Generic Timer Module Configuration Tool
AMF-ENT-T1039

John H. Floros
Senior Software Engineer
Agenda

• **Overview: 40 minutes**
  - Introduction and Objectives
  - Overview of Generic Timer Module
  - Overview of GTM Configuration Tool
  - Matterhorn/GTM and GTM Configuration Tool Interaction

• **GTM Configuration Tool Examples Demos: 70 minutes**
  - Building code with the GTM Configuration Tool
  - TOM Example Demos: SIMPLE_IRQ
  - ATOM Example Demo: ATOM_SOMP
  - DPLL Example Demo: DPLL_DPLL_ACTION

• **Summary and Q&A: 10 minutes**
Introduction: WHAT DO WE DO?

- The Automotive Silicon Support Tools group’s objective is to develop software enablement tools to assist our customers with rapid prototyping and accelerate algorithm development on their target Freescale MCU.

- This includes software tools that automatically generate peripheral initialization code through GUI configuration, to generating peripheral driver code from a Model Based Design environment like Simulink™.
Objectives

• After this GTM Training, you will know:
  
  − What the Generic Timer Module (GTM) is and what features it contains for Powertrain and Motor Control applications
  
  − How the GTM and MCU interact at a high level
  
  − What the GTM configuration tool can do to help accelerate the ramp up and development of Qorivva MCUs that contain the GTM IP.
  
  − Understand the flexibility of being able to customize the GTM configuration tool to fit your coding conventions with the use of project settings and template files.
Overview - GTM Motivation

• In Powertrain applications there are two dominant timer implementation methods:
  - One uses a more peripheral timer approach where the timer module consists of capture/compare units and counters (ex. EMIOS/Etimer). This approach has the problem of the main core needs to service the interrupts from the timer module interrupting other processing.
  - The second approach uses a more software processing oriented approach with a micro-machine or co-processor that is programmable and fulfils timer specific tasks (ex. eTPU). This approach often has lower resolution for signal processing and is difficult to program because of special instruction sets.
• The GTM combines both the approaches. While there are some submodules that can fulfill a specific function in hardware, there is a RISC-like processing engine build within the GTM that can do a kind of signal processing and flexible signal generation using a special instruction set. Therefore, while the peripheral timer submodules offer real time processing capabilities, the RISC-like processing engine adds flexibility to the GTM.
Overview - GTM Purpose

- The GTM is a large scalable timer with a modular design and a central routing unit allowing for flexibility in channel numbers and application specific modules.
  - Designed for 4 to 8 cylinder applications
  - Powertrain, transmission control and some motor control
  - Specific application sub-modules include Angle clock hardware, safety functions and motor commutation sub-modules
- The GTMs purpose is to unload the I/O core by allowing tasks to run independently with run once set up at MCU initialization.
Overview - GTM Architecture

- The GTM is built up from various sub-modules with each having dedicated functionality (six types):
  - Data movement (ARU, BRC & PSM)
  - Time base (TBU & CMU)
  - I/O Modules (TIM, TOM & ATOM)
  - Programmable core (MCS & MCFG)
  - Special purpose sub-modules (DPLL, MAP & SPE)
  - Safety related (CMP & MON)
- Submodules can be combined through ARU in a scalable and configurable manner to implement complex timer systems suitable for many different applications.
Overview - GTM Example

- Diagram of In Cylinder Pressure Sensing in angle domain using GTM and ADC without any CPU load.
Overview - GTM Benefits

• Most functions performed in parallel with dedicated hardware units, ensuring simple latency calculations.

• Reduced interrupt load removing the need for low-latency interrupts.

• CPU can be run with a slow clock in low-end projects giving low power dissipation and low EMI.

