

CodeWarriors & Processor Expert Seminar

AMF-GEN-T0011

CodeWarriors IDE and Processor Expert In-depth Training
Presented by Hy Mai and Ruth Rhoades

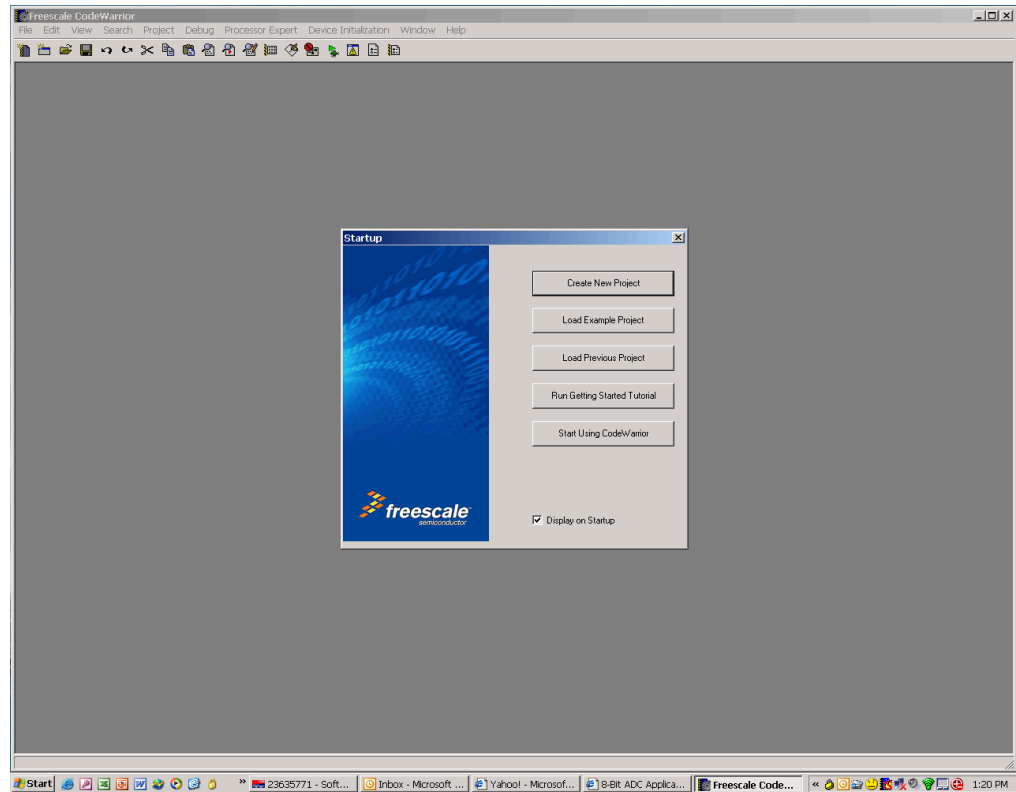
- ▶ How to position and sales Processor Expert
- ▶ LAB1 -- Blinking LED (Timer and I/O)
- ▶ LAB2 – KBI & Analog Compartor
- ▶ LAB3 -- ADC and PWM
- ▶ LAB4 – SCI (Uart) Communication with PC
- ▶ LAB5 – SPI Serial Peripheral Comm neighbor's board
- ▶ LAB6 – I2C Comm with neighbor's board
- ▶ LAB7 – Input Capture to measure Pulsewidth (Optional)
- ▶ LAB8 – Make check payable to He My FAE

LAB1 -- Blinking LED

- ▶ Configure Bus Clock to 8Mhz
- ▶ Set Timer to generate an interrupt every 250ms
- ▶ Configure output on LED2 (PORTB7)
- ▶ Toggle LED at every Timer Interrupt

