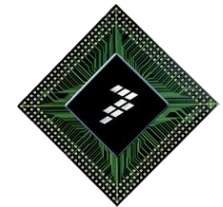


June, 2010

Hands-on Workshop: Multicore Initiation: System Initialization for the MPC5643L

FTF-AUT-F0403



Gene Fortanely
Applications Engineer

► This workshop will cover:

- A hands-on, direct experience with the *MPC5643L EVB*, an evaluation kit
- The start-up and initialization process for the *MPC5643L*
- Lock Step Mode (LSM) and Decoupled Parallel Mode (DPM)
- How to best use the SW/HW debug tools included in the EVB system kit

- **Presenter:** Gene Fortanelly
- **Expertise:** Embedded Software Development

- This hands-on workshop is slated to take up to 4 hours

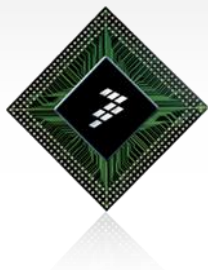
► At the end of this session, you will be able to:

- Apply the development tools for the *MPC5643L* in your application
- Bring up the MPC5643L in LSM and DPM modes.
- Load and run a simple executable in LSM and DPM modes.

Required Tools (Provided)

1. ASD *MPC5643L* Mini-Module (daughtercard)
2. P&E *xPC56XXMB* (motherboard)
3. P&E *USB Multilink* (JTAG interface device)
4. Laptop pre-loaded with *CodeWarrior* software development tools, and P&E software debugger and Flash programmer.
5. *MPC5643L* DPM and LSM source code, linker files, and pre-built images
6. Documentation

- ▶ Items are provided for the duration of the workshop.
- ▶ Further P&E information available at www.pemicro.com



Agenda

- ▶ **2:00-2:15 pm Introductions**
- ▶ **2:15-2:30 pm Introduction to the Freescale MPC5643L**
- ▶ **2:30-2:45 pm Lock Step Mode (LSM) and Decoupled Parallel Mode (DPM)**
- ▶ **2:45-3:00 pm Introduction to the MPC5643L EVB Kit**
- ▶ **3:00-3:15 pm Programming the Shadow Flash**
- ▶ **3:15-3:30 pm CodeWarrior Development Tools support for LSM/DPM**
- ▶ **3:30-4:15 pm Doing LSM with CodeWarrior** (Hands-on programming the shadow flash, building, loading, and executing the LSM source code)
- ▶ **4:15-5:00 pm Doing DPM with CodeWarrior** (Hands-on programming the shadow flash, building, loading, and executing the DPM source code)
- ▶ **5:00-6:00 pm Review and questions**

- ▶ Gene Fortanely, Freescale Applications Engineer

Gene.Fortanely@Freescale.com

Telephone: (512) 996-5551

- ▶ Engineers attending the workshop – including any learning goals they have for the workshop.

Introduction to the Freescale *MPC5643L* 32-bit MCUs for Safety Critical Applications

Electric power steering

Vehicle dynamic and
chassis control



Front & rear radar

Blindspot detection

Pre-crash detection

Electronic stability program

Hybrid electric vehicles

Safety Application Challenges

► Increasing integration and system complexity

- Increasing electrification of the vehicle (replacing traditional mechanical systems)
- Subsequent increase in use of high-performance sensor systems has driven increased MCU performance needs

► Functional Safety & mounting cost pressures

- Automotive system manufacturers need to guarantee IEC61508 (SIL3) and ISO26262 (ASIL-D) system safety capability
- Applications typically need to duplicate system and/or have additional system level components in order for this to be achieved
- Move from passive to active safety is increasing the number of safety functions distributed in many ECUs (Electronic Control Unit)
- Costly and results in long and complex development cycles per application
- Mounting costs pressure leading to integration of more functionality in a single ECU

► Continuing demand for quality

- Zero defects, 10x quality improvement expected

MPC564xL MCU Family Addresses Market Challenges

► Offers exceptional integration and performance

- *MPC5643L* provides a cost effective solution by reducing board size, chip count and logistics/support costs
- Unique dual mode of operation allows customers to choose how best to address their safety requirements without compromising on performance

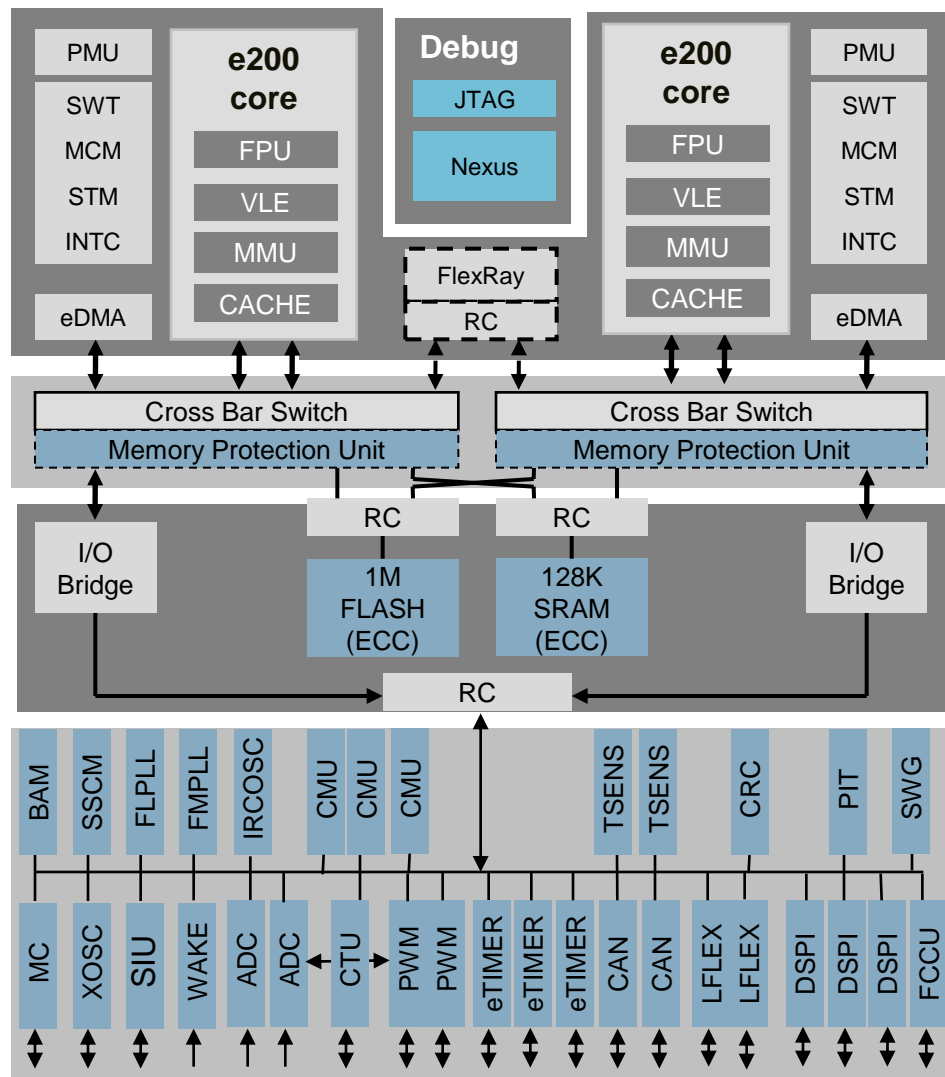
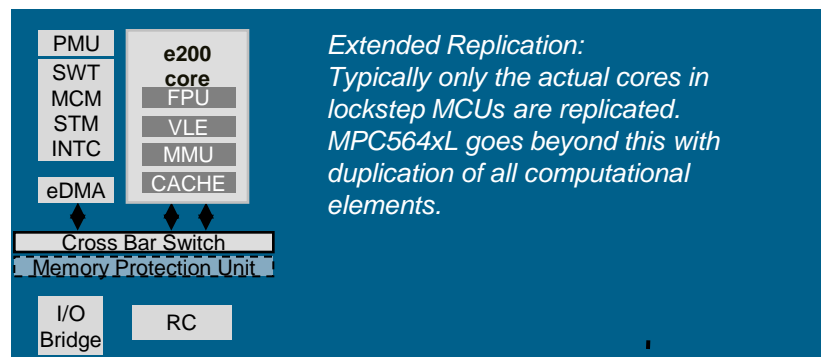
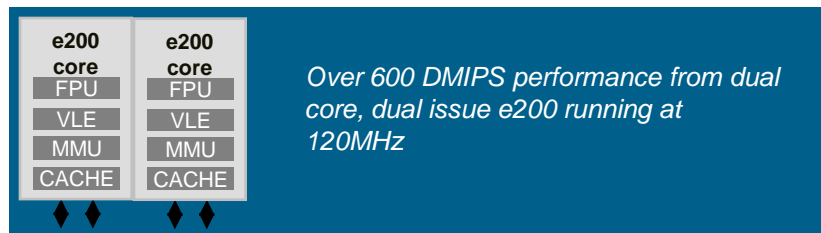
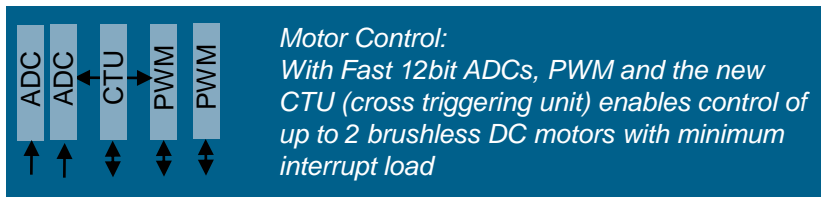
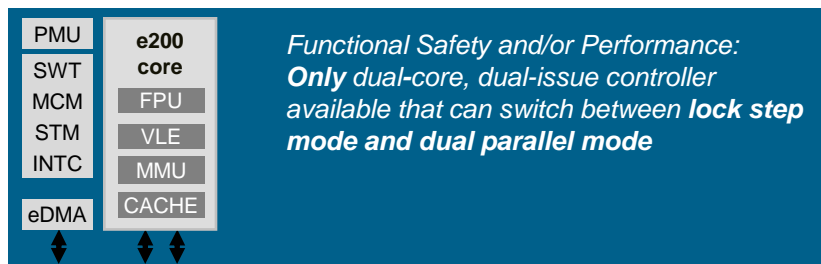
► Solves Functional Safety

