Hands-on Workshop: Designing Applications with the S12 MagniV Family

AMF-AUT-T0665

Derrick Klotz
Regional FAE
Senior Member of Technical Staff
Agenda

- Automotive Body Electronics
- MagniV Solutions and Roadmap
  - Lab 1
    - Getting Started with CodeWarrior 10.4

- S12Z Enhanced CPU Core
- S12ZVM for BLDC Motor Control
- S12ZVL for General Purpose LIN
- S12ZVC for General Purpose CAN

- Lab 2
  - Developing a Project for the S12ZVL

freescale.com/automotive
Automotive Market Trends – Sensors and Actuators

- **Reduction of power consumption**
  - LIN to reduce cables weight
  - Smarter motor control techniques
  - Smarter and more sensors

- **Reduction of physical size**
  - Electro-mechanical integration
  - High temperature >125°C Ta

- **High Growth**
  - Driven by affordability of LIN
  - Driven by comfort and convenience features
What’s Behind Body Electronics?

**Comfort Features**
- Door Module
- Window Lift
- Seat Module/position
- HVAC
- Electric sunroof/shade
- Interior lighting

**Vehicle Networking**
- Central Body Control Module
- Central Gateways: CAN, LIN, Flexray, Ethernet, MOST

**Safety Related**
- Steering column lock
- Steering column adjustment
- AFS
- Anti-pitch power windows
- Wipers and rain sensors

**Security**
- Immobilizer
- Keyless Entry
- Preventing hacking and counterfeit modules

- Body systems embrace a broad variety of applications inside the cabin
- Body systems cover the widest range of performance requirements
  - Small 8-bit controllers and watchdogs
  - General purpose 16-bit controllers
  - High performance 32-bit compute engines
Body Market Trends and Challenges

Connected
- More complex Gateways with higher performance and multicore usage (Expanding memory)
- Ethernet and wireless communication
- Personalization options driving LIN nodes

Safe and Secure
- Functional Safety - ISO26262
  Several body-apps need ASIL A/B (some C/D)
- Reduced ASIL-assessment effort
- Security/Cryptography for Gateway and BCM modules

Green
- Power management in stop and run modes
- Autosar SW management of partial/pretended networking
- EC-motors, LED-lighting

Cost Efficient
- Cost reduction via ECU integration
- Electrification of the car replacing mech. components
- Scalability of hardware and software
- Auto generated code to decrease dev costs

Software Integration:
- Autosar:
  - Multicore-support
  - OS
  - MCAL
- Safety:
  - SW-routines supporting self-test (Core/Memory)
- Security:
  - Cryptography algorithmic support
- Application-support
  - Motor Control-library
  - Reference designs
## Automotive Microcontroller Portfolio

<table>
<thead>
<tr>
<th><strong>MPU</strong></th>
<th>i.MX</th>
<th>Multimedia processors</th>
<th>Multicore and graphics performance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>32-bit MCU</strong></td>
<td>Qorivva</td>
<td>Highest-performance automotive MCU</td>
<td>Multicore with integrated safety and security</td>
</tr>
<tr>
<td><strong>16-bit MCU</strong></td>
<td>S12</td>
<td>An industry-leading 16-bit architecture</td>
<td><strong>MagniV: Most integrated MCU</strong></td>
</tr>
<tr>
<td><strong>8-bit MCU</strong></td>
<td>S08</td>
<td>15-year automotive workhorse</td>
<td>Most optimal and robust solution</td>
</tr>
</tbody>
</table>

### Target Applications
- Center stack
- Instrument cluster
- Telematics
- Powertrain
- Safety & chassis
- Central body
- Vehicle networking
- Body electronics
- Instrument cluster
- LIN slaves
- Sensor/Actuators
- Switch and touch panels
- Companion MCUs
What is S12 MagniV?

SIP = System in Package

Semi-Discrete Solution
- Standard MCU
- Application Specific Analog IC (ASIC)

Multi-die SiP
- Single package
- Die-to-die bonding

Monolithic SiP
- MCU and Analog on the same die
Over 15 Years of System in Package (SiP) Experience

- **1997**: Start of SiP R&D
- **1998**: Monolithic Architecture
  - HC05 Microcontroller with EEPROM (IDR60% -1.2um)
  - Mechatronics Package
- **1999**: 1st Gen
- **2000**: 2nd Gen
  - S12Z Core (0.25um)
  - Architecture Repartitioning
  - SMOS8 (0.25um)
  - QFN Package
- **2001**: 3rd Gen
  - S12 Core (0.25um)
- **2002**
- **2003**
- **2004**
- **2005**
- **2006**
- **2007**
- **2008**
- **2009**
  - 912F634
    - 16-bit S12I32 + SMOS8 Relay & Switch Driver
- **2010**
- **2011**
- **2012**
- **2013**: 4th Gen
- **2014**
- **2015**

**Products**

- **908E622/1**: 8-bit HC08 + SMOS5 Mirror Driver
- **908E624**: 8-bit HC08 + SMOS5 Relay Driver
- **908E626**: HC08 + SMOS5 AFS Stepper Driver
- **908E624/1**: 8-bit HC08 + SMOS5 Relay Driver

**Technologies**

- **S12ZVL**: First S12 MagniV product:
  - SMOS8 (0.25um)
- **LL18UHV technology**: Monolithic SIP
  - S12 Core (0.25um)
LL18UHV Technology Summary for S12 MagniV

Digital Logic
S12, PWMs, Timers, SRAM, SPI, SCI, GPIO, Watchdogs, etc.

High-Voltage Analog
Low Side & High Side Drivers, Voltage Regulator LIN/CAN Phy. etc.

Non-Volatile Memory
Flash, EEPROM

Existing
Low Leakage 180nm CMOS+NVM

40V UHV Devices
## MagniV building blocks

### High-Voltage Components
- VREG for tot. supply:
  - 70mA w/o ext comp. or
  - 170mA with ext. ballast
- LIN PHY
- CAN PHY
- V-BAT SENSE
- V-SUP SENSE
- HVI (12V-input with ADC)
- LS-drivers
- HS-drivers
- Charge Pump
- 4-6ch MOSFET Predriver (50-100nC)

### Digital Components
- MS-CAN
- SPI
- GPIO
- PGPIO 20mA
- NGPIO 25mA
- BDM/BDC
- Key Wakeups
- Win Wdog
- Timer 16Bit (25-64MHz)
- PWM 8/16Bit (25-50MHz)
- Motorcontrol PWM With Fault protection
- Prog. Trigger Unit
- Sound Generator
- Segment LCD (4x40)
- Stepper Motor Driver with SSD

### MCU Core and Memories
- S12- or S12Z-CPU 25/32/50MHz bus
- 10-12Bit resolution
- 1-2 S/H-units
- Up to 16ch total
- Flash (ECC) 8kB – 128kB
- EEPROM (ECC) 128B – 4kB
- RAM (ECC) 512B - 8kB
- ADC
- Temp Sense
- Current Sense (2 x Op-Amp)
- Pierce Osc.