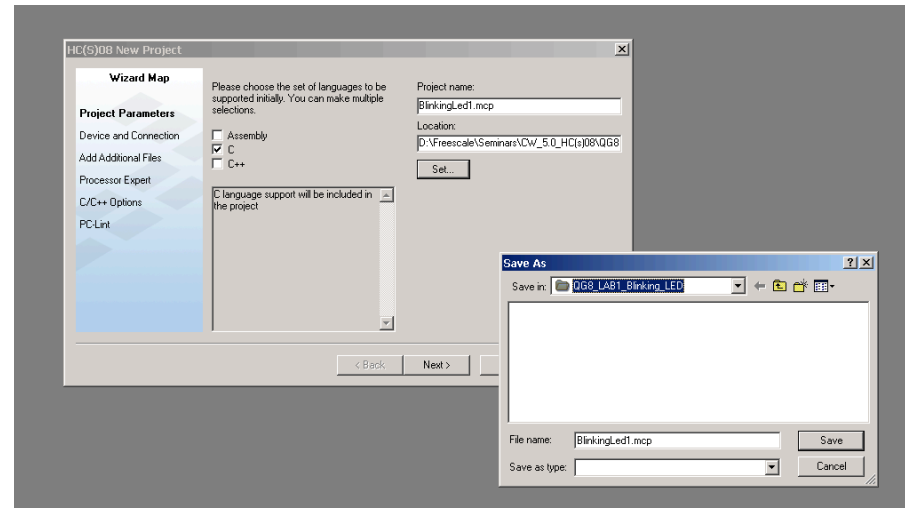
LAB1 -- Blinking LED

- Click “Create New Project”

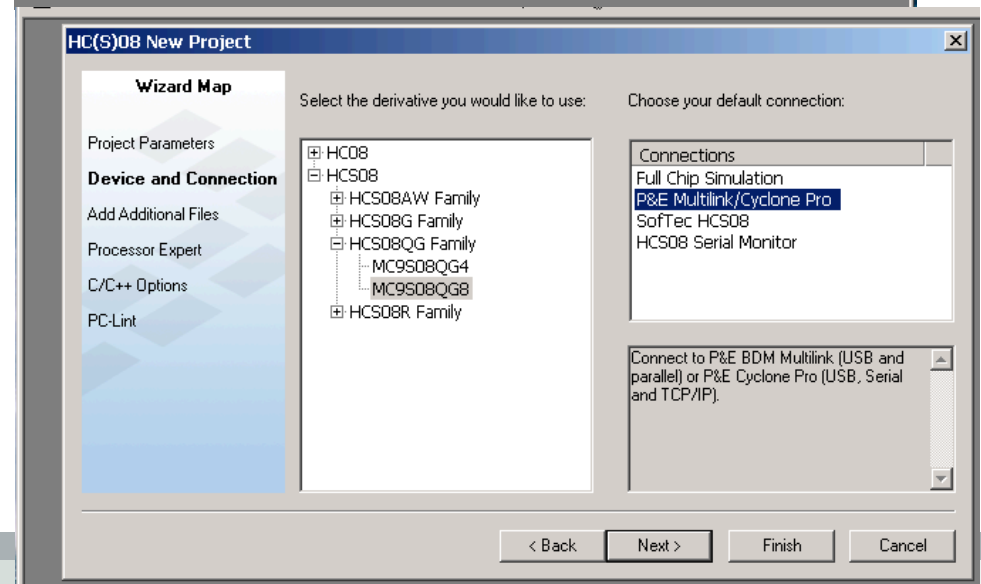


LAB1 -- Blinking LED

- ▶ Click on the “Set” button to set the directory
- ▶ Give the project a name and hit “Save” and then hit “Next”