- The *MPC5643L* Functional Safety architecture has been specifically designed to support IEC61508 (SIL3) and ISO26262 (ASIL-D) safety standards.
- *MPC5643L* architecture provides redundancy checking of all computational elements in order to help ensure the operation of safety-related tasks

► Offers best-in-class quality

- Zero defects are integral in a successful “design for quality” initiative
- Test and manufacture are aligned to lifetime warranty needs

MPC5643L: 1 MB Safety & Chassis Controller

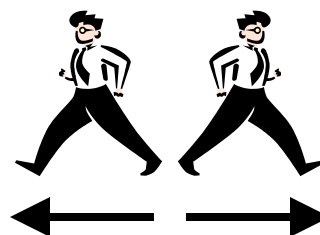


Dual Core – Modes of Operation



LSM
LockStep Mode

- ▶ 1x performance
- ▶ MCU mode which allows SIL3/ASIL-D with minimal software overhead
- ▶ Formal check of outputs for replicated IP modules
- ▶ Checker (RC) guarantee detection of non-common cause faults when redundant channels are merged



DPM
Decoupled Parallel Mode

- ▶ Up to 2x performance
- ▶ MCU mode which allows SIL3/ASIL-D with algorithm diversity
- ▶ CPU cores and subsystems run independently
- ▶ Checker units (RC) are disabled in this mode

- Symmetric dual-core device
- Device concept supports static configuration of the operating mode during Reset
- Selection between the two modes is done by programming a user option bit stored in the shadow block of the flash array

MPC5643L (Leopard) – Key Benefits Summary

► Higher Performance

- Up to **25% more performance thru Dual Issue** - e200z4 **dual issue** core architecture provides 2.31 DMIPS/MHz intrinsic performance
- **SIMD & FPU unit** - provides DSP (Digital Signal Processing) capabilities
- Small instruction cache - boosts performance for localized motor control code

► Peripherals for complex motor control

- **Cross Triggering Unit** – coordinates ADC (Analog-to-Digital Converter), Timer and PWM (Pulse Width Modulations) generation and minimizes CPU interrupt load
- **High precision A/D conversion** – 12bit resolution ADC with TUE +/-2 LSB

► Turn key solution for SIL3 & ASIL-D systems

- **Two modes of operation** - Decoupled Parallel Mode (DPM) & statically configurable Lockstep Mode (LSM)
- **Sphere of Replication (SoR)** – Offering on-chip redundancy, reducing need for additional ECU and SW components by duplicating critical MCU elements (CPU, DMA, INTC, XBAR, MPU, etc.)
- **Fault Collection and Control Unit** - offers a systematic approach to fault detection, collection and control

► Two relevant safety standards

- IEC 61508 (in revision)
 - Generic standard for functional safety of electronic systems
 - Specialized versions for individual industries
- ISO 26262 (in preparation)
 - 'Derivate' of IEC 61508 for automotive applications
 - Already in used although not complete

► Goal

- Prevent unacceptable risk due to failures of equipment

► Approach

- Reduction of Systematic faults
- Human-introduced 'bugs'
- Constraints on development process
- Reduction of Random faults
- Failures due to aging, interference, ...
- Quantitative requirements via Safety metrics

Note: The MCU is not certified
it's the ECU/system

MPC564xL Family

Built on Power Architecture® Technology

► Simplifies Design

- A flexible, configurable architecture addresses both lockstep and dual parallel operation modes on a single dual core chip and helps simplify design

► Safety Standards Compliance

- A redundant architecture provides a compelling solution for real-time applications that require compliance with the IEC61508 SIL3 & ISO26262 ASIL-D safety standards

► Lower Systems Cost

- Dual core architecture reduces the need for component duplication at the system level, and lowers overall system costs



Lock Step Mode (LSM) and Decoupled Parallel Mode (DPM)

- ▶ In LSM the two cores of the MCU operate in lockstep.
 - Any deviation in the output of the two cores is detected by hardware and signaled as a possible failure.
 - The LSM provides a high diagnostic coverage and short detection intervals on hardware level without software interaction. This capability comes at the cost of a higher computational overhead since the two cores provide the performance of only one.
- ▶ In DPM the two channels of the MCU work independently and redundancy checkers (RCCU) are disabled.
 - No automatic hardware check for equal operation between them is executed although some master/checker combinations orthogonal to the channels will continue to check for correct hardware operation.

- ▶ The choice between operating in LSM and DPM is often driven by the tradeoff between performance and SW transparency.
- ▶ In some automotive systems it is desirable to have the higher performance of the DPM for non-safety-related functions.

LSM and DPM (continued)

- ▶ The decision of which mode the chip runs (LSM and DPM) is determined by the LSM_DPM bit, which is in the BIU4 default in the shadow block of the flash.
- ▶ If this bit is 0, then the unit starts up in DPM.
- ▶ If this bit is 1, then the unit starts up in LSM.
- ▶ The shadow flash sector is located at 0x00F0_0000.
- ▶ The BIU4 default is located at 0x00f0_3e10. Also mirrored at 0x00ff_fe10 (+0x000f_c000) for the convenience of some software tools.