### 5V Analogue Components
- ADC
- 10-12Bit resolution
- 1-2 S/H-units
- Up to 16ch total
- Charge Pump
- 4-6ch MOSFET Predriver

### Packaging
- LQFP: 32/48/64/100/144-pin
- LQFP-EP: 48/64-pin
- QFN: 32-pin (5x5mm)
S12 MagniV Roadmap

Motor Control
- S12VRxx
  2 LS for relay based DC motor control
- S12ZVML/Cxx
  6ch-MOSFET pre-driver for BLDC/PMSM motor control
- S12ZVMxx
  No PHY (PWM-control) + 6ch-MOSFET pre-driver

Instrument Cluster
- S12ZVHxx
  LCD + Gauge
- S12ZVF
  LCD
- S12ZVH
  LCD + Gauge

Multi-PHY companion chip

LED Lighting
- S12ZVLxx

General Purpose
- LIN enabled
- CAN enabled

With LIN-PHY
No comms (PWM-control)
With CAN-PHY

freescale.com/MagniV
Agenda

Automotive Body Electronics
MagniV Solutions and Roadmap

Lab 1  ___________
Getting Started with CodeWarrior 10.4

S12Z Enhanced CPU Core
S12ZVM for BLDC Motor Control
S12ZVL for General Purpose LIN
S12ZVC for General Purpose CAN

Lab 2  ___________
Developing a Project for the S12ZVL
Launch CodeWarrior 10.4

① Select a workspace
The C/C++ Perspective

② Select “New MCU project”
Start a New Project – “Lab”

③ Select a Project name

④ Next >
Select the Target Device

⑤ Type “S12ZVL”

⑥ Select “MC9S12ZVL32”

⑦ Next >
Select the Debug Connection

8. Only Select "Open Source BDM"

9. Next >
Language and Build Options

1. Select the language:
   - C
   - C++
   - Mixed C and ASM
   - ASM

2. Select the floating point format supported:
   - None
   - Float is IEEE32, Double is IEEE32 optimized
   - Float is IEEE32, Double is IEEE32 compliant
   - Float is IEEE32, Double is IEEE64 optimized
   - Float is IEEE32, Double is IEEE64 compliant

3. Select the memory model:
   - Small
   - Medium

4. Click Next to proceed.
Enable Processor Expert

⑪ Select “Processor Expert”

⑫ Finish
Double-Left-Mouse-Click on “Component Inspector - Cpu” tab
Select “Expert”

Expand “Clock settings”
S12 Clock, Reset and Power Management Unit

Table A-19. IRC electrical characteristics

<table>
<thead>
<tr>
<th>Num</th>
<th>C</th>
<th>Rating</th>
<th>Symbol</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P</td>
<td>Junction Temperature - 40 to 150 Celsius</td>
<td>f_{IRC1M_TRIM}</td>
<td>0.987</td>
<td>1.00</td>
<td>1.013</td>
<td>MHz</td>
</tr>
</tbody>
</table>

25MHz Bus
50MHz CPU Core
Enter “25” MHz

Double-Left-Mouse-Click on “Component Inspector - Cpu” tab
Build Project “Lab”

- Expand “Lab: FLASH”
- Build ‘FLASH’ for project ‘Lab’
Investigate Processor Expert Generated Code

Expand “Generated Code”
...investigate Processor Expert Generated Code – “Cpu.c”

Double-Left-Mouse-Click on “Cpu.c”

Double-Left-Mouse-Click on “Cpu.c” tab
Double-Left-Mouse-Click on “Cpu.c” tab
Disassemble “Cpu.c”

Right-Mouse-Click on “Cpu.c” and Select “Disassemble”

Double-Left-Mouse-Click on “Cpu_cxxxx.lst” tab
Investigate Assembly Language Code

Double-Left-Mouse-Click on “Cpu.c” tab

Close “Cpu_cxxxx.lst” tab
Open the “Window” Pull-Down Menu then Select “Open Perspective” then Select “Hardware”
The Hardware Perspective

Either Close the “Hardware” Perspective or Switch back to the “C/C++” Perspective
Ready for the Next Part of the Lab
Agenda

- Automotive Body Electronics
- MagniV Solutions and Roadmap
- Lab 1
  - Getting Started with CodeWarrior 10.4

- S12Z Enhanced CPU Core
  - S12ZVM for BLDC Motor Control
  - S12ZVL for General Purpose LIN
  - S12ZVC for General Purpose CAN
- Lab 2
  - Developing a Project for the S12ZVL
The S12Z CPU is the next generation of CPU in the CPU12 line.

- High-speed 16-bit processor with an expanded programmers model
- Improved addressing modes support efficient use of the 16MB (24-bit) linear address space.
- Improved support for C code-size efficiency and overall performance
Linear vs. Paged Addressing Mode

• S12 MCUs typical architecture provides 64KB memory space
  - To achieve extended memory sizes, a paging mechanism is needed
    ▪ 256 pages of 16kB each = 4MB maximum
  
  - Paging is one of the biggest dissatisfiers from customers’ perspective
Linear vs. Paged Addressing Mode

• The S12Z CPU includes a 24-bit address bus, which provides up to 16MB memory space

• Linear addressing simplifies memory accesses
S12Z – Absolute Addressing

16k Space
2 Address Bytes for all operations

256k Space
3 Address Bytes for all operations

16Meg
3 Address Bytes for LD, ST, JMP, JSR
4 Address Bytes for all other operations
### S12(X) CPU

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>8-bit Accumulators A &amp; B or 16-bit Accumulator D</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td></td>
<td>Index Register X</td>
</tr>
<tr>
<td>IX</td>
<td></td>
<td>Index Register Y</td>
</tr>
<tr>
<td>IY</td>
<td></td>
<td>Stack Pointer</td>
</tr>
<tr>
<td>SP</td>
<td></td>
<td>Program Counter</td>
</tr>
<tr>
<td>PC</td>
<td></td>
<td>Condition Code Register</td>
</tr>
</tbody>
</table>

- **U**
- **0**
- **0**
- **0**
- **IPL[2:0]**
- **S**
- **X**
- **H**
- **I**
- **N**
- **Z**
- **V**
- **C**
S12Z CPU

Data Register 0
Data Register 1
Data Register 2
Data Register 3
Data Register 4
Data Register 5
Data Register 6
Data Register 7

Data Registers A & B
16-bit Accumulator D

Index Register X
Index Register Y
Stack Pointer
Program Counter
Condition Code Register

$2^{24} = 16 \text{Mbyte Address Space}$
S12Z Core Architecture … a 16-bit MCU?