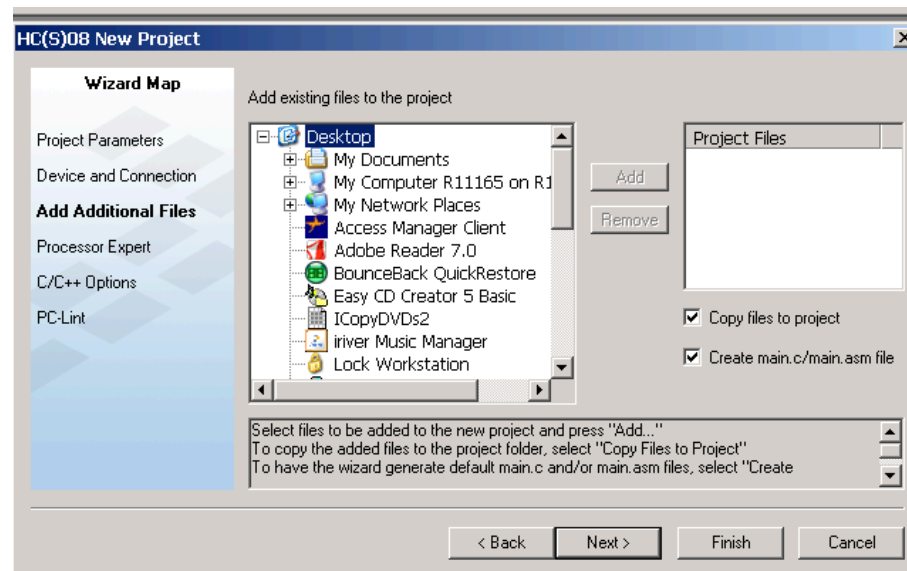


- Expand the “HCS08” directory structure to and find the “MC9S08QG8” device
- Select “P&E Multilink/Cyclone Pro” under the Connections box, then hit “Next”

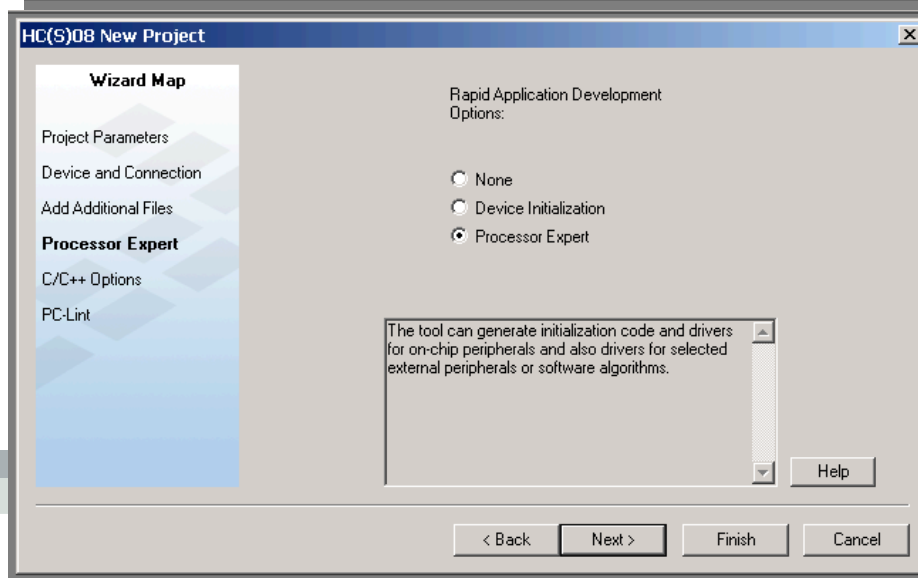


LAB1 -- Blinking LED

► Since we are not importing any existing files just hit “Next”