• Less data traffic between CPU and GTM due to dedicated hardware – Timers, ARU, programmable cores and engine position hardware.
- Each Source has a unique and fixed address
- ARU configures the destination address
- Data from a Source can only go to one Destination (See BRC)
Overview - Broadcast Module (BRC)

- Duplicates data streams
- 12 inputs, 22 outputs
Overview - CTBM (TBU, DPLL, CMU, MAP)

- Clock and Time Base Management Sub-modules
  - Time Base Unit
    - Generates 3 time bases
  - Clock Management Unit
    - Generates 12 sub module clocks
    - Generates 3 external clocks
  - Digital PLL Module
    - Frequency Multiplier i.e increased precision of position
  - TIM0 Input Mapping Module
    - Generates TRIGGER and STATE for DPLL
Overview - Parameter Storage Module (PSM)

- Comprised of 3 sub units
  - AEI-to-FIFO Data Interface (AFD)
  - FIFO (a.k.a RFO)
  - FIFO-to-ARU Interface (F2A)
- Data storage for incoming data characteristics or as parameter storage for outgoing data
- This data is stored in RAM that is logically located inside the FIFO subunit
Overview - Timer Units

• TIM – Timer Input Module
  - Filters and captures input signals e.g.
    - Time stamp of event
    - Number of edges
    - PWM duration or period
  - Can be routed through ARU and SPE for processing

• TOM – Timer Output Module
  - PWM generator
  - Linked to SPE

• ATOM – ARU connected TOM
  - Generates complex output signals
  - Output signal characteristics routed through DPLL or MCS or PSM
Overview - Multi Channel Sequencer (MCS)

- This Data Processing Module can calculate complex output sequences based on TBU and ATOM
- There are Custom 24-bit RISC-like programmable cores inside of the GTM.
  - Fine grain temporal multi-threaded
  - Von Neumann (common bus) architecture
  - 32-bit fixed instruction width
  - Optimized instruction set
    - 24-bit data operations
    - Channel flow control
- Simple and complex triggering between channel
- Processes TIM data
- Process CPU sourced data
- These cores have their own internal RAM where the code and data can be stored.
Overview - Memory Configuration (MCFG)

- Organizes physical memory blocks and maps them to MCS submodules
- Defaults to 6K (4K + 2K) for each instance of MCS
- SWAP and BORROW for more or less memory
Overview - Sensor Pattern Evaluation (SPE)

- Supports BLDC engines by evaluation Hall sensor inputs (TIMx3)
- Drive BLDC with TOM
- Can be used at input to MAP for electric engine control
Overview - Deadtime Module (DTM)

- Deadtime generation in hardware to support motor control
  - Electrical drive, charger control and sensor evaluation
  - PWM Phase Shifting
Overview - Output Compare Unit (CMP)

- For use in Safety applications
- Ensures duplicate outputs match, if not generates an error
Overview - Monitor Unit (MON)

- Supervisor for use in Safety applications
- Monitors ARU and CMU (via Activity Checker – AC) activities
Overview - GTM Configuration Tool

• The GTM Configuration Tool is developed by Freescale to help enable the development of software for Freescale MCUs that contain the GTM by allowing the user to configure the GTM thru a Graphical User Interface (GUI) and then using the settings to automatically generate initialization code for the GTM which is run once by the CPU on initialization.

• The GTM Configuration Tool provides the means to configure all the individual register sets with the target goal of supporting specialized timer input and output signals for specific application goals like 4, to 8 cylinder applications for powertrain, transmission and motor control, including angle clock hardware and motor commutation.

• The GTM Configuration Tool is Eclipse based and is available as a standalone installation or a plug-in to existing Eclipse based Integrated Development Environments.

• The tool will allow the user to save multiple configurations in project files that can be later recalled. The user will also be able to export and import signals within projects to allow reuse in other GTM Configuration Tool projects.

• As part of the GTM Tool install package there are many examples included with pre-written CPU start up code to allow the user with the use of the examples to start seeing signals using the GTM very quickly. The examples support builds for the main Qorivva Compilers.
The GUI layout is in three sections:

**Project** – Project, Sub-Module Navigation, and Signal Explorer.

**Signal/Data Flow** – Will contain the signals/data flows built for a project and what elements are used.

**Register** – This is where the user will manipulate the registers of the GTM to obtain the desired signal/data flow desired.
Overview - GTM Configuration Tool Project Section

Project Section

The Project Section will contain the list of projects that are open thru the use of panes. Under each project there will be the following panes:

Sub-Module Navigator
Will contain a list of the all the sub-modules down to the channel level so that the user can select the desired element and drag that element to the Signal Section to activate for use and configuration.