- It uses **32-Bit** data paths, ALU, Data registers
  - Single-cycle 16x16 multiply (2.5 cycles 32x32)
  - MAC unit 32bit += 32bit*32bit (3.5 cycles)
  - Hardware divider 32bit = 32bit/32bit (18.5 cycles)
  - Single cycle multi-bit shifts (Barrel shifter)
  - Fractional Math support

- It uses a **24-bit** address bus, Stack Pointer, Program Counter and Index registers

- Its native “int” data type is **16-bit**, with 16-bit I/O data path

- It can handle **8-Bit** data and indexes better than S12X

- CPU operates at **100MHz**
  - Optimized bus architecture with 100MHz load and store to RAM
  - NVM works with 1 Wait-state => effective 20ns accesses

- Harvard Architecture => parallel code and data access
  - Instructions and addressing modes optimized for C programming and compiler
S12Z Optimized Opcode Set

- Most used instructions are 8-bit opcodes
  - First Page

LD, ST, MOV, MUL, ADD, SUB, AND, OR, CMP, BSET
Shifts, BCLR, BTGL, INC, DEC, BRA, JMP, JSR

- Less used opcodes are 16-bit and require the prefix
  - Second Page

MAC, MIN, MAX, ABS, DIV, MOD, ADC, SBC, EOR
Bit Operations

• Bit operations were massively changed
  - BSET, BCLR, BRSET, BRCLR test only a single bit to save code space
    ▪ Bit takes a maximum of 5 bits instead of a mask for 32Bit takes 4 bytes!
    ▪ >90% of all codes test/set/clear only one bit!
  
  - The bit number is either an immediate value or defined by an accumulator.
  
  - They can operate on a memory location but also on an accumulator.
  
  - The size of a memory operand can be 8, 16 or 32-bits or an accumulator
    ▪ Example:
      \[ j = 1 << n; \Rightarrow \text{CLR D2}; \text{LD D3}, \#n; \text{BSET D2}, \text{D3} \]
  
  - Bitfield operations were added: BFEXT, BFINS can extract or insert a bitfield
    ▪ Example:
      \[ \text{BFEXT D7}, \text{D6}, \#7:23 \Rightarrow \text{extracts 7 bits #23 from D6 into D7} \]
S12Z – Enhanced Multiplication Instructions

S12(X):

\[
\text{MUL} \quad (A) \times (B) \rightarrow A:B \\
\text{EMUL(S)} \quad (D) \times (Y) \rightarrow Y:D
\]

S12Z:

\[
\text{MULU} \quad (Dj) \times (Dk) \rightarrow Dd \\
(Dj) \times \text{IMM} \rightarrow Dd \\
(Dj) \times (M) \rightarrow Dd \\
(M1) \times (M2) \rightarrow Dd
\]

\[
\text{QMULU} \quad (Dj) \times (Dk) \rightarrow Dd \\
(Dj) \times \text{IMM} \rightarrow Dd \\
(Dj) \times (M) \rightarrow Dd \\
(M1) \times (M2) \rightarrow Dd
\]

\[
\text{MULS} \quad (Dj) \times (Dk) \rightarrow Dd \\
(Dj) \times \text{IMM} \rightarrow Dd \\
(Dj) \times (M) \rightarrow Dd \\
(M1) \times (M2) \rightarrow Dd
\]

\[
\text{QMULS} \quad (Dj) \times (Dk) \rightarrow Dd \\
(Dj) \times \text{IMM} \rightarrow Dd \\
(Dj) \times (M) \rightarrow Dd \\
(M1) \times (M2) \rightarrow Dd
\]

Fractional Math
Fractional Math Support - Instructions

• Fractional Multiply (Signed & Unsigned)
  – Multiplies two signed/unsigned fractional two’s complement operands and stores the signed fractional two’s complement result to register Dd.

• Saturate
  – Saturate the content of the operand register using the information stored in the overflow (V-) and negative (N-) flags by a previous instruction.

• Count-Leading-Bits
  – Counts the number of leading sign-bits in the source register, decrements this number and then copies the result into the destination register.
S12Z Platform Architecture

• Harvard Architecture (Parallel Data and Code access)
# S12Z vs. S12X – Features

<table>
<thead>
<tr>
<th>Attribute</th>
<th>S12Z</th>
<th>S12XE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Harvard</td>
<td>Von Neumann</td>
</tr>
<tr>
<td>Address space</td>
<td>16MByte Linear</td>
<td>64KByte Linear (up to 4MByte Paged)</td>
</tr>
<tr>
<td>Data Bus Width</td>
<td>32-Bit RAM &amp; Flash, 16-Bit I/O</td>
<td>16-Bit</td>
</tr>
<tr>
<td>ALU Width</td>
<td>32-Bit</td>
<td>16-Bit</td>
</tr>
<tr>
<td>Data Registers</td>
<td>2 x 8-Bit, 4 x 16-Bit, 2 x 32-Bit</td>
<td>2 x 8-Bit (can be used as a single 16-Bit accumulator)</td>
</tr>
<tr>
<td>Pointers</td>
<td>2 x 24-Bit</td>
<td>2 x 16-Bit</td>
</tr>
</tbody>
</table>
S12Z vs. S12X – Performance

<table>
<thead>
<tr>
<th>Attribute</th>
<th>S12Z</th>
<th>S12XE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Shifter</td>
<td>32-bit multi-bit 1 cycle</td>
<td>16-bit single-bit 2 cycles</td>
</tr>
<tr>
<td>Multiplier</td>
<td>32*32 2.5 cycles</td>
<td>16*16 1 cycle</td>
</tr>
<tr>
<td></td>
<td>16*16 1 cycle</td>
<td></td>
</tr>
<tr>
<td>Divider</td>
<td>32 = 32/32 18.5 cycles</td>
<td>32 = 32/16 11 cycles</td>
</tr>
<tr>
<td>MAC</td>
<td>32 += 32*32 3.5 cycles</td>
<td>32 += 16*16 13 cycles</td>
</tr>
<tr>
<td>Fractional math</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus speed</td>
<td>50MHz</td>
<td>50MHz</td>
</tr>
</tbody>
</table>

- CPU operates at up to 100MHz, bus at 50 MHz
  - RAM bus can load and store w/ 100MHz
  - I/O buses @50MHz to reduce power consumption and die area
S12Z Benchmarks Results

- S12Z typically saves 20% code size versus S12X
- S12Z typically uses 30% less memory accesses than S12X which saves power

Large Application Code Example:

- Code Size
  - S12X: 250,000 bytes
  - S12Z: 200,000 bytes, -23%

- Memory Access
  - S12X: 25,000 loads, 18,000 stores
  - S12Z: 15,000 loads, 7,000 stores, -42% loads, -57% stores

Digital Filter:
- S12Z is faster and denser than any optimization option of S12X
  - Bytes: S12X CW5i size, S12X CW5i speed, S12Z Cosmic
  - Cycles: S12X CW5i size, S12X CW5i speed, S12Z Cosmic
In Depth Code Size Analysis

<table>
<thead>
<tr>
<th>Feature</th>
<th>S12X</th>
<th>S12Z</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Size</td>
<td>237277</td>
<td>182362</td>
<td>-23%</td>
</tr>
<tr>
<td>pg1</td>
<td>86.3%</td>
<td>96.9%</td>
<td>Shows well chosen pg1 instructions</td>
</tr>
<tr>
<td>Loads</td>
<td>23550</td>
<td>13725</td>
<td>-42%</td>
</tr>
<tr>
<td>Stores</td>
<td>9280</td>
<td>4015</td>
<td>-57%</td>
</tr>
<tr>
<td>Moves</td>
<td>4347</td>
<td>4332</td>
<td>About equal</td>
</tr>
</tbody>
</table>

- Code is about 200k byte large Body Application
  - Significant code size savings
  - Large reduction of memory instructions (not accesses!)
    - => Power Savings expected
Improved Performance for Motor Control

- 100 MHz CPU @ 50 MHz bus speed
- Harvard architecture accelerates data handling
- Fractional math instructions added

Improved Software Friendliness

- 24-bit linear address map to ease software development and porting
- Added 8- and 32-bit registers to allow further compiler code size optimization
Agenda

- Automotive Body Electronics
- MagniV Solutions and Roadmap
- Lab 1
  - Getting Started with CodeWarrior 10.4
- S12Z Enhanced CPU Core
- S12ZVM for BLDC Motor Control
- S12ZVL for General Purpose LIN
- S12ZVC for General Purpose CAN
- Lab 2
  - Developing a Project for the S12ZVL
Overview of S12ZVM Feature Set

Ext Osc | BDM | S12Z core |
---|---|---|
IRC | PLL |

GPIO
- KWU
- RTI
- Wdog
- SPI
- SCI
- SCI
- TIM 4ch/16b
- MSCAN

Dual 12-bit ADC
- 5 + 4ch. Ext. (Mux’d with Op-Amps)
- + 8ch. Int.

PTU | PMF 6-ch PWM | VREG (5V VDD, VLS, VDD sensor)
---|---|---

128 KB Flash | Temp Sense | Charge Pump

8 KB RAM |

512Bytes EEPROM |

Lin Physical Interface |

GDU
- 3-phase / H-Bridge Predriver

Separate 5V VREG (VDDC) |

Current Sense (2 x Op-Amp)

Current Sense (2 x Op-Amp)

Ext Osc | BDM | S12Z core |
---|---|---|
IRC | PLL |

GPIO
- KWU
- RTI
- Wdog
- SPI
- SCI
- SCI
- TIM 4ch/16b
- MSCAN

Dual 12-bit ADC
- 5 + 4ch. Ext. (Mux’d with Op-Amps)
- + 8ch. Int.

PTU | PMF 6-ch PWM | VREG (5V VDD, VLS, VDD sensor)
---|---|---

128 KB Flash | Temp Sense | Charge Pump

8 KB RAM |

512Bytes EEPROM |

Lin Physical Interface |

GDU
- 3-phase / H-Bridge Predriver

Separate 5V VREG (VDDC) |

Current Sense (2 x Op-Amp)

Ext Osc | BDM | S12Z core |
---|---|---|
IRC | PLL |

GPIO
- KWU
- RTI
- Wdog
- SPI
- SCI
- SCI
- TIM 4ch/16b
- MSCAN

Dual 12-bit ADC
- 5 + 4ch. Ext. (Mux’d with Op-Amps)
- + 8ch. Int.

PTU | PMF 6-ch PWM | VREG (5V VDD, VLS, VDD sensor)
---|---|---

128 KB Flash | Temp Sense | Charge Pump

8 KB RAM |

512Bytes EEPROM |

Lin Physical Interface |

GDU
- 3-phase / H-Bridge Predriver

Separate 5V VREG (VDDC) |

Current Sense (2 x Op-Amp)

Ext Osc | BDM | S12Z core |
---|---|---|
IRC | PLL |

GPIO
- KWU
- RTI
- Wdog
- SPI
- SCI
- SCI
- TIM 4ch/16b
- MSCAN

Dual 12-bit ADC
- 5 + 4ch. Ext. (Mux’d with Op-Amps)
- + 8ch. Int.

PTU | PMF 6-ch PWM | VREG (5V VDD, VLS, VDD sensor)
---|---|---

128 KB Flash | Temp Sense | Charge Pump

8 KB RAM |

512Bytes EEPROM |

Lin Physical Interface |

GDU
- 3-phase / H-Bridge Predriver

Separate 5V VREG (VDDC) |

Current Sense (2 x Op-Amp)
Overview of S12ZVM Feature Set

New S12Z CPU
- Up to 100MHz
- 32-bit ALU & 32bit MAC unit,
- Optimized 32-bit math. operations

Safe RAM
- ECC

Flash & EEPROM
- ECC
- Memory Protection
- Margin Read

- S12Z core
- 128 KB Flash
- 8 KB RAM
- 512Bytes EEPROM
Overview of S12ZVM Feature Set

- **Sensor Supply**
  - 20mA I/O for sensor

- **Window Watchdog**
  - Independent RC osc

- **2 UARTs**
  - one linked to LIN Phy
  - 2nd as test interface

- **Multiple timers**
  - IOC/periodic wakeup

- **CAN Option**
  - CAN controller

- **Programmable Trigger Unit**
  - synchronize ADC to PWM
  - Trigger Command list with up to 32 triggers per cycle

- **PWM Module**
  - Complementary mode with deadtime ctrl.
  - Fault protection
  - Double-Switching