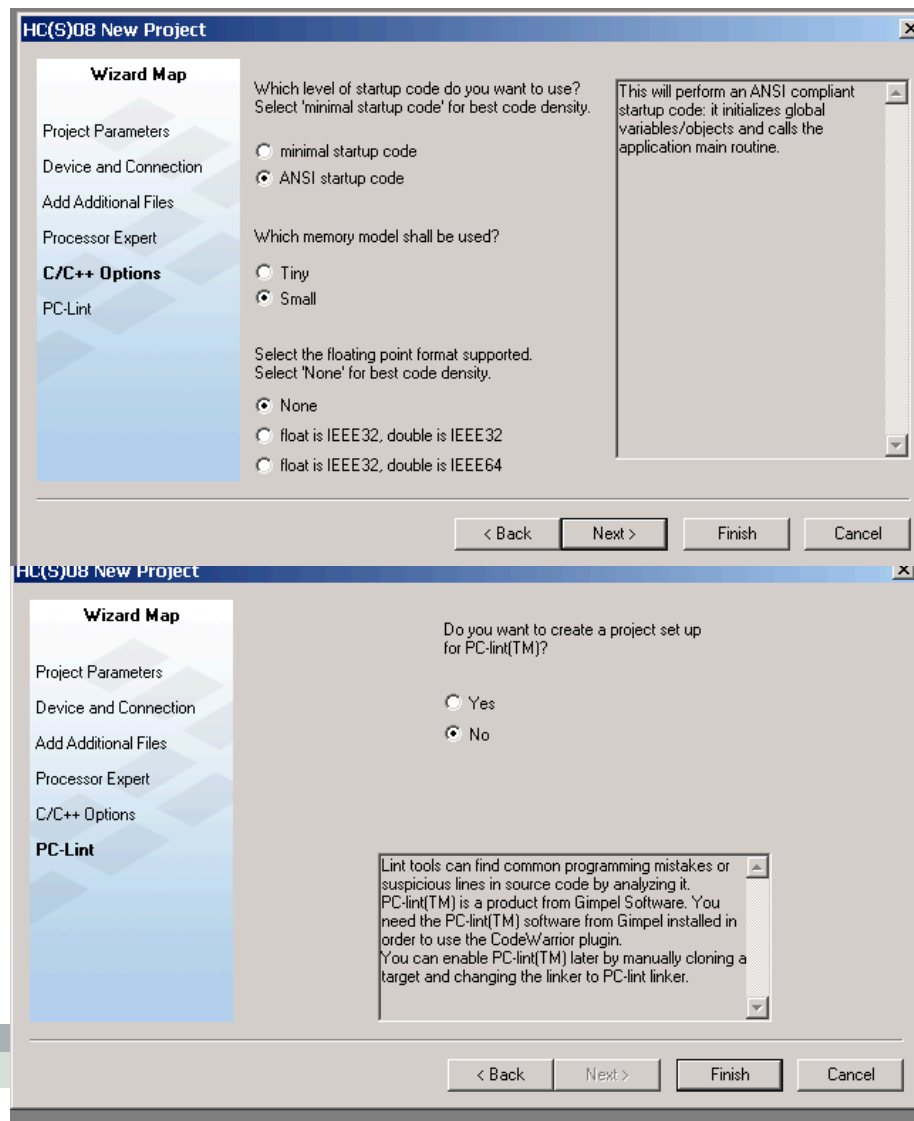
► Select “Processor Expert” and hit “Next”



LAB1 -- Blinking LED

► Leave everything as default and click “Next”

► Leave everything as default and click “Finish”