Signal Explorer
Will contain a list of the all the Signals/Data flows constructed by the user down to the sub-module level so that the user can view/select all of the signals and channels used in a project in one view.
**Signal Section**

The Signal Section will contain the signals of a project with each signal on a different pane. The user is able to create new signals and name them. When a signal pane is selected the corresponding register panes become active in the Register Section. Register panes not associated with the signal are not shown. Under each signal there will be the following panes:

- **Description**
  Will contain a list of the all the sub-module down to the channel level that the user has selected for their signal/data flow. When selecting the elements in the navigation tree the corresponding register tab will become active and displayed for the user to configure. Also, there will be a text box available for the user to put a description of the signal/data flow.

- **Block Diagram**
  Will contain a block diagram of the GTM and will highlight only the blocks that the signal configuration is using. The rest of the blocks in the diagram will be grayed out.
Register Section

Contains the panes sub-modules down to the channel level. When a signal pane is selected the corresponding register panes become visible and selectable. Register panes not associated with the signal shall not be visible. Other panes in the Register Section are:

Description
Contains the register/bit description of the last configurable element that the mouse pointer was/is hovering over.

Problems
Contains the results of the last consistency check preformed which is a list of settings that the user may not have considered when configuring the GTM. Also, will generate links to the settings flagged taking the user directly to the problem location.

Console
This pane contains the results of the last code generation performed (ex. which files were generated, location of generation, and any problems that occurred).
Project Preferences

This menu which is accessible from the GTM Tool pull down will allow the user to customize how the code is generated. The user can modify the file names for the code files generated and the function names as well.

The user can also create their own code template file location to allow customization of the initialization code itself while still preserving the original template files that came with the GTM tool.

You can then point the GTM tool to use the default code templates for code generation or the user specific code templates.
## Overview - GTM Configuration Tool Toolbar

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create New Project</td>
<td>Invokes the New GTM Project wizard for project parameters definition.</td>
</tr>
<tr>
<td>Load Project</td>
<td>Opens the Load Project dialog, the project can be selected.</td>
</tr>
<tr>
<td>Save All Signals</td>
<td>Saves all signals in current selected project.</td>
</tr>
<tr>
<td>Close Project</td>
<td>This command closes the currently selected project.</td>
</tr>
<tr>
<td>Create Signal</td>
<td>Creates new signal.</td>
</tr>
<tr>
<td>Import Signal</td>
<td>Imports signal.</td>
</tr>
<tr>
<td>Export Signal</td>
<td>Exports signal.</td>
</tr>
<tr>
<td>Delete Signal</td>
<td>Deletes signal.</td>
</tr>
<tr>
<td>Generate Code</td>
<td>Start code generation for the project.</td>
</tr>
<tr>
<td>Consistency Check</td>
<td>Start consistency checking.</td>
</tr>
<tr>
<td>Help</td>
<td>Open GTM tool help</td>
</tr>
</tbody>
</table>
The GTM configuration tool will generate software to download an image for the MCS cores of the GTM. The GTM tool does not generate a source image for MCS cores; however, the GTM will generate initialization code for the MCS configuration registers.

The tool does not create a build environment for the user but the software that is auto-code generated is tested for functionality on Qorivva compilers. As mentioned previously the example code provided with the installation does have a build and make using the main Qorivva compilers.

Consistency check support is available so that a list of settings can show the user options that may not have been considered when configuring the GTM.
Key Functional Characteristics

- Two independent 300 MHz Power Architecture z7 computational cores
- Single 300 MHz Power Architecture z7 core in delayed lockstep for ASIL-D safety
- Single I/O 200 MHz Power Architecture z4 core
- eDMA controller – 128 channels
- 8M Flash with ECC
- 596k total SRAM with ECC
  - 404k of system RAM (incl. 64k standby)
  - 192k of tightly coupled data RAM
- 10 ΣΔ & 12 SAR converters – 84 channels
- Ethernet (MII/RMII)
- DSP1 – 8 channels (3 supporting µSec ch.)
- LINFlex - 6 channels (3 supporting µSec ch.)
- MCAN-FD/TTCAN – 4x modules/1x module
- GTM – 248 timer channels

Key Electrical Characteristics

- -40 to +125 °C (ambient)
- 165 °C junction for KGD
- 1.26V Vdd, 5.0V I/O, 5V ADC

Package

- 292 PBGA, 416 PBGA, 512 PBGA
- eCal emulation device for each package
The GTM subsystem requires 2 clocks to operate.
All GTM I/O signals are connected via the multiplexing in SIU to pins on the device.