- **BDM**
- **S12Z core**
- **KWU**
- **RTI**
- **Wdog**
- **SPI**
- **SCI**
- **SCI**
- **TIM 4ch/16b**
- **KWU**
- **128 KB Flash**
- **8 KB RAM**
- **512Bytes EEPROM**
- **PTU**
- **PMF 6-ch PWM**
- **MCAN**
- **IO/periodic wakeup**
- **Window Watchdog**
  - Independent RC osc
- **CAN Option**
  - CAN controller
**Overview of S12ZVM Feature Set**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ext Osc</td>
<td>BDM</td>
</tr>
<tr>
<td>IRC</td>
<td>PLL</td>
</tr>
<tr>
<td>G P I O</td>
<td>KWU</td>
</tr>
<tr>
<td></td>
<td>RTI</td>
</tr>
<tr>
<td></td>
<td>Wdog</td>
</tr>
<tr>
<td></td>
<td>SPI</td>
</tr>
<tr>
<td></td>
<td>SCI</td>
</tr>
<tr>
<td></td>
<td>SCI</td>
</tr>
<tr>
<td></td>
<td>TIM 4ch/16b</td>
</tr>
<tr>
<td></td>
<td>MSCAN</td>
</tr>
<tr>
<td>Dual 12-bit ADC</td>
<td>5 + 4ch. Ext. (Mux'd with Op-Amps) + 8ch. Int.</td>
</tr>
<tr>
<td></td>
<td>PTU</td>
</tr>
<tr>
<td></td>
<td>PMF 6-ch PWM</td>
</tr>
<tr>
<td></td>
<td>Separate 5V VREG (VDDC)</td>
</tr>
<tr>
<td></td>
<td>Current Sense (2 x Op-Amp)</td>
</tr>
<tr>
<td>Internal RC osc.</td>
<td>• +/-1.3% over tmp</td>
</tr>
<tr>
<td>Two 12-bit ADC</td>
<td>• List Based Architecture allows flexible definition of order and number of conversions • 2.5μs conv. time</td>
</tr>
<tr>
<td>CAN Support</td>
<td>• 5V Vreg controller for external transceiver</td>
</tr>
<tr>
<td>Current Sense Amplifiers</td>
<td>• 2-shunt system supported with additional selectable over-current protection comparator</td>
</tr>
</tbody>
</table>
Overview of S12ZVM Feature Set

- **Voltage Regulator**
  - Boost Option
  - Support for external ballast transistor
  - Vbat sense

- **Charge Pump**
  - Optional to support 100% duty cycle

- **Gate Drive Unit**
  - Operational from 3.5V to 26V
  - Bootstrap circuit based
  - 11V Vreg
  - Phase comparators
  - Desaturation comp. for HS/LS protection
  - Under-/Over-voltage detection
  - DC-link and phase voltage internally accessible on ADC
  - Selectable HS/LS slew rate

- **LIN Physical Interface**
  - 250kb/s fast mode
  - +/-6kV

- **Ext Osc**
  - BDM
  - S12Z core
  - VREG (5V VDD, VLS, VDD sensor)

- **IRC**
  - PLL
  - Temp Sense
  - Charge Pump

- **GPIO**
  - KWU
  - RTI
  - Wdog
  - SPI
  - SCI
  - SPI
  - TIM 4ch/16b
  - MSCAN
  - 128 KB Flash
  - 8 KB RAM
  - GDU
  - 3-phase / H-Bridge Predriver
  - Current Sense (2 x Op-Amp)

- **Ext Osc**
  - 512Bytes EEPROM
  - PTU
  - PMF 6-ch PWM
  - Separate 5V VREG (VDDC)
  - Dual 12-bit ADC

- **Ext Osc**
  - 5 + 4ch. Ext. (Mux’d with Op-Amps) + 8ch. Int.
Overview of S12ZVM Feature Set

New S12Z CPU
- Up to 100MHz
- 32-bit ALU & 32bit MAC unit,
- Optimized 32-bit math. operations

Safe RAM
- ECC
- Margin Read

Flash & EEPROM
- ECC
- Memory Protection
- Margin Read

Voltage Regulator
- Boost Option
- Support for external ballast transistor
- Vbat sense

Charge Pump
- Optional to support 100% duty cycle

Gate Drive Unit
- Operational from 3.5V to 26V
- Bootstrap circuit based
- 11V Vreg
- Phase comparators
- Desaturation comp. for HS/LS protection
- Under-/Over-voltage detection
- DC-link and phase voltage internally accessible on ADC
- Selectable HS/LS slew rate

Internal RC Osc.
- +/-1.3% over temp

Sensor Supply
- 20mA I/O for sensor

Window Watchdog
- independent RC osc

2 UARTs
- one linked to LIN Phy
- 2nd as test interface

Multiple timers
- IOC/periodic wakeup

CAN Option
- CAN controller

Two 12bit ADC
- List Based Architecture allows flexible definition of order and number of conversions
- 2.5µs conv. time

Programmable Trigger Unit
- synchronize ADC to PWM
- Trigger Command list with up to 32 triggers per cycle

PWM Module
- Complementary mode with deadtime ctrl.
- Fault protection
- Double-Switching

LIN Phys. Interface
- 250kb/s fast mode
- +/-6kV

CAN Support
- 5V Vreg controller for external transceiver

Current Sense Amplifiers
- 2-shunt system supported with additional selectable over-current protection comparator

Ext Osc
- IRC
- PLL

S12Z core
- KWU
- RTI
- Wdog

128 KB Flash
- SPI
- SCI
- SCI

8 KB RAM
- TIM 4ch/16b
- 512Bytes EEPROM

GDU
- 3-phase / H-Bridge Predriver

Temp Sense
- VREG
- (5V VDD, VLS, VDD sensor)

Charge Pump
- Separate 5V VREG (VDDC)

Current Sense
- (2x Op-Amp)

Dual 12-bit ADC
- 5 + 4ch. Ext.
- (Mux’d with Op-Amps) + 8ch. Int.

List Based Architecture allows flexible definition of order and number of conversions
- 2.5µs conv. time

Programmable Trigger Unit
- synchronize ADC to PWM
- Trigger Command list with up to 32 triggers per cycle

PWM Module
- Complementary mode with deadtime ctrl.
- Fault protection
- Double-Switching

LIN Phys. Interface
- 250kb/s fast mode
- +/-6kV

CAN Support
- 5V Vreg controller for external transceiver

Current Sense Amplifiers
- 2-shunt system supported with additional selectable over-current protection comparator
S12ZVML Application Schematic

- **S12Z core**: 128 kB Flash
- **Vregs**: (5V VDD, VLS, VDD sensor)
- **Temp Sense**: 8 kB RAM
- **Charge Pump**: 512 bytes EEPROM
- **GDU**: 3 phase H-Bridge Predriver
- **Current Sense**: (2 x Op-Amp)
- **Dual 12bit ADC**: 5+4ch. Ext. (Mux’d with Op-Amps) + 8ch. Int.
- **PMF**: 6-ch PWM