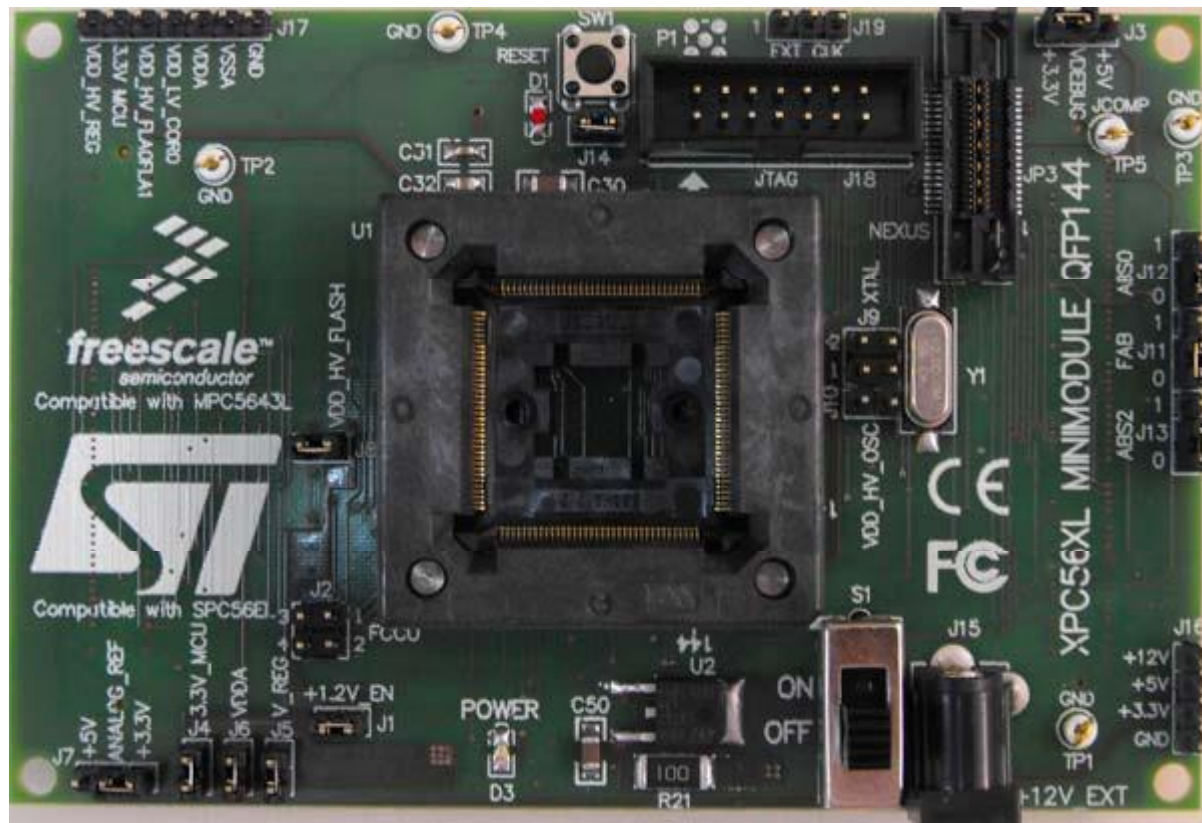
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SWT									LSM/ DPM	STCU_ CFG		Reserved			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reserved				FCCU_CFG											

Introduction to the *MPC5643L* EVB Kit

- ▶ The *xPC5643L EVB* kit is an evaluation system supporting Freescale *MPC5643L* microprocessors.
- ▶ The *xPC5643L EVB* kit consists of:
 - *xPC56XXMB* Motherboard from P&E Microsystems
 - *xPC56xLADPT144s* (LQFP-144) or *xPC56xLADPT257s* (BGA-257) Mini-Module from ASD which plugs into the motherboard.
 - *CodeWarrior MPC55xx/MPC56xx* v2.6 software development tools
 - P&E debugger
 - P&E USB multilink

Introduction to the *MPC5643L* EVB (continued)

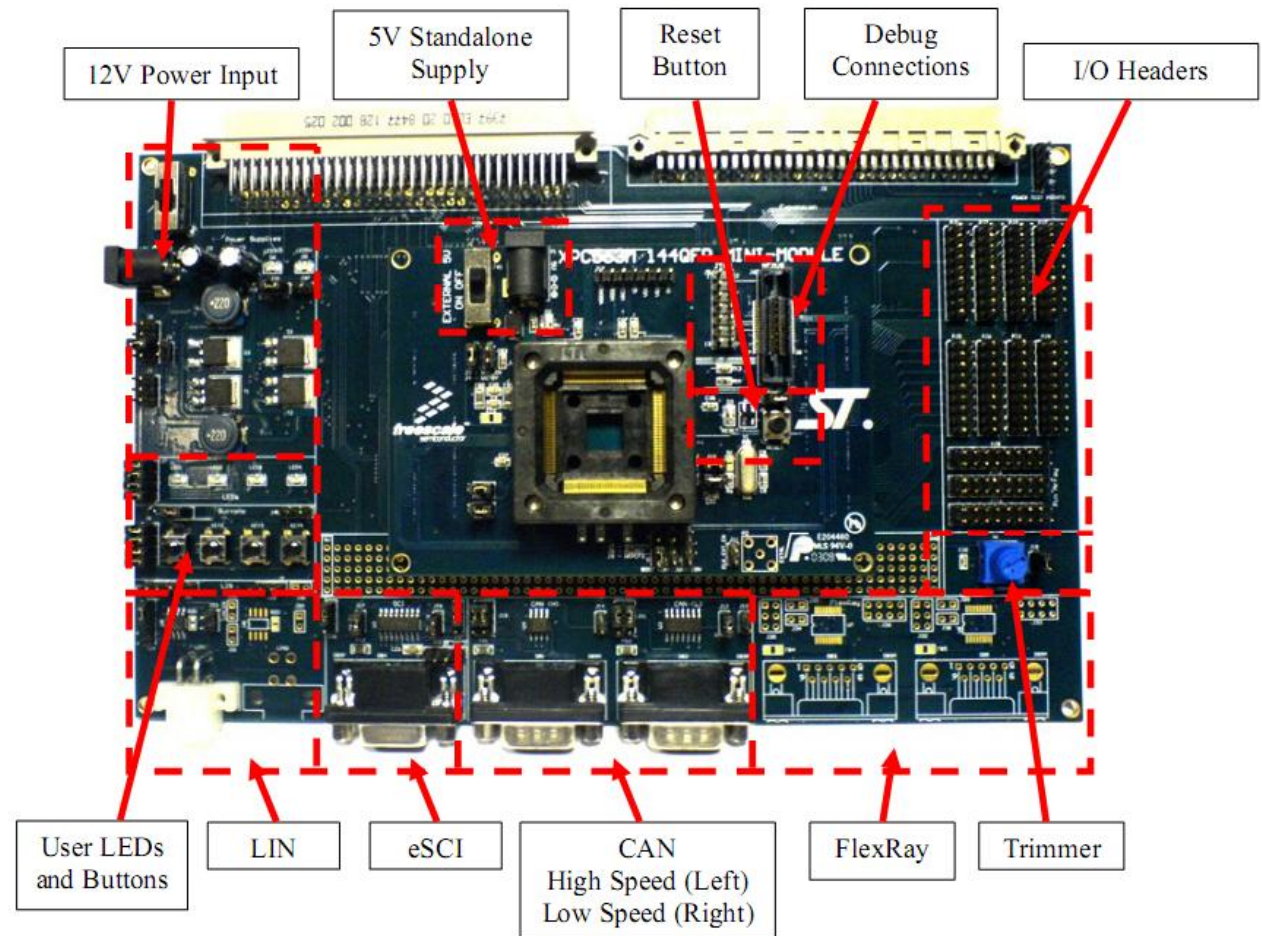
- ▶ The Mini-Module may be used as a stand-alone unit, which allows access to the CPU, but allows no access to the I/O pins or any motherboards peripherals.



Introduction to the *MPC5643L* EVB (continued)

► The Motherboard, when used with the Mini-Module enables:

- full access to the CPU
- all of the CPU's I/O signals
- motherboard peripherals



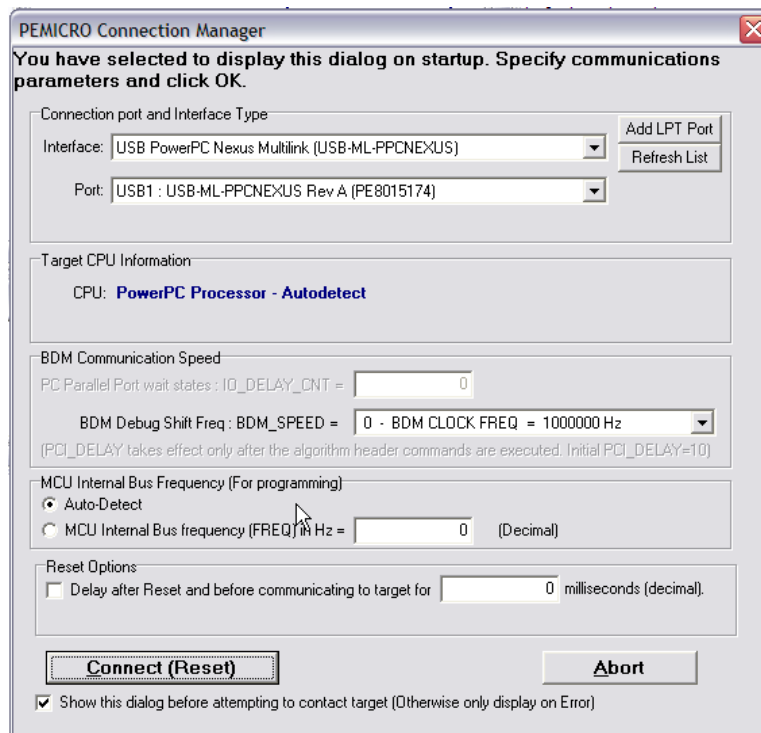
Programming the Shadow Flash

- ▶ These are directions on how to program the contents of the Shadow Flash on the MPC5647L for the purpose of switching between DPM and LSM.
- ▶ For this you will need the following P&E items (part of kit):
 - CW_ICDPPCNEXUS.EXE v1.23.3.2 or newer
 - cw_progppcnexus.exe v1.30.1.1 or newer
 - USB multilink
- ▶ Procedure:
 - Configure the Flash programming utility
 - Dump the shadow flash to an s-record file
 - Modify s-record to change the LSM/DPM configuration
 - Program the shadow flash with the updated configuration
 - Verify changed configuration