- The System Integration Unit Lite2 (SIUL2) provides control over all the I/O ports on this device and supports 13 ports with 16-bits of bidirectional, general-purpose input and output signals.

- Peripheral and input multiplexing assignments

Table in the McKinley RM details the MSCR

- See TIM 0 CH0 example here

### Table 271. Peripheral and input multiplexing assignments

<table>
<thead>
<tr>
<th>Instance</th>
<th>Input</th>
<th>SIUL2 MSCR Register</th>
<th>Source Signal Select (MSCR[SSS])</th>
<th>Source Instance</th>
<th>Source Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTM</td>
<td>TIM0_0</td>
<td>512</td>
<td>0000_0000</td>
<td>-</td>
<td>disable low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000_0001</td>
<td>IO_PAD</td>
<td>PD[14]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000_0010</td>
<td>IO_PAD</td>
<td>PE[4]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000_0011</td>
<td>IO_PAD</td>
<td>PF[1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000_0100</td>
<td>IO_PAD</td>
<td>PB[9]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000_0101</td>
<td>IO_PAD</td>
<td>PF[10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000_0110</td>
<td>IO_PAD</td>
<td>PF[13]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000_0111</td>
<td>IO_PAD</td>
<td>PH[3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000_1000</td>
<td>IO_PAD</td>
<td>PH[10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000_1001</td>
<td>IO_PAD</td>
<td>PH[13]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000_1010</td>
<td>IO_PAD</td>
<td>PM[9]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000_1011 - 0010_1111</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0011_0000</td>
<td>SDADC_0</td>
<td>WATCHDOG_LIM IT_TRIGGER</td>
</tr>
</tbody>
</table>
GTM Example Projects

- You can immediately start generating code with the GTM Configuration Tool by opening one of the available examples.
GTM Loading an Example Project

- Select GTM > Load Project to open the load dialog. All GTM Configuration Tool examples are located at: {GTM tool installation root}\Examples.
- Use the Browse button and select the examples directory.
- The list of available example projects will be displayed, select the project(s) you would like to use.
- Click Finish.
GTM Creating a New Project

• Specify project name in the **Project name** field, the directory with this name will be created, so make sure that project name is convenient for project configuration management and build system.

• The project location either can be specified explicitly in the **Location field**, or defined workspace can be used – then check the **Use default location** checkbox.

• In the **Select project device** pull down list select required device name. The list of devices might depend on tool version.

• Click **Finish** when all parameters are specified, new project will be listed in the **Project view**
GTM Generating Code

• After opening an example, invoke the code generation (GTM > Generate) of the project to obtain all sources. In the Code Generation Wizard dialog select the project and click Finish.

• The results of the build are displayed in the Console pane.
The generated code (outlined in Red) is placed in the Code sub-folder of the project.
Building a project using example provided

- The example projects contain main application code which initializes the MCU, some pin configuration used by the example, etc.

- To build the executable the Examples directory contains build_<compiler>.bat and makefiles which are used to generate a binary. The parameter to start the batch file is the path to the project.
Building a project using example provided make

- For example, to build the SIMPLE_IRQ project for the TOM module with Greenhills compiler, type the following command:

  ```
  build_ghs.bat
tom\SIMPLE_IRQ
  ```

- When the build is complete the executable files will be created in the bin subdirectory.
Timer Output Module (TOM)

- Contains 16 output channels
  - Arranged into 2 8-channel groups for synchronous PWM output
  - Channels are 16-bits wide
- Generates PWM outputs
- 0% & 100% duty cycle supported of up to 20kHz with a 0.025% resolution
- There is complex triggering to support start/stop of PWM signals, output enable/disable and update of PWM period and duty (synch and asynch) values, and Force Immediate Update
- 5 dedicated prescaler clocks can be used
- Greater than or equal compares used
- One dedicated channel to generate pulse code modulated (PCM) signal
One Channel Architecture