**Optional Components**:

- **Ext Osc BDM IRC PLL PWFSUP PWFBST**: Dual 12bit ADC
- **IO/ MISO**: 5+4ch. Ext. (Mux’d with Op-Amps) + 8ch. Int.
- **IO/ MOSI**: Dual 12bit ADC
- **IO/ SCLK**: Dual 12bit ADC
- **IO/ SS**: Dual 12bit ADC
- **IO/ RXD0**: Dual 12bit ADC
- **IO/ TXD0**: Dual 12bit ADC
- **IO/ KWP0**: Dual 12bit ADC
- **IO/ KWP1**: Dual 12bit ADC
- **IO/ IOC0**: Dual 12bit ADC
- **IO/ IOC1**: Dual 12bit ADC
- **IO/ IOC2**: Dual 12bit ADC
- **IO/ IOC3**: Dual 12bit ADC

- **Hallout**: Dual 12bit ADC
- **AMRcos AMRsin**: Dual 12bit ADC

**Legend**:

- **optional**
Space Savings with S12ZVML

Full Discrete

- VREG (8pin)
- LIN phy (8pin)
- MCU or DSC (48pin)
- Gate Driver (48pin)
- Op-amps

Semi Integrated

- MCU or DSC (48pin)
- ASIC (GDU+ VREG+ LIN+ Op-amps) (64pin)

S12ZVML
64pin
10x10mm

6 to 8 cm² PCB space savings
3 to 5 cm² PCB space savings
SMTA (Surface Mount Technology Association):
• NMACPCs (Non-Material Assembly Cost Per Component) range from $0.01 to $0.15 per component assembled
• NMACPI/O (Non-Material Assembly Cost Per Input/Output) range from $0.005 to $0.01 per I/O assembled.
Highest level of Integration

Of your BLDC system:
Agenda

- Automotive Body Electronics
- MagniV Solutions and Roadmap
- Lab 1
  - Getting Started with CodeWarrior 10.4
- S12Z Enhanced CPU Core
- S12ZVM for BLDC Motor Control
- S12ZVL for General Purpose LIN
- S12ZVC for General Purpose CAN
- Lab 2
  - Developing a Project for the S12ZVL
Ultimate Flexibility and Scalability

**Connectivity**
- CAN or LIN support with on-chip physical interface

**On-chip 12 V voltage regulator**
- For flexible on- and off-chip supply

**Flash memory size (KB)**
- 8 KB
- 64 KB
- 192 KB

**Range of memory options**
- Address multiple applications and allow for platform design

*True platform solutions through hardware and software reuse*
## Overview of S12ZVL Feature Set

<table>
<thead>
<tr>
<th>LIN Physical Interface</th>
<th>Pierce Oscillator</th>
<th>Temp Sense</th>
<th>10-bit ADC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCI</td>
<td>RC osc ±1.3%</td>
<td>PLL</td>
<td>16-bit Timer 6-ch + 2-ch</td>
</tr>
<tr>
<td>SCI</td>
<td>S12Z Core</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPI</td>
<td>32k Byte Flash (ECC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2C</td>
<td>128 Byte EEPROM (ECC)</td>
<td>1k Byte RAM (ECC)</td>
<td>BDM</td>
</tr>
<tr>
<td>GPIO</td>
<td>EVDD × 1</td>
<td>HVI × 1</td>
<td>V-SUP Sense</td>
</tr>
<tr>
<td></td>
<td>NGPIO × 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **VREG for Total Supply:**
  - 70mA
  - 170mA with ext ballast
Overview of S12ZVL Feature Set

New S12Z CPU
- up to 50MHz
- 32-bit ALU & 32-bit MAC unit
- Optimized for 32-bit math operations

Flash & EEPROM
- with ECC
- Memory Protection
- Margin Read

Safe RAM
- with ECC

S12Z Core

32k Byte Flash (ECC)

128 Byte EEPROM (ECC)

1k Byte RAM (ECC)
# Overview of S12ZVL Feature Set

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCI</td>
<td>16-bit Timer 6-ch + 2-ch</td>
</tr>
<tr>
<td>SCI</td>
<td>PWM 8-ch, 8-bit (or 4-ch, 16-bit)</td>
</tr>
<tr>
<td>S12Z Core</td>
<td>32k Byte Flash (ECC)</td>
</tr>
<tr>
<td>SPI</td>
<td>128 Byte EEPROM (ECC)</td>
</tr>
<tr>
<td>I2C</td>
<td>1k Byte RAM (ECC)</td>
</tr>
<tr>
<td>GPIO × 1</td>
<td>3 × 25mA Output Sink • GPIO and/or PWM control</td>
</tr>
<tr>
<td>GPIO × 3</td>
<td>1 × 20mA Output Source • GPIO and/or PWM control</td>
</tr>
<tr>
<td>EVDD</td>
<td>66 Byte Flash (ECC)</td>
</tr>
<tr>
<td>NGPIO × 3</td>
<td>1k Byte RAM (ECC)</td>
</tr>
<tr>
<td>BDM</td>
<td>66 Byte Flash (ECC)</td>
</tr>
</tbody>
</table>

**Notes:**
- 3 × 25mA Output Sink
- 1 × 20mA Output Source
- GPIO and/or PWM control
### Overview of S12ZVL Feature Set

<table>
<thead>
<tr>
<th>SCI</th>
<th>Pierce Oscillator</th>
<th>Temp Sense</th>
<th>10-bit ADC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCI</td>
<td>Internal RC oscillator</td>
<td>±1.3%</td>
<td>16-bit Timer 6-ch + 2-ch</td>
</tr>
<tr>
<td>SPI</td>
<td>S12Z Core</td>
<td>PWM 8-ch, 8-bit (or 4-ch, 16-bit)</td>
<td></td>
</tr>
<tr>
<td>I2C</td>
<td>32k Byte Flash (ECC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPIO</td>
<td>128 Byte EEPROM (ECC)</td>
<td>1k Byte RAM (ECC)</td>
<td>BDM</td>
</tr>
<tr>
<td>EVDD × 1</td>
<td>NGPIO × 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Internal RC oscillator**
  - ±1.3% over temp
### Overview of S12ZVL Feature Set

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
</table>
| **LIN Physical Interface**    | • 250kbps Fast Mode
• ±6kV (IEC61000-4-2)                                                              |
| **1 × High Voltage Input**    | • with Interrupt capability                                                   |

