 Timeout Interrupt Priority 0
  - LIN Receive Buffer Full Interrupt Priority 0
  - CAN4 Receive Buffer Full Interrupt Priority 0
Interrupt Implementation

Legacy Function (mask) (link)
The Interrupt block selects interrupt service routine (ISR).

A function call subsystem is required at the output of this block. It will act as an ISR.

This block calls the function call subsystem (ISR) after the corresponding interrupt is generated.

Example:

If timer0 timeout interrupt is activated in interrupt configuration block then select PIT_tim0_timeout_call isr from the drop down menu of this block.

The function call subsystem placed at the output of this block acts as an Timer 0 timeout ISR.

The logic that is required to be executed after timer 0 timeout interrupt generation should be put in this function call subsystem.

Parameters

<table>
<thead>
<tr>
<th>Interrupt Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN4_Rx_Full</td>
</tr>
<tr>
<td>PIT_tim0_timeout</td>
</tr>
<tr>
<td>PIT_tim1_timeout</td>
</tr>
<tr>
<td>PIT_tim2_timeout</td>
</tr>
<tr>
<td>PIT_tim3_timeout</td>
</tr>
<tr>
<td>LIN_rx_buffer_full</td>
</tr>
<tr>
<td>CAN4_Rx_Full</td>
</tr>
</tbody>
</table>
Interrupt Implementation

```c
/* Output and update for function-call system: '<Root>/CAN4_Rx_Full' */
void ISRFUNC_CANRX(void)
{
    // User code

ISR(CAN_RX)
{
    /* Write your interrupt code here ... */
    ISR_FUNC_CANRX();
}

The generated function is hardcoded
```
Implementation of Single Step Code generation process

- Automation script for single step auto code generation is implemented through a Build Model button.

  - Build Model button Functionality:
    1. Automatically configures the model configuration parameter as required for ECU.
    2. Build the model (Simulink). Generates code and .a2l without address mapping.
    3. Invokes compiler
    4. Copies the auto generated code from model into the template project of IDE (which has all driver codes)
    5. Compiles the code
    6. Generates .s19 (hex) file and .map file
    7. Calls the post gen commands to copy address from .map file to .a2l file.
Implementation of Single Step Code generation process

Build Model

Build_CW_IDE button in background calls Build_CW_IDE.m script

Build_CW_IDE.m

Call the function S12x_TSP_Config_Set() to set model configuration parameter for freescale IDE

rtwbuild(bdroot) Build model generate c code of current model

Call the function CW_IDE_link

Copy .s19 .a2l .map file from their respective folder to user current directory

Copy addresses from .map file to .a2l by calling the script tml_asap2post.m

.a2l and .s19(hex) are ready for flashing the freescale ECU

Copy generated code of model to the template Code warrior project

Compile the project generate .s19(hex) and .map
TSP Block generation:
All peripheral block generation through legacy code tool.
Working blocks for every peripheral are generated.
Milestones Contd...

- **Interrupts:**
  The tlc file was modified such that block is not called at every time step.
  The block which can take user code and create a function from it was developed.
  To invoke call to function call subsystem which contained user code.

- **Testing:**
  Unit testing is performed on every block.

- **Help Documents:**
  Help document is linked to the block help button on the TSP block.
  Help document explains each parameters, input, output and functionality of block. It provides examples of how block can be used in Simulink model.
Milestones Contd...

- **Library:**
  TSP was made a part of Simulink library
  Can be used as any other Simulink blocks from library.

- **Automation:**
  Automation script for single step auto code generation.

TSP Library added in Simulink Library
Results

- In house developed TSP is used in model along with control strategy. Thus TSP has brought Device driver on model level.

- Executable code is generated within a single step.

- Complex build process is hidden from user.

- The code generated is flashed in the in house prototyping ECU.

- It is used successfully in Hybrid car project under development.
THANK YOU!
QUESTIONS?