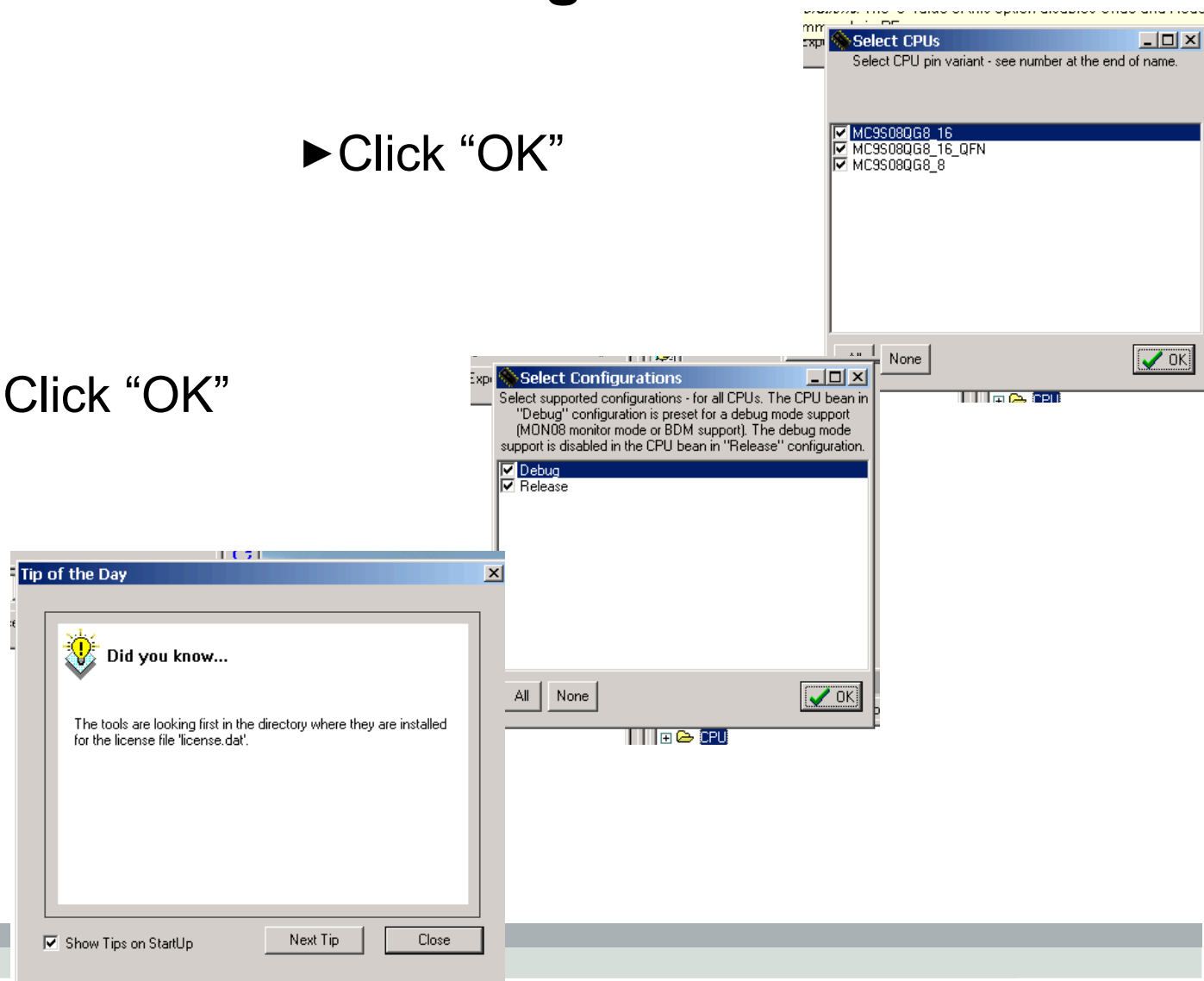


LAB1 -- Blinking LED

► Click "OK"

► Click "OK"

► Click "Close"



LAB1 -- Blinking LED

The screenshot shows the Freescale CodeWarrior IDE interface. The main window displays a project named 'BlinkingLed1.mcp' for a 'P&E Multilink/Cyclone Pro' target. The 'Bean Inspector' window is open, showing the configuration for the 'Cpu:MC9S08QG8CPB' bean. The 'Internal bus clock' is set to 8.0 MHz. A tooltip is visible over the 'Internal bus clock' value, stating: 'The internal bus clock frequency [MHz]. Producer recommends frequency under 10.0MHz.' The 'Bean Selector' window shows the 'CPU Internal Peripherals' category selected. The Windows taskbar at the bottom shows the Start button, several application icons, and the system tray with the time 1:46 PM.

► You should get a window that looks somewhat like this.

► In the “Bean Inspector Window” change the “Internal Bus Clock” from “4Mhz” to “8Mhz”.

LAB1 -- Blinking LED

The screenshot shows the Freescale CodeWarrior IDE interface. The main window displays the 'Bean Selector' dialog, which is used to configure beans for a target CPU. The 'Bean Selector' window is expanded to show the 'CPU Internal Peripherals' category, and the 'Timer' sub-category is selected. The 'TimerInt' bean is highlighted. A 'Confirm' dialog box is overlaid on the screen, asking: 'Do you want to enable the bean in all configurations? Select "No" if you want to use the bean in current configuration only.' The 'Yes' button is highlighted.

Confirm

Do you want to enable the bean in all configurations?
Select "No" if you want to use the bean in current configuration only.

Do not ask again

► In the “Bean Selector” window, expand the “CPU Internal Peripherals”, “Timer”, and select the “TimerInt” bean. Then Click “Yes” to confirm.

LAB1 -- Blinking LED

► In the “Bean Inspector” window, type in “250”(mSec)

Property	Value
Bean name	TI1
Timer	MTIMmod
Counter	MTIM
Interrupt service/event	Enabled
Interrupt	Vmtim
Interrupt priority	medium prior
MTIM clock source in high speed mode	Auto select
Prescaler	Auto select
Interrupt period	250 ms

Period/frequency of the interrupts generation. It is necessary to specify both a value and a unit (see [Timing Setting Syntax](#)). The setting can be made with the help of the [Timing dialog box](#) that gets open when clicking on button (...).
Selected runtime setting: fixed value

LAB1 -- Blinking LED

► In the “Bean Selector” window, expand the “Port I/O” and select the “BitIO” and click “Yes” to confirm.