Counter Compare Unit 0 (CCU0)

counter compare unit 1 (ccu1)

signal generation unit (sou)

output logic

3 key registers

CN0 is the Timer counter
CM0 is the compare (frequency/period)
CM1 is the duty cycle control
TOM – PWM generator

- Complex triggers sources:
  - TBU0,1,2 values
  - CPU
  - Internal triggers
- Complex trigger mechanism for synchronous:
  - Start/Stop of PWM signal generation
  - Output enable/disable functionality
  - Update of period and duty cycle
  - Forced Immediate Update
- Synchronous and asynchronous update of duty cycle
- Continuous and single shot mode of operation
TOM SIMPLE_IRQ Demo

• Description:
  − This demo will initialize TOM 0 Channel 0 to generate PWM with frequency 2500Hz and duty cycle 50%. Interrupts occurs on CCU0 and CCU1 trigger.

• Setup:
  − Configure GTM Configuration Tool (see next slides)
  − Build application by one of the following commands depends on compiler:
    ▪ build_ghs.bat tom\SIMPLE_IRQ_Demo
    ▪ build_diab.bat tom\SIMPLE_IRQ_Demo
    ▪ build_htc.bat tom\SIMPLE_IRQ_Demo
  − Load application into the target start execution.
Clocks are derived from main GTM-IP module input clock.

Generates 13 sub module and 3 external clocks.

3 sub units:
- External Generation Unit (EGU)
- Configurable Clock Generation Subunit (CFGU)
- Fixed Clock Generation (FXU)
TOM SIMPLE_IRQ Demo – CMU

- Generates 8 sub module clocks
  - TIM, ATOM, TBU and MON
- Global Clock Divider like External Clock Divider
- Clock Source n Divider changes duty cycle
- CH6 and CH7 differ from other channels as SUB_INCn [DPLL] can be used as a clock enable

\[ T_{SYS\_CLK} = \frac{1}{f_{SYS\_CLK}} \]

\[ T_{CMU\_CLK[x]} = \frac{1}{f_{CMU\_CLK[x]}} \]
TOM SIMPLE_IRQ Demo – Time Base Unit

• Provides common/global time bases for GTM

• The time base channels can run independently of each other and can be enabled and disabled synchronously by control bits in a global TBU channel enable register
TOM SIMPLE_IRQ Demo

Summary of steps:
1. Create New Project “SIMPLE_IRQ_DEMO”
2. Create a New Signal “PWM”
3. Drag in the CMU Fixed Clock Signal to be used as the clock for PWM
4. Drag in the TOM channel to configure the output of the PWM
5. Perform Consistency Check
6. Correct any Errors
7. Generate Code
TOM SIMPLE_IRQ Demo – Create Signal PWM

This demo will initialize TOM 0 Channel 0 to generate PWM with frequency 2500Hz and duty cycle 50%. Interrupts occurs on CCU0 and CCU1 trigger.
TOM SIMPLE_IRQ Demo – Drag in the CMU

Enable all fixed clock signal

FXCLK_SEL - Clock source selection: CMU_GCLK_EN
CMU_FCLK0 Clock Frequency (Hz) 4000000
CMU_FCLK1 Clock Frequency (Hz) 2600000.0
CMU_FCLK2 Clock Frequency (Hz) 168250.0
CMU_FCLK3 Clock Frequency (Hz) 9785.625
CMU_FCLK4 Clock Frequency (Hz) 610.36087566785689

This demo will initialize TOM0
Channel 0 to generate PWM with frequency 2500Hz and duty cycle 50%. Interrupts occurs on CCU0 and CCUI trigger.
TOM SIMPLE_IRQ Demo – Drag in the TOM CH

This demo will initialize TOM 0 Channel 0 to generate PWM with frequency 2500Hz and duty cycle 50%. Interrupts occurs on CC0 and CC1 trigger.
TOM SIMPLE_IRQ Demo – Configure TOM CH
TOM SIMPLE_IRQ Demo – Consistency Check

Consistency check completed with no errors! 08-22-2013 04:32:07
TOM SIMPLE_IRQ Demo – Consistency Check

Consistency check completed with 1 error(s)! 08-22-2013 04:33:55
1: Clock source CMU signal CMU_FXCLK for TOM CLK_SRC_SR is disabled
TOM SIMPLE_IRQ Demo – Generate Code