Programming the Shadow Flash (continued)

– Configure the Flash Programming Utility

- ▶ Run the Flash programming Utility: *cw_progppcnexus.exe*
- ▶ Ensure the MCU is powered and the Multilink connected
- ▶ Press the Connect (Reset) button to connect to the *MPC5643L*



Programming the Shadow Flash (continued) – Configure the Flash Programming Utility

- ▶ You will now be prompted to Specify the programming Algorithm to use. P&E provides a special Algorithm for programming the shadow flash of the MCU. In this case we must select the file:

Freescale_MPC5643L_1x32x4k_Shadow_Blк_Freescale_C90FL2_Driver_031.PCP

- ▶ This file is located in:

**C:\Program Files\Freescale\CW for MPC55xx and MPC56xx
2.6\pemicro\algorithms\shadow**

if you follow the default CodeWarrior installation path.

Programming the Shadow Flash (continued) – Configure the Flash Programming Utility

- ▶ This file is located in:

pemicro\pkgppcnexus_starter\Algorithms\shadow

if you have used the default P&E installation path.

- ▶ The tool is now configured to work with the *MPC5643L* Shadow Flash.

Programming the Shadow Flash (continued)

– Dump the Shadow Flash to an S-record File

- ▶ Once the tool has been configured for the *MPC5643L* shadow flash, you can dump the existing contents of the shadow flash to a s-record file. (Contents will be the factory default if the shadow row has never been programmed.)
- ▶ Press the upload module button on the console to do this.
- ▶ You will be prompted to provide a location and name for the file.

Programming the Shadow Flash (continued)

- ▶ The default s-record can be opened and edited in a text editor. For this example the default has been taken and the following line changed to update from Lock Step mode with watchdog enabled to Dual Core with Watchdog disabled:
- ▶ Original Line:
 - S214FFFE10FFFFFFFFFFFFFFFFFFFFFFFFFFFFFEE
- ▶ Updated Line:
 - S214FFFE107FBFFFFFFFFFFFFFFFFFFFFFFFFFAE
- ▶ Note: S2 xx yyyyyy zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz cs
- ▶ Note that the last byte is a checksum and must be hand modified. The checksum is the least significant byte of the one's complement of the sum of the values represented by the pairs of characters making up the record length, address, and the code/data fields.

Programming the Shadow Flash (continued)

- Modify S-record to Change the LSM/DPM Configuration

► Note the S2 s-record format is:

- S2xyyyyyyyzzzzcs
 - S = one character start code
 - 2 = one character record type
 - xx = two hex digits indicating the number of bytes (hex digit pairs) for address, data, and checksum fields
 - yyyyyy = in an S2 s-record 6 hex digits specifying the memory location of the first data byte
 - zzzz = 0-64 pairs of hex characters specifying the data bytes
 - cs = 2 hex digits checksum character

► Note that the last byte is a checksum and must be hand modified. To calculate this, take the sum of the values of all bytes from the byte count up to the last data byte, inclusive, modulo 256. Subtract this result from 255.

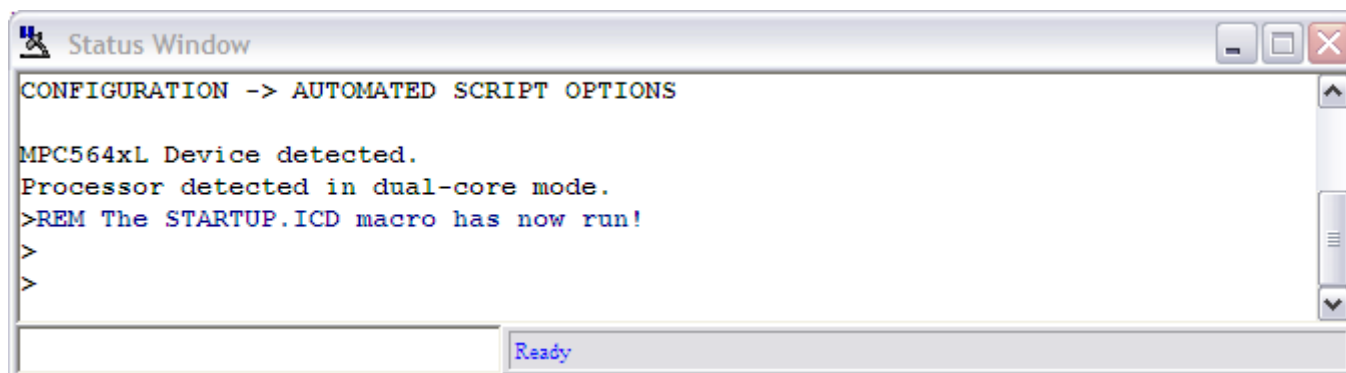
Programming the Shadow Flash (continued)

- Program the Shadow Flash with the Updated Configuration

- ▶ First the shadow row must be erased by pressing the *Erase Module* Button
- ▶ Now the updated s-record file can be programmed. Specify the file by pressing the *Specify Object File* Button and selecting the updated s-record. Now download the s-record into the Shadow flash by pressing the *Program Module* button.
- ▶ The *Verify Module* Button can be used to check the programming operation.

Programming the Shadow Flash (continued) - Verify Changed Configuration

- ▶ Power cycle the EVB to cause it to restart in the new configuration.
- ▶ Start the P&E debugger (*CW_ICDPPCNEXUS.EXE*) and connect to the MCU as before. The programmer and debugger use the same connection utility.
- ▶ The status window, shown below, will confirm what mode the processor is operating in. Check that it matches with your shadow flash configuration.

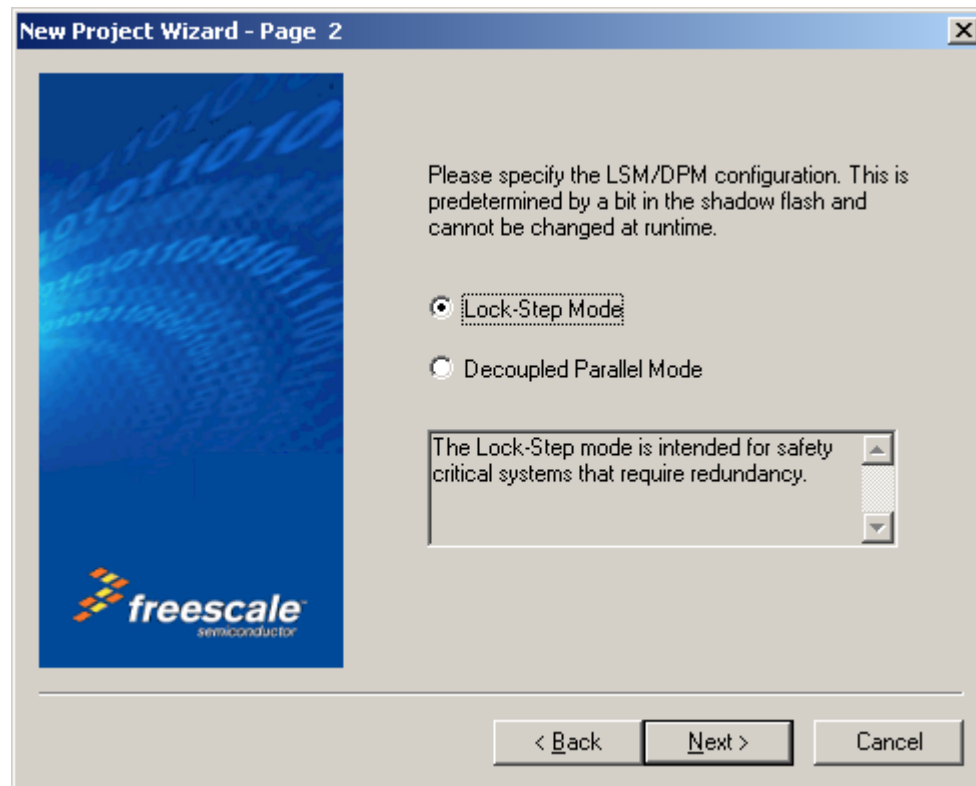


CodeWarrior Development Studio

- ▶ The CodeWarrior Development Studio is a tool chain developed by Freescale. These tools ship as part of the EVB kit.
- ▶ *CodeWarrior MPC55xx/MPC56xx v2.6* supports the Freescale *MPC5643L* microprocessor.