The screenshot displays the Freescale CodeWarrior IDE interface. The main window shows a project named "BlinkingLed1.mcp" with a target CPU of "Cpu:MC9S08QG8CPB". The Bean Selector window is open, showing a tree view of beans. Under "Port I/O", the "BitIO" bean is selected. A "Confirm" dialog box is displayed, asking "Do you want to enable the bean in all configurations? Select 'No' if you want to use the bean in current configuration only." The "Yes" button is highlighted. The Bean Inspector window is also visible, showing properties for the selected "BitIO" bean, such as "Bean name: Bit1", "Pin for I/O: PTB7_SCL", and "Direction: Input/Output".

LAB1 -- Blinking LED

► In the “Bean Selector” window, expand the “Port I/O” and select the “BitIO” and click “Yes” to confirm.

The screenshot displays the Freescale CodeWarrior IDE interface. The main window shows a project named "BlinkingLed1.mcp" with a target CPU of "MC9S08QG8CPB". The Bean Selector window is open, showing a tree view of components. The "Port I/O" category is expanded, and "BitIO" is selected. The "Bean Inspector" window is also open, showing the configuration for the "BitIO" bean. The configuration includes:

Property	Value
Bean name	Bit1
Pin for I/O	PTB7_SCL_EXTAL
Pin signal	PTB7_SCL_I
Pull resistor	autoselected pull
Open drain	no open drain
Slew rate control for PTB7	yes
Drive strength for PTB7	Low
Direction	Output
Initialization	Input
Init. direction	Output
Init. value	Input/Output
Safe mode	yes
Optimization for	speed

The Bean Selector window also shows a list of categories: CPU, CPU External Devices, CPU Internal Peripherals, Communication, Converter, Interrupts, Measurement, Memory, Peripheral Initialization Beans, Port I/O, and Timer. The "Port I/O" category is expanded, and "BitIO" is selected. The "Bean Inspector" window is also open, showing the configuration for the "BitIO" bean. The configuration includes:

LAB1 -- Blinking LED

► In the “Bean Inspector” window, select the “Methods” tab, and toggle the “Circular Arrow” for “NegVal” to switch to “generate code”

The screenshot shows the Freescale CodeWarrior IDE interface. The main window displays the project tree on the left, with the 'Beans' folder expanded to show 'Bit1:BitIO'. The 'Bean Inspector' window is open, showing the 'Methods' tab. The 'NegVal' method is selected, and its 'generate code' icon (a circular arrow) is active. A tooltip for the 'NegVal' method is displayed, stating: 'Negates (inverts) the Input/Output bean. It is the same as PutVal(!GetVal()). (Method is available only if the direction = output or input/output). void Bit1_NegVal(void);'. The 'Bean Selector' window is also visible, showing the 'Port I/O' category selected.

LAB1 -- Blinking LED

The screenshot shows the Freescale CodeWarrior interface. The 'Processor Expert' menu is open, with 'Generate Code 'BlinkingLed1.mcp'' selected. A 'Code generation' dialog box is also open, showing the project name 'BlinkingLed1', module 'Events', and a progress bar. The 'Code Generation' dialog box displays the following information:

- Project: BlinkingLed1
- Module: Events
- Preparing for code gener. Total lines: 1
- Errors: 0 Warnings: 0 Hints: 0
- Cancel button

A blue callout box on the left side of the screenshot contains the text: "► From the 'Processor Expert' tab, select 'Generate Code'"

LAB1 -- Blinking LED

- ▶ Select the “Processor Expert” Tab, then expand the “BitIO” bean.
- ▶ Drop n Drag the “NegVal” call into the “Write your code here” section.

The screenshot shows the Freescale CodeWarrior IDE interface. The top menu bar includes File, Edit, View, Search, Project, Debug, Processor Expert, Device Initialization, Window, and Help. The main workspace is divided into several panes:

- Processor Expert Tab:** Shows a project named "BlinkingLed1.mcp" for the target "P&E Multilink/Cyclone Pro". The "Processor Expert" sub-tab is active, displaying a tree view of beans. Under the "Beans" folder, the "Bit1:BitIO" bean is expanded, showing various methods like GetDir, SetDir, SetInput, SetOutput, GetVal, PutVal, ClrVal, SetVal, and NegVal. The "NegVal" method is highlighted with a blue selection box.
- Events.c File:** The code editor shows the definition of the "TI1_OnInterrupt" event. The code includes a comment describing the event and a function signature: `void TI1_OnInterrupt(void)`. Below the function signature, there is a comment: `/* Write your code here ... */`. The `Bit1_NegVal();` call is visible in the code, with an arrow pointing from the "NegVal" method in the Processor Expert tree to this call.
- Target Properties:** The "Target CPU" is set to "Cpu:MC9S08QG8CPB".
- Bean Inspector:** The "Bean Inspector" shows the "Bit1:BitIO" bean selected.
- Measurement and Memory:** The bottom right pane shows "Measurement" and "Memory" sections, with "Port I/O" expanded to show "BitIO", "BitsIO", and "ByteIO".