---

Code Generation: Success
Created 95 Files
Consistency Check Errors: 0
Code Generation Errors: 0
TOM SIMPLE_IRQ Demo – Program and Run

• Results on Oscilloscope:
  - PB[9] – Pin control by PWM TOM 0 channel 0 output
  - PA[1] – Pin Toggles when TOM 0 CCU0 trigger occurs
  - PA[3] – Pin Toggles when TOM 0 CCU0 trigger occurs
ATOM – ARU Connected TOM

- Contains 8 output channels
- Similar to TOM channels except
  - 24-bits wide
  - Connected to ARU
  - More modes and complex triggers
- 4 channel output modes
  - Signal Output Mode PWM (SOMP) – TOM mode
  - Signal Output Mode Immediate (SOMI)
  - Signal Output Mode Compare (SOMC)
  - Signal Output Mode Serial (SOMS)
- As on TOM, complex trigger mechanism for **synchronous**:
  - Start/Stop of PWM signal generation
  - Output enable/disable functionality
  - Update of PWM signal characteristics
The Signal Output Mode PWM (SOMP) is principally the same as the output mode for the TOM sub module except PCM mode is not included in the ATOM.

It is possible to reload the shadow registers over the ARU without the need of a CPU interaction.
ATOMIC SOMP Demo

• Description:
  - Sample initializes ATOM 0 Channel 0 and Channel 1 to SOMP mode.
  - ATOM0 CH1 get signal parameters from FIFO CH0.

• Setup:
  - Configure GTM Configuration Tool (see next slides)
  - Build application by one of the following commands depends on compiler:
    - build_ghs.bat atom\SOMP
    - build_diab.bat atom\SOMP
    - build_htc.bat atom\SOMP
  - Load application into the target start execution.
ATOM SOMP Demo Uses the PSM – AFD

- Data interface between the AEI bus and the FIFO, of eight logical FIFO channels
- One Buffer Access Register AFD[i]_CH[x]_BUFF_ACC which CPU access.
Configurable (per ch) storage unit:
- Size (start and end address)
- FIFO operation modes (normal or ring buffer operation mode)
- Fill level control / memory region read protection

Organized as a single RAM (29 wide) which is mapped into the address space of the MCU
In ring buffer mode the FIFO provides a continuous data stream to the F2A sub module. The first word of the FIFO is delivered first and after the last word is provided by the FIFO to the ARU, the first word can be obtained again.

If the application needs to change some data inside the continuous data stream this can be done through direct memory access provided by the FIFO AEI interface.
The F2A has to distribute data from and to the FIFO channels in a configurable manner.

Data transfer between FIFO and ARU is organized with eight different streams that are connected to the eight different channels of the corresponding FIFO module.

Within these streams the F2A can transmit/receive the lower, the upper or both 24 bit values of the ARU together with the ARU control bits according to the configuration.
ATOM SOMP Demo Data Transfer Overview
Summary of steps:

1. Create New Project “SOMP_DEMO”
2. Create a New Signal “PWM”
3. Drag in the CMU Configurable Clock Signal to be used as the clock for ATOM channel.
4. Drag in the TOM channel to configure the output of the
5. Perform Consistency Check
6. Correct any Errors
7. Generate Code
ATOM SOMP Demo – Create Signal PWM
ATOM SOMP Demo – Drag in the CMU
ATOM SOMP Demo – Drag in the ATOM CH 0
ATOM SOMP Demo – Configure ATOM CH 0
ATOM SOMP Demo – Drag in the ATOM CH 1
ATOM SOMP Demo – Configure ATOM CH 1
ATOM SOMP Demo – Drag in the FIFO CH 0
ATOM SOMP Demo – Configure FIFO CH 0
ATOM SOMP Demo – Generate Code

---

**Code Generation: Success**
- Created 95 files
- Consistency Check Errors: 0
- Code Generation Errors: 0

---
Results on Oscilloscope:
- PC[2] - ATOM 0 CH 0 output
- PC[0] - ATOM 0 CH 1 output
Timer Input Module – TIM