<table>
<thead>
<tr>
<th>Interface</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCI</td>
<td>Pierce Oscillator</td>
<td>RC osc ±1.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Temp Sense</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10-bit ADC</td>
</tr>
</tbody>
</table>
| SCI       | S12Z Core        | 16-bit Timer
6-ch + 2-ch                                                                |
| SPI       | 32k Byte Flash   | 8-bit (or 4-ch, 16-bit)                                                     |
|           | (ECC)            | PWM 8-ch, 8-bit                                                             |
| I2C       | 128 Byte EEPROM  | 1k Byte RAM                                                                 |
|           | (ECC)            | VREG for Total Supply:                                                     |
|           |                  | • 70mA                                                                     |
|           |                  | • 170mA with ext ballast                                                   |
| GPIO      | EVDD × 1         | HVI × 1                                                                     |
|           | NGPIO × 3        | V-SUP Sense                                                                 |
|           |                  | BDM                                                                        |
|           |                  | Voltage Regulator                                                          |
|           |                  | • 5.5V to 18V normal operating range                                        |
|           |                  | • protected operation up to 40V                                            |
|           |                  | • 3.5V minimum operating voltage                                           |
|           |                  | • support for external ballast transistor                                  |
### Functional Safety – ISO 26262 ASIL A Compliance:
- Developed in accordance with ISO 26262 as a Safety Element out of Context (SEooC).
- Usable in safety relevant systems classified as ISO 26262 ASIL A.
- MCU development process fulfills ASIL A requirements of ISO 26262.
S12ZVL Key Features

**RGB ambient lighting**

**Sensor (airflow)**

**Switchpanels / Userinterfaces**

### Key Feature

<table>
<thead>
<tr>
<th>S12Z core, 25 MHz Bus</th>
<th>Improved code-efficiency &amp; core performance vs. S12</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 KB to 32 KB flash</td>
<td>Family-concept, no need for external Flash/EPROM/ROM</td>
</tr>
<tr>
<td>Up to 128 Bytes EEPROM</td>
<td>4B erasable page; Easier to use vs. data flash; no need for EE-emul.</td>
</tr>
<tr>
<td>all memories (Flash, RAM, EE) with ECC</td>
<td>Error Code Correction provides high reliability</td>
</tr>
<tr>
<td>Built-in automotive voltage regulator operating between 3.5 and 40 V</td>
<td>Operates directly from car-battery, no need for extra Voltage-regulator saving PCB-boardspace. handles automotive design issues, such as double battery, crank voltage and load dump conditions</td>
</tr>
<tr>
<td>Built-in LIN physical layer</td>
<td>No need for an external LIN physical layer device, saving space and design time. Meets automotive OEM specifications for LIN conformance and EMC requirements</td>
</tr>
<tr>
<td>NGPIO &amp; EVDD</td>
<td>EVDD able to supply 5V/20 mA offchip; N-GPIO able to sink up to 3 x 25mA off Chip (useful for RGB-LED-drive)</td>
</tr>
<tr>
<td>Protected 12V input (HVI)</td>
<td>Allows automotive battery voltage-level inputs (with ADC-capability &amp; ESD-protection)</td>
</tr>
<tr>
<td>on Chip RC oscillator trimmed to 1,3% tolerance</td>
<td>Due to accurate on chip clock generation LIN-communication can be done without external Crystal or Resonator and without the need of SW-intense synchronization</td>
</tr>
</tbody>
</table>

### Benefits

- **SCI**
  - RC osc ±1.3%
  - Pierce Oscillator
  - PLL
  - 16-bit Timer 6-ch + 2-ch

- **SCI**
  - S12Z Core
  - PWM 8-ch, 8-bit (or 4-ch, 16-bit)

- **SPI**
  - 32k Byte Flash (ECC)

- **I2C**
  - 128 Byte EEPROM (ECC)
  - 1k Byte RAM (ECC)

- **GPIO**
  - EVDD × 1
  - NGPIO × 3
  - HVI × 1
  - V-SUP Sense

- **Temp Sense**

- **10-bit ADC**

- **LIN Physical Interface**

- **PLL**

- **Temp Sense**

- **10-bit ADC**

### Package Options:

- 48 LQFP, 32 LQFP
- 32QFN (5 x 5 mm)
# S12ZVC Key Features

![Seatbelt pretention](image)

- **CAN Physical Interface**
  - High Speed – up to 1Mbps
  - ISO 11898-2 & ISO 11898-5

- **High-Resolution Timer**
  - 20ns with 25MHz bus

- **High-Resolution PWM**
  - 20ns with 25MHz bus

- **Separate CAN Supply**
  - with external ballast transistor

---

## Key Feature | Benefits
--- | ---
S12Z core, 32 MHz Bus | Improved code-efficiency & core performance versus S12
64 KB to 192 KB flash | Family-concept, no need for external Flash/EPROM-ROM
Up to 2k Bytes EEPROM | 4B erasable page; Easier to use vs. data flash; no need for EE-emul.
all memories (Flash, RAM, EE) with ECC | Error Code Correction provides high reliability
Built-in automotive voltage regulator operating between 3.5 and 40 V | Operates directly from car battery without the need for extra voltage regulator, saving PCB board space. Handles automotive design issues, such as double battery, crank voltage and load dump conditions
Built-in CAN physical layer | No need for an external CAN physical layer device, saving space as well as time & cost for design & testing. Meets automotive OEM requirements for CAN conformance and EMC.
Analog Comparator with DAC | Allows for fast reaction of ext. signals vs. preprogrammed threshold
Protected 12V input (HVI) | Allows automotive battery voltage-level inputs (with ADC-capability & ESD-protection)
12-Bit ADC; 16ns-Timer/PWM | High resolution mixed signal peripherals for many sensor-applications (e.g. ultrasonic)

## Package Options:
- 64 LQFP
- 48 LQFP
S12ZVL Application Example: Switch Panel

Automotive Voltage

- Voltage Regulator
- LIN PHY
  - LIN BUS
  - S12Z
  - GPIO
  - PWM
  - ADC 10-Bit
- HVI
- 5V Switch Bank
- Wakeup
- Backlight
- Dials
- 12V Switch

Digital Components
5V Analogue Components
MCU Core and Memories
High-Voltage Components
S12ZVL Application Example: Intelligent Sensor

Automotive Voltage

- Voltage Regulator
- LIN PHY
- SPI
- S12Z
- GPIO
- 5V Supply
- Rain / Light Sensor ASIC

LIN BUS

Digital Components
5V Analogue Components
MCU Core and Memories
High-Voltage Components

S12ZVL Application Example: LIN RGB LED
S12ZVL Application Example: LIN RGB LED
(with autoaddressing / daisychaining)
TRK-S12ZVL: LIN Daisychain
S12VL: S12 MagniV Mixed-Signal MCU for LIN Applications

Overview

Features:
- Enhanced S12Z core at 26 MHz bus speed
- Up to 32 KB Flash (with ECC)
- 1024 EEPROM (with ECC)
- Up to 1 KB RAM (with ECC)
- Yseg for 5.5 to 22-Volt operating range, scalable in supply for 3V and off-chip systems
- LIN physical layer
- On-chip RC oscillator 1.3% accurate

Related Products:
- S12VR/S12D MagniV Mixed-Signal MCU for Relay Driver
- S12ZUC/S12Z MagniV Mixed-Signal MCU for CAN Applications
- S12ZVI/S12VL MagniV Mixed-Signal Microcontroller for Automotive Instrument Cluster Applications

Target Applications:
- Automotive
  - Heating, Ventilation and Air Conditioning (HVAC)
  - Lighting
  - Doors, Window Lift and Seat Control
  - Steering Wheel Controllers
  - Watchdog Controller for Chassis/Safety/Powertrain
  - LIN Nodes
  - LIN User Interface
  - LIN Switch Panel
  - LIN Actuators, Sensors
- Industrial
  - Ambient Lighting Control

This page contains information on a preproduction product. Specifications and information herein are subject to change without notice.