LAB1 -- Blinking LED

- ▶ Hit the “Debug” Icon, and then click “Connect” to engage the BDM
- ▶ Click “Yes” to load image to Flash

The screenshot shows the Freescale CodeWarrior IDE interface. The main window displays the 'BlinkingLed1.mcp' project with a 'Debug' icon highlighted. A blue callout box points to this icon with the instruction: 'Hit the “Debug” Icon, and then click “Connect” to engage the BDM'. Below this, another instruction says: 'Click “Yes” to load image to Flash'. The 'ICD - Connection Manager' dialog box is open, showing the 'USB HCS08/HCS12 Multilink - USB Port' interface and 'DEMO9S08QG8 on USB1 (Name=PE5122327) (Autodetected)' port. The 'Target CPU Information' section shows 'CPU: HCS08 Processor - Autodetect'. The 'Erase and Program Flash?' dialog box is also open, displaying the message: 'Load image contains flash memory data. Erase and Program flash?' with 'Yes' and 'No' buttons. The 'Yes' button is highlighted with a green checkmark. The 'Connect' button is visible in the background dialog box. The Windows taskbar at the bottom shows the system tray with the time 4:45 PM.

LAB1 -- Blinking LED

True-Time Simulator & Real-Time Debugger D:\Freescale\Seminars\CW_5.0_HC(s)08\QG8_LAB1_Blinking_LED...

File View Run MultilinkCyclonePro Component Memory Window Help

Start/Continue (F5)

Source D:\Freescale\Seminars\CW_5.0_HC(s)08\QG8_LAB1_Blinking_LED\Blir Line: 38

```
{  
  /** Processor Expert internal initialization. DON'T REMOVE  
  PE_low_level_init();  
  /** End of Processor Expert internal initialization.  
  
  /* Write your code here */  
  /* For example: for(;;) { } */  
}
```

Assembly

```
main  
E0FD JSR 0xE02D  
E100 BRA *+0 ;abs = 0xE100  
E102 BRSET 0,0x00,*+2 ;abs =  
E105 STX ,X
```

Register

HCS08 Auto

A 0
HX E100 SP BF
SR 6A Status VHINZC

Memory

Auto

```
0080 C7 18 26 A6 80 C7 18 25 ..&...%  
0088 81 87 C6 18 25 A4 40 27 ....%.0'  
0090 F9 86 81 01 87 88 89 F6 .....
```

Data:2

main Auto Symb Local

Command

```
Breakpoint  
  
in>
```

Start/Continue program Automatic (triggers, breakpoints, watchpoints, and trace possible) 9S08QG8 ICD Bre

- ▶ Hit “Start” icon to run the program.
- ▶ You should see a blinking LED

LAB 2 – KBI & ACMP

- ▶ Initialized KBI on SW2(PORTA3)
- ▶ Use SW2 to toggle LED1(PORTB6).
- ▶ Initialized Analog Comparator
- ▶ To use Photolight sensor to toggle LED2(PORTB6)

LAB2 – KBI & ACMP

The image shows two windows from a development environment. The 'Bean Selector' window on the left displays a tree view of components. The 'KBI' bean is highlighted under the 'Peripherals' folder. A tooltip for 'KBI' is visible, stating 'Bean Level: Low Level Bean' and 'Double click to insert the bean into current project'. The 'Bean Inspector KB1:KBI' window on the right shows the configuration for the selected bean. The 'Properties' tab is active, showing the following settings:

Property	Value	Control
Bean name	KB1	
KBI	KBI	▼ KBI
Pins	1	+ -
Pin0		
Pin	PTA3_KBIP3_SCL_ADP3	▼ PTA3_KBIP3_SCL_A
Pin signal		
Invert interrupt trigger condition	no	⊗
Pull resistor	pull up	▼ pull up
Generate interrupt on	falling edge	▼ falling edge
Interrupt service/event	Enabled	⊗
Interrupt	Vkeyboard	▼ Vkeyboard
Interrupt priority	medium priority	▼ not supported
Initialization		
Enable in init. code	yes	⊗
Events enabled in init.	yes	⊗