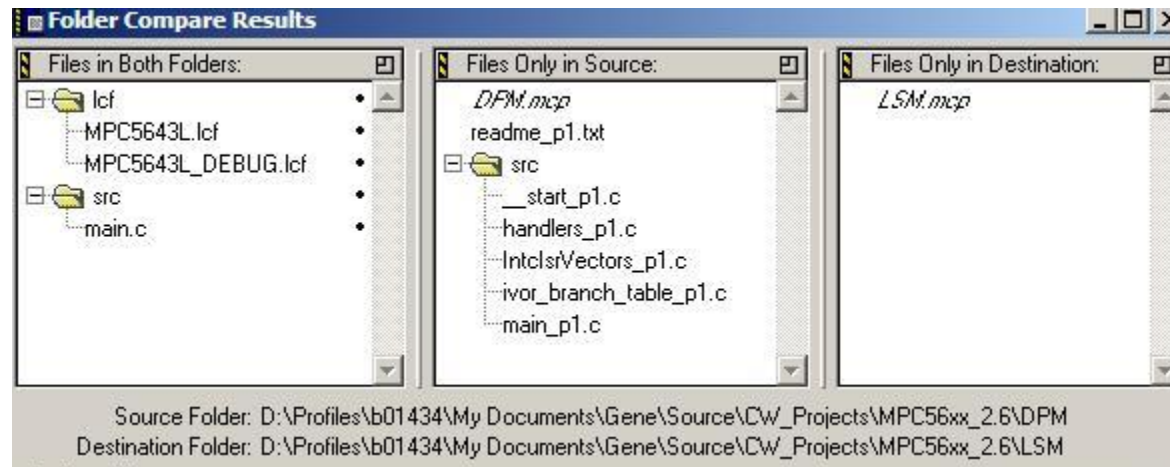
CodeWarrior LSM/DPM Support

- ▶ In *CodeWarrior MPC55xx v2.6*, a user has the option to create a single or dual-core project. If dual-core is selected, then LSM or DPM configuration can be selected:



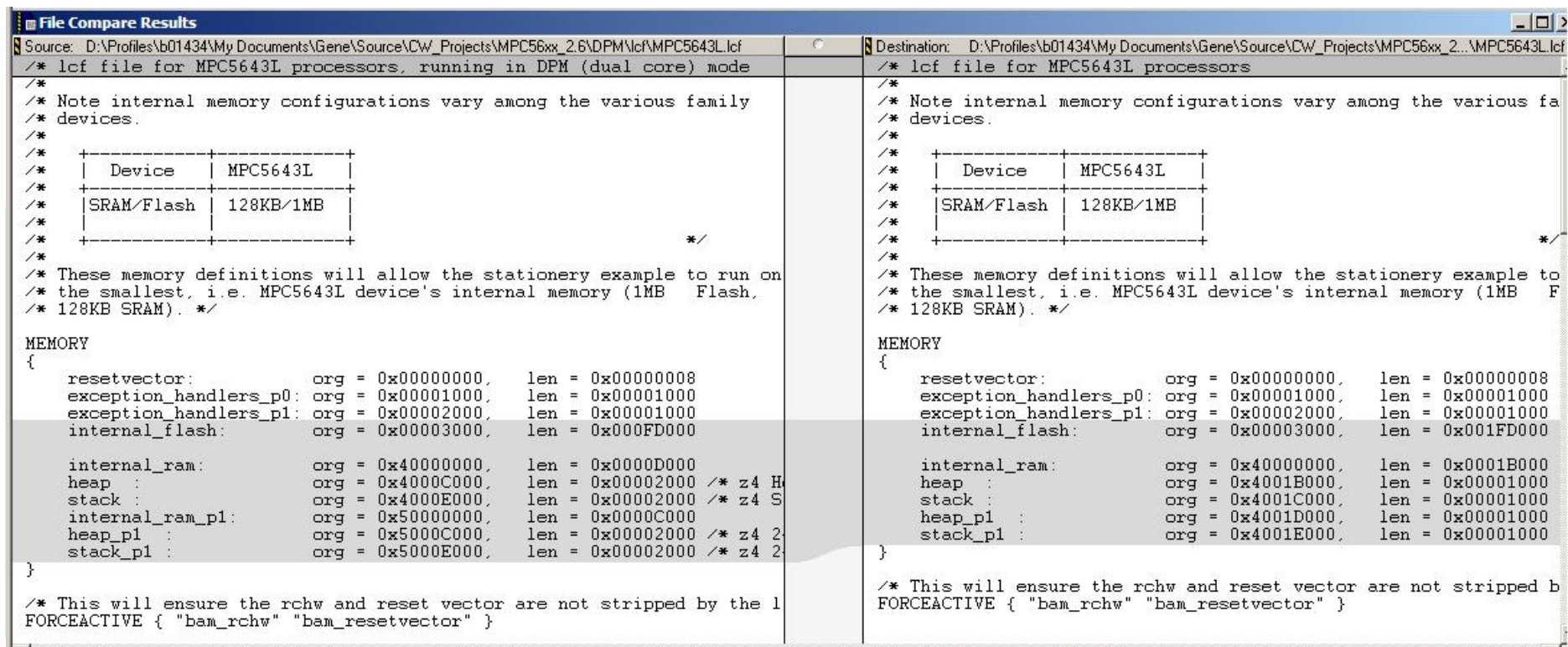
CodeWarrior LSM/DPM Support (continued)

- ▶ The LSM project provides the source code and the linker configuration file to build a sample *MPC5643L* LSM image.
- ▶ The DPM project provides unique source code for both cores and the linker configuration file to build a sample *MPC5643L* DPM image.
- ▶ Below is a screen shot listing of the files that differ between the LSM and DPM projects:



CodeWarrior LSM/DPM Support

- ▶ The DPM linker control file (left) differs from the LSM linker control file (right) because it allocates memory for the 2nd core:



```
File Compare Results
Source: D:\Profiles\b01434\My Documents\Gene\Source\CW_Projects\MPC56xx_2.6\DPM\lcf\MPC5643L.lcf
/* lcf file for MPC5643L processors, running in DPM (dual core) mode
/*
/* Note internal memory configurations vary among the various family
/* devices.
/*
/* Device      MPC5643L
/* SRAM/Flash  128KB/1MB
/*
/*
/* These memory definitions will allow the stationery example to run on
/* the smallest, i.e. MPC5643L device's internal memory (1MB Flash,
/* 128KB SRAM). */

MEMORY
{
    resetvector:      org = 0x00000000, len = 0x00000008
    exception_handlers_p0: org = 0x00001000, len = 0x00001000
    exception_handlers_p1: org = 0x00002000, len = 0x00001000
    internal_flash:    org = 0x00003000, len = 0x000FD000

    internal_ram:      org = 0x40000000, len = 0x0000D000
    heap :              org = 0x4000C000, len = 0x00002000 /* z4 H
    stack :             org = 0x4000E000, len = 0x00002000 /* z4 S
    internal_ram_p1:    org = 0x50000000, len = 0x0000C000
    heap_p1 :           org = 0x5000C000, len = 0x00002000 /* z4 2
    stack_p1 :          org = 0x5000E000, len = 0x00002000 /* z4 2
}

/* This will ensure the rchw and reset vector are not stripped by the 1
FORCEACTIVE { "bam_rchw" "bam_resetvector" }
```

```
Destination: D:\Profiles\b01434\My Documents\Gene\Source\CW_Projects\MPC56xx_2.6\LSM\lcf\MPC5643L.lcf
/* lcf file for MPC5643L processors
/*
/* Note internal memory configurations vary among the various fa
/* devices.
/*
/* Device      MPC5643L
/* SRAM/Flash  128KB/1MB
/*
/*
/* These memory definitions will allow the stationery example to
/* the smallest, i.e. MPC5643L device's internal memory (1MB F
/* 128KB SRAM). */

MEMORY
{
    resetvector:      org = 0x00000000, len = 0x00000008
    exception_handlers_p0: org = 0x00001000, len = 0x00001000
    exception_handlers_p1: org = 0x00002000, len = 0x00001000
    internal_flash:    org = 0x00003000, len = 0x001FD000

    internal_ram:      org = 0x40000000, len = 0x0001B000
    heap :              org = 0x4001B000, len = 0x00001000
    stack :             org = 0x4001C000, len = 0x00001000
    heap_p1 :           org = 0x4001D000, len = 0x00001000
    stack_p1 :          org = 0x4001E000, len = 0x00001000
}

/* This will ensure the rchw and reset vector are not stripped b
FORCEACTIVE { "bam_rchw" "bam_resetvector" }
```