Featured Documentation:
- S12ZVLS: S12 MagniV S12VL Family - First Steel
- S12ZVI: S12 MagniV LIN-Ready, LIN-Compliant 8-Bit 12-Bit 16-Bit Microcontroller - White Paper

freescale.com/MagniV
Freescale LIN 2.1 Driver (version 4.4.1)
Agenda

- Automotive Body Electronics
- MagniV Solutions and Roadmap
- Lab 1
  - Getting Started with CodeWarrior 10.4
- S12Z Enhanced CPU Core
- S12ZVM for BLDC Motor Control
- S12ZVL for General Purpose LIN
- S12ZVC for General Purpose CAN
- Lab 2
  - Developing a Project for the S12ZVL
Lab Part 2: PWM LED Control

① Double-Left-Mouse-Click on the Processor Expert “Component Library” tab
Add a PWM Component to the Project

② Expand “CPU Internal Peripherals”, then “Timer”

③ Right-Mouse-Click Select “PWM” then “Add to project”

④ Double-Left-Mouse-Click on the Processor Expert “Component Library” tab
Change PWM Component Properties

⑤ Double-Left-Mouse-Click on "*Component Inspector - PWM8" tab
7. Change the Component name to “PWM_RED” and select the PWM channel “PWMPER1”

6. Select “Expert”

8. Select anywhere in the Value column beside “Period” and then select the ellipsis (“…”).
Enter “4” ms and then select “OK”
10. Repeat with the “Starting pulse width” by making it “0”ms.

11. Select the “Methods” tab
Configure PWM Component Properties

⑫ Select “generate code” for “SetRatio8”

⑬ Double-Left-Mouse-Click on “*Component Inspector – PWM_RED” tab
Create a PWM Component for Each LED Colour

• Repeat the previous steps ① through ⑬ two more times creating separate identical PWM channels for:

  – “PWM_GREEN” on “PWMPER5”
    ▪ on Output pin “PP5_XIRQ_KWP5_PWM5”

  – “PWM_BLUE” on “PWMPER3”
    ▪ on Output pin “PP3_IRQ_KWP3_PWM3”
Add a Periodic Timer Component to the Project

Select “*Component Inspector – TI1” tab

Right-Mouse-Click Select “TimerInt” then “Add to project”
Change the Component name to “Periodic”

Change the Interrupt period to “5”ms

Double-Left-Mouse-Click on “*Component Inspector – Periodic” tab
Build the Processor Expert Components

- Build ‘FLASH’ for project ‘Lab’
- Find and Double-Left-Mouse-Click on “ProcessorExpert.c”
- Double-Left-Mouse-Click on “ProcessorExpert.c” tab
Add Code to ProcessorExpert.c

```c
/* User includes (#include below this line is not maintained by Processor Expert) */

volatile uint8_t tick;
static uint8_t tick_last;

void demo (void)
{
    static uint8_t state = 5;
    static uint8_t duty_RED = 0x00;
    static uint8_t duty_GREEN = 0xFF;
    static uint8_t duty_BLUE = 0x00;

    if (tick == 0)
    {
        state++;
        if (state == 6)
        {
            state = 0;
        }
    }

    switch (state)
    {
    case 0: // go to Red
duty_BLUE++;
        break;
    case 1: // go to Yellow
duty_GREEN--;
        break;
    case 2: // go to Green
duty_RED++;
        break;
    case 3: // go to Aqua
duty_BLUE--;
        break;
    }
}
```

- Global variables
- Local variables
- “state” control
Add Code to ProcessorExpert.c

```c
if (state == 0)
{
    state = 0;
}

switch (state)
{
    case 0: // go to Red
        duty_BLUE++;
        break;
    case 1: // go to Yellow
        duty_GREEN--;
        break;
    case 2: // go to Green
        duty_RED++;
        break;
    case 3: // go to Aqua
        duty_BLUE--;
        break;
    case 4: // go to Blue
        duty_GREEN++;
        break;
    case 5: // go to Purple
        duty_RED--;
        break;
    default:
        break;
}

PMM_RED_SetRatio8(duty_RED);
PMM_GREEN_SetRatio8(duty_GREEN);
PMM_BLUE_SetRatio8(duty_BLUE);

void main(void)
```

"state" case switch

LED control
Add Code to ProcessorExpert.c

Double-Left-Mouse-Click on “ProcessorExpert.c” tab

Simple interrupt-based delay
Add Code to Events.c

Find and Double-Left-Mouse-Click on “Events.c”

Double-Left-Mouse-Click on “Events.c” tab
Add Code to Events.c

```c
#include "Cpu.h"
#include "Events.h"

/** User includes (#include below this line is not maintained by Processor Expert) */
extern volatile uint8_t tick;

#pragma CODE_SEG DEFAULT

/** Event : Periodic_OnInterrupt (module Events) */
/** Component : Periodic [TimerInt] */
/** Description : When a timer interrupt occurs this event is called (only when the component is enabled - <Enable> and the events are enabled - <EnableEvents>). This event is enabled only if a <interrupt service/event> is enabled. */
/** Parameters : None */
/** Returns : Nothing */

void Periodic_OnInterrupt(void)
{
  /* Write your code here ... */
  tick++;
}

/* END Events */
```

Global variable

Increment global variable
Prepare the Hardware

• Connect USB cable to the TRK-S12ZVL Evaluation Board

• Connect 12V Power Supply
Build and Prepare to Debug

Build ‘FLASH’ for project ‘Lab’

Select Debug Pull-down Menu and then “Debug Configurations…”
Select the Debug Configuration

Select “CodeWarrior Download then “Lab_FLASH_Open Source BDM”

Select “Debug”
Agenda

- Automotive Body Electronics
- MagniV Solutions and Roadmap
  - Lab 1
    - Getting Started with CodeWarrior 10.4

- S12Z Enhanced CPU Core
- S12ZVM for BLDC Motor Control
- S12ZVL for General Purpose LIN
- S12ZVC for General Purpose CAN
  - Lab 2
  - Developing a Project for the S12ZVL