- Contains 8 24-bit input channels with dedicated input filters and timeout units for each channel
- Dedicated filter mechanism with two different filter strategies and edge filter thresholds for each channel
- Shadow registers to hold measurement data while new input signal is processed
- Control by CPU and/or ARU possible
- Five different edge characterization modes configurable
  - Measure duty & period
  - Measure active time of signal
  - Count edges
  - Prescaler mode
  - Bit concentrator mode
TIM – Architecture

- Contains 8 24-bit input channels with dedicated input filters and timeout units for each channel
TIM Channel Architecture

- Data stored in GPR0 and GPR1
- Edge Counter (ECNT)
- Clock Counter and Shadow (CNTS)
- Various Interrupt Events
TIM TPWM_NEWVAL_IRQ Demo

• Description:
  – Sample initializes TIM 0 Channel 3 to PWM Measurement Mode.
  – Input signal is generated on PA[1] pin by PIT 0 Channel 2 Interrupts.
  – When input signal rising edge is detected then TIM_NEWVAL_IRQ is raised.
  – Interrupt handler reads registers containing Duty Cycle and Period value in CMU_CLK_0 ticks and recalculate them into microseconds and percentage.

• Setup:
  – Start GTM plugin, open "project" and generate code for the project.
  – Build application by one of the following commands depends on compiler:
    ▪ build_ghs.bat tim\TPWM_NEWVAL_IRQ
    ▪ build_diab.bat tim\TPWM_NEWVAL_IRQ
    ▪ build_htc.bat tim\TPWM_NEWVAL_IRQ
  – Load application into the target start execution.
DPLL

• Transforms an input signal into a higher frequency/resolution signal called a micro tick which is dependent on TRIGGER and STATE inputs.

• Module is highly configurable and does a lot of calculations.

• Angle clock for automotive applications:
  - Configurable profiles for camshaft and crankshaft.
  - Angle calculations and information centralized in DPLL.
  - Position minus time prediction (AN013).
  - Plausibility checks of input signal conditions.
  - Supports normal and emergency (loss of Cam/Crank) modes with transition between them.
  - Integrated RAM blocks (3 kB – 13.5 kB) contain position and increment duration history of crankshaft and camshaft PWM signals.

• Encoder support for Permanent Magnet Synchronous electric motors:
  - 2 channels with 3 HALL sensors each with support for missing input signal (hybrid support).
DPLL Tasks

- Predict of the duration of the current increment
- Generate of SUB_INC1,2 pulses for up to 2 position counters in normal or emergency mode
- Seamlessly switch to emergency mode and back to CPU control
- Synchronize actual position (under CPU control)
- Predict position and time related events
DPLL Block Diagram
The DPLL is configured to generate 4 micro ticks per input tick (one tick from rising to rising edge) on the TRIGGER input signal. The four micro ticks are generated for the intervals a, and b correctly.
The DPLL predicts for interval c the same adder value. But the input signal frequency for interval c decreases.

When the DPLL is programmed in Automatic End Mode, which can be controlled by DMO bit, the sub_inc1c output ticks will stop after 4 ticks were generated.
Another adder value is calculated which results in a smaller frequency of the sub_inc signal. Now, since the input signal accelerates, there are not enough micro ticks generated.
To compensate, the DPLL offers two possibilities. One possibility is to distribute for the next interval evenly six micro ticks or to generate two micro ticks fast and the regular 4 micro ticks evenly in the next interval. If the micro ticks should be generated fast, the CMU_CLK0 is used as tick frequency.
The micro tick generation holds for signal `sub_inc1c`. As can be seen from the figure, `sub_inc1` is always generated with the frequency calculated from the last increment duration. Therefore, the `sub_inc1` ticks do not reflect the physical position of the TRIGGER input signal.
How to turn GTM on

• The GTM module is gated off out of reset
• The GTM Integration Registers are located at GTM Base Addr + 0xC0, with the Module Disable (MDIS) bit in the Module Config Register
  - This bit must be cleared to enable writes to the GTM registers for configuration before operation
• Next, the top level GTM Configuration Registers should be set
  - These include the GTM Control, Bridge Mode and IRQ registers
• In general, the next stage is the Clocks
  - The CMU sub module controls the GTM clocking
  - The TBU can then be configured
Summary

• Q&A