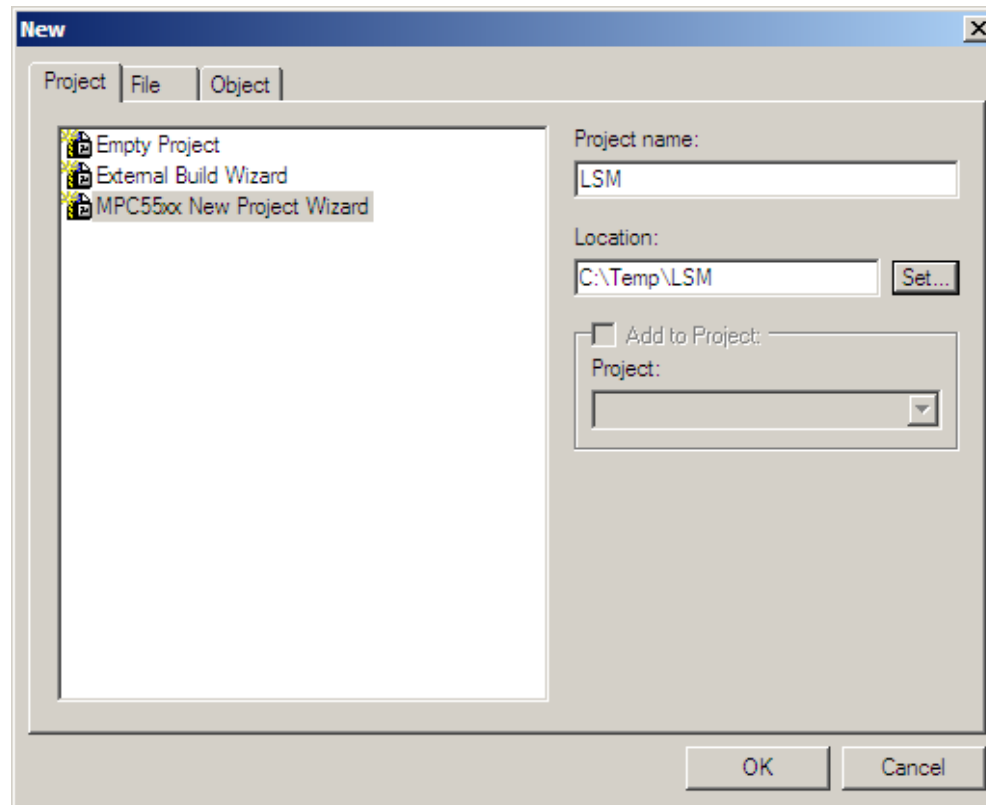
CodeWarrior LSM/DPM Support

- ▶ The DPM main.c file (left) differs from the LSM main.c file (right) as it must start the code execution on the 2nd core:

Source: D:\Profiles\b01434\My Documents\Gene\Source\CW_Projects\MPC56xx_2.6\DPM\src\main.c	Destination: D:\Profiles\b01434\My Documents\Gene\Source\CW_Projects\MPC56xx_2.6\LSM\src\main.c
<pre>#include "MPC5643L.h" /* Prototype for second core startup */ extern void __start_p1(); int main(void) { volatile int i = 0; /* Start the second core, VLE mode*/ SSCM.DPMBOOT.R = (unsigned long)__start_p1 + 0x00000002; SSCM.DPMKEY.R = 0x00005AF0; SSCM.DPMKEY.R = 0x0000A50F; /* Loop forever */ for (;;) { i++; } }</pre>	<pre>#include "MPC5643L.h" int main(void) { volatile int i = 0; /* Loop forever */ for (;;) { i++; } }</pre>

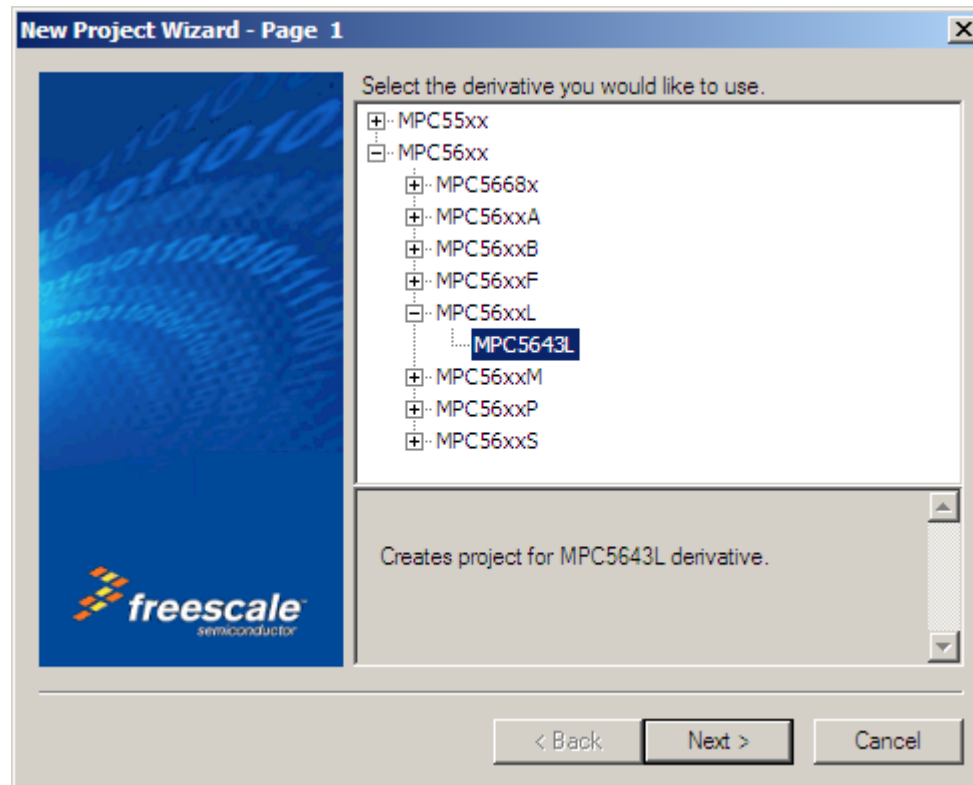
Building LSM/DPM with CodeWarrior

- ▶ Start the CodeWarrior IDE.
- ▶ Press *File* → *New*
- ▶ Select the *MPC55xx New Project Wizard* and give a name to the new project in a location and press *OK*:



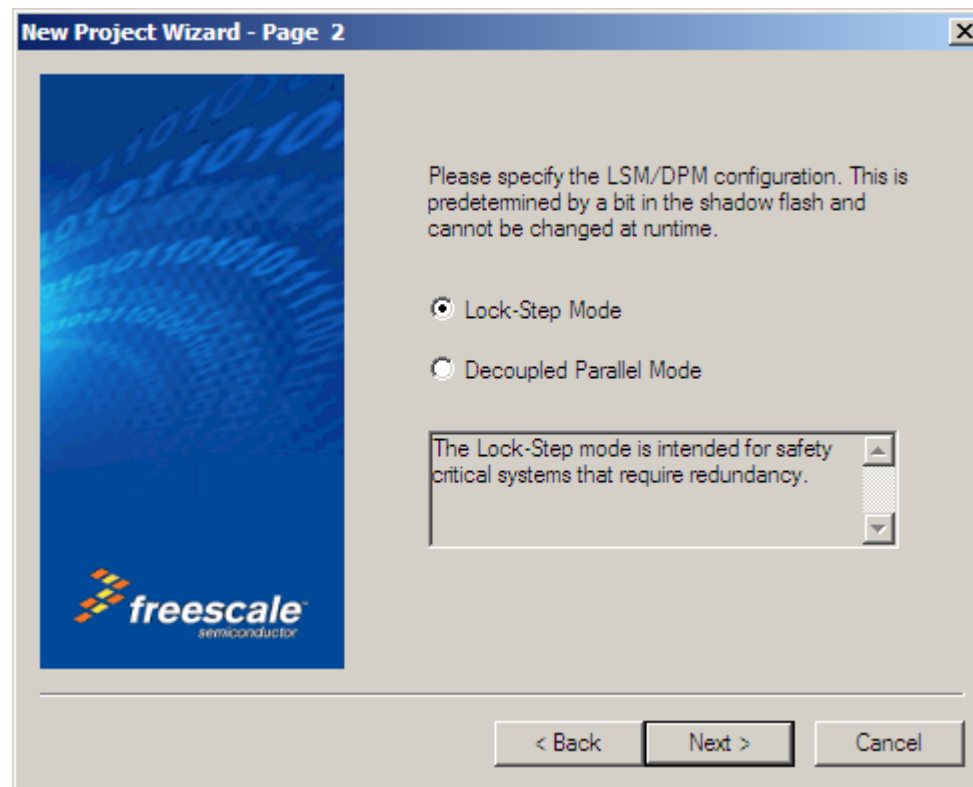
Building LSM/DPM with CodeWarrior (cont'd)

- Ensure the project is being built for an *MPC5643L* and press *Next*.



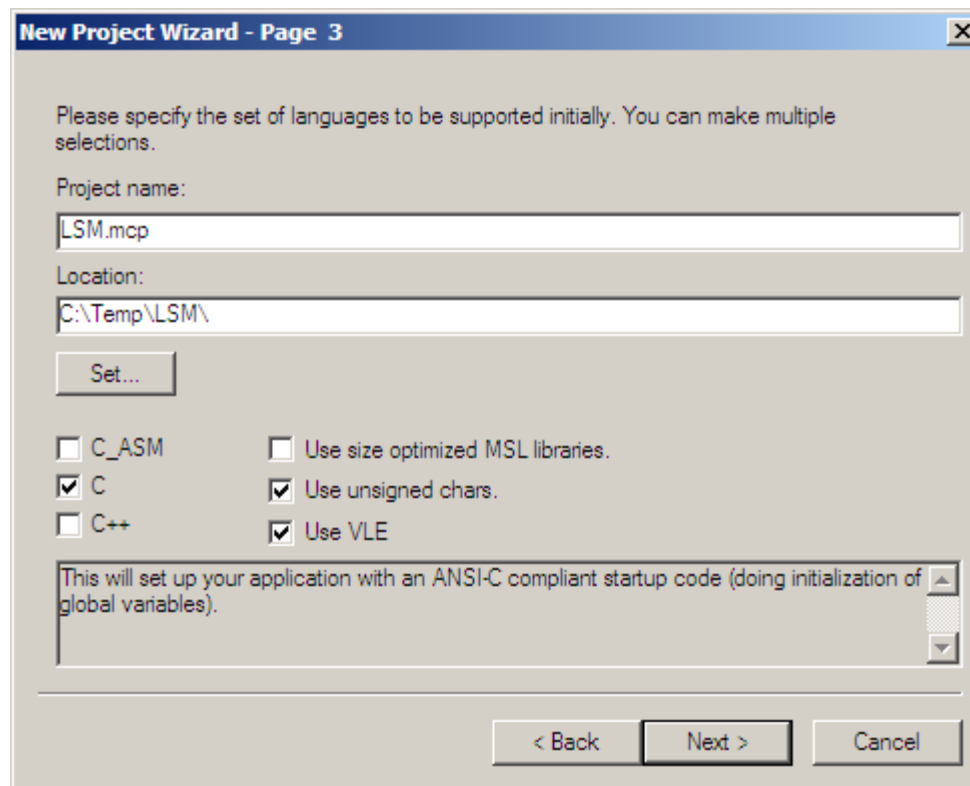
Building LSM/DPM with CodeWarrior (cont'd)

- ▶ Ensure the project is being built for Lock-Step Mode (this is the only difference from making an *MPC5643L* DPM project) and press *Next*.



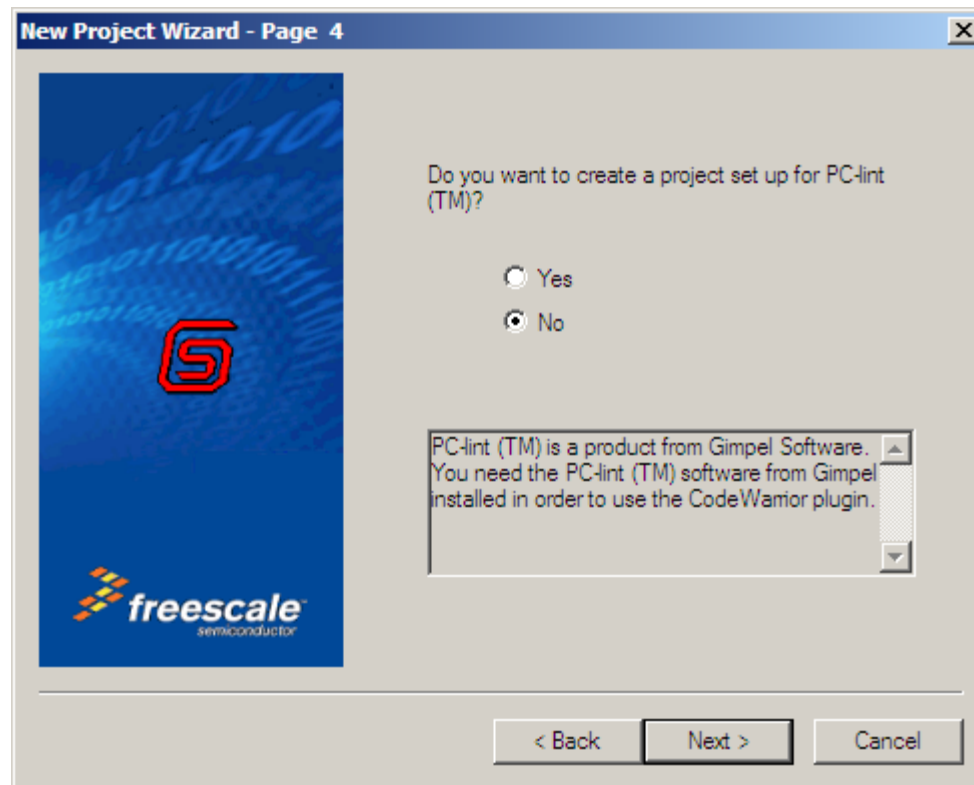
Building LSM/DPM with CodeWarrior (cont'd)

- Accept the language support defaults offered by the project wizard by pressing *Next*.



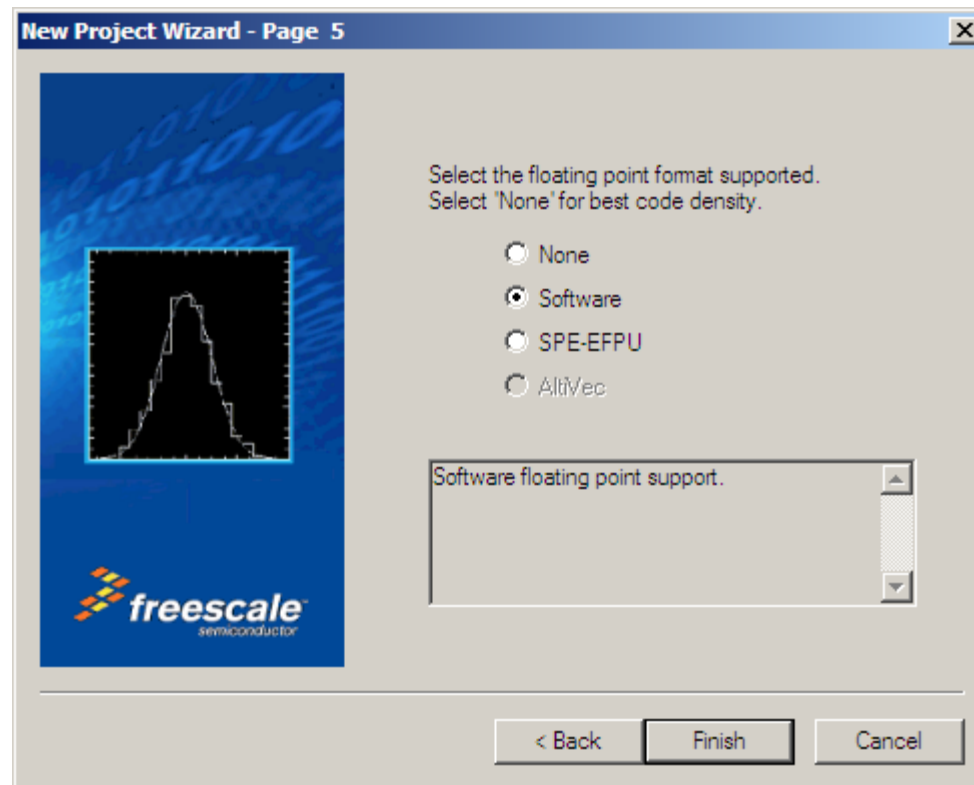
Building LSM/DPM with CodeWarrior (cont'd)

- Accept the PC-lint defaults offered by the project wizard by pressing *Next*.



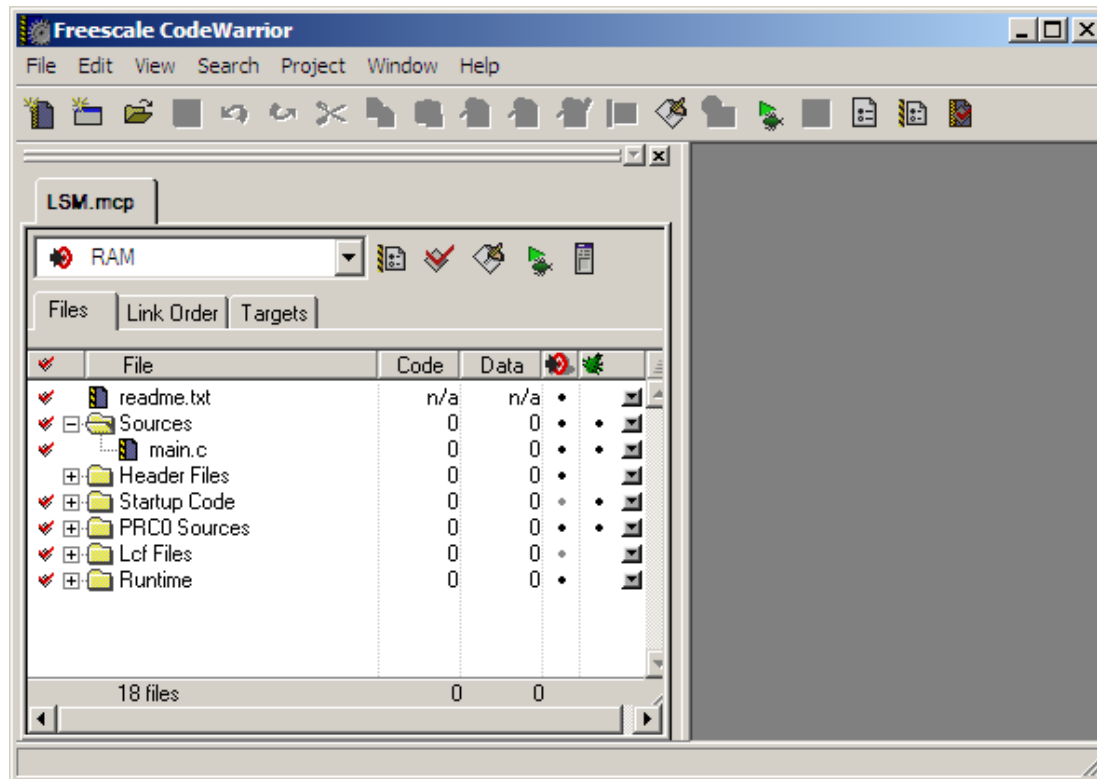
Building LSM/DPM with CodeWarrior (cont'd)

- Accept the floating point format defaults offered by the project wizard and press *Finish*:



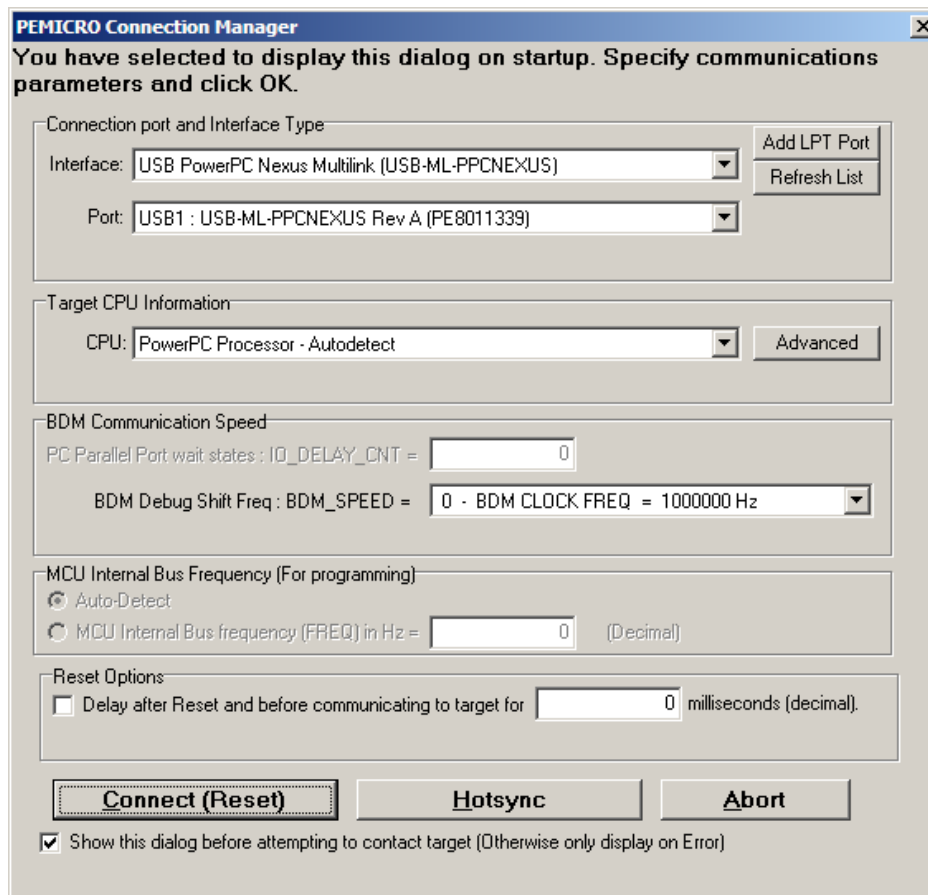
Building LSM/DPM with CodeWarrior (cont'd)

- ▶ Once you have the project screen, select the “*internal_Flash*” target instead of RAM.
- ▶ Press “*F7*” to “make” the target.
- ▶ Press “*F5*” to “debug” the target.



Loading and Executing LSM/DPM with CodeWarrior (cont'd)

- Once you have the debugger screen, press the *Connect (Reset)* button to burn and run the Flash image:



The image shows the PEMICRO Connection Manager dialog box. It has a title bar with a close button. The main text says: "You have selected to display this dialog on startup. Specify communications parameters and click OK." The dialog is divided into several sections:

- Connection port and Interface Type:** Contains a dropdown for "Interface:" set to "USB PowerPC Nexus Multilink (USB-ML-PPCNEXUS)", a dropdown for "Port:" set to "USB1 : USB-ML-PPCNEXUS Rev A (PE8011339)", and two buttons: "Add LPT Port" and "Refresh List".
- Target CPU Information:** Contains a dropdown for "CPU:" set to "PowerPC Processor - Autodetect" and an "Advanced" button.
- BDM Communication Speed:** Contains a text field for "PC Parallel Port wait states : IQ_DELAY_CNT =" set to "0", and a dropdown for "BDM Debug Shift Freq : BDM_SPEED =" set to "0 - BDM CLOCK FREQ = 1000000 Hz".
- MCU Internal Bus Frequency (For programming):** Contains two radio buttons: "Auto-Detect" (selected) and "MCU Internal Bus frequency (FREQ) in Hz =" (set to "0" with "(Decimal)" next to it).
- Reset Options:** Contains a checkbox "Delay after Reset and before communicating to target for" (unchecked) followed by a text field set to "0" and the text "milliseconds (decimal)".

At the bottom, there are three buttons: "Connect (Reset)" (highlighted with a dashed border), "Hotsync", and "Abort". Below these buttons is a checkbox "Show this dialog before attempting to contact target (Otherwise only display on Error)" which is checked.