- ▶ From the “Bean Selector” click on “KBI” bean
- ▶ Configure the Pin 0 for the KBI to be “PTA3_KBIP3_SCL_ADP3”
- ▶ Select “Pull Up” and “falling edge” options .
- ▶ Make sure that Interrupts service is “Enabled”

LAB2 – KBI & ACMP

► Click on “FreescaleAnalogComp” bean

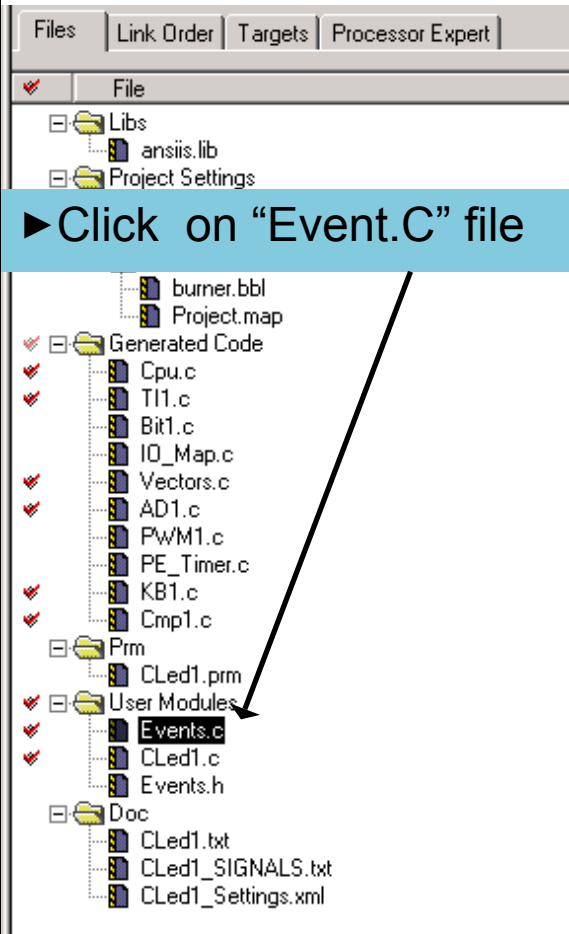
► Configure “Analog comp mode to “falling edge” option

► Make sure that Interrupts service is “Enabled”

► Click on “Generate Code”

Property	Value
Bean name	Cmp1
Analog comparator.	ACMP
Interrupt service/event	Enabled
Interrupt	Vacmp
Pin	VoltageReference
Signal	VoltageRefer
Negative input	
Pin	PTA1_KBIP1_ADP1_ACMPMI
Signal	PTA1_KBIP1
Comparator output	Disabled
Analog comp. mode	falling edge
Initialization	
Enabled in init. code	yes

LAB2 – KBI & ACMP



```
**      comparator output. Interr
**      enabled for this event to
**      Parameters   : None
**      Returns      : Nothing
**      =====
*/
void Cmp1_OnCompare(void)
{
    /* Write your code here ... */
}

/*
**      =====
**      Event       : KBI_OnInterrupt
**      From bean   : KBI [KBI]
**      Description :
**      This event is called when
**      occurs.
**      Parameters  : None
**      Returns     : Nothing
**      =====
*/
void KBI_OnInterrupt(void)
{
    /* Write your code here ... */
}

/* END Events */
```

- Scroll down to these two Events
- This is where you would add the code to toggle LED1

```
#####
ated by UNIS
HCS08 serie
#####
```

LAB2 – KBI & ACMP

The screenshot shows the P&E Multilink/Lyclose Pro IDE. On the left, the 'Beans' tree is expanded to show 'Cmp1:FreescalarAnalogComp'. On the right, the code editor shows the implementation of the KBI interrupt service routine (ISR) and the compare event handler. The KBI ISR code includes a 'poor man debounce code' consisting of two nested loops over 0xff. The compare event handler also calls 'Bit1_NegVal()'. A blue callout box with the following instructions is overlaid on the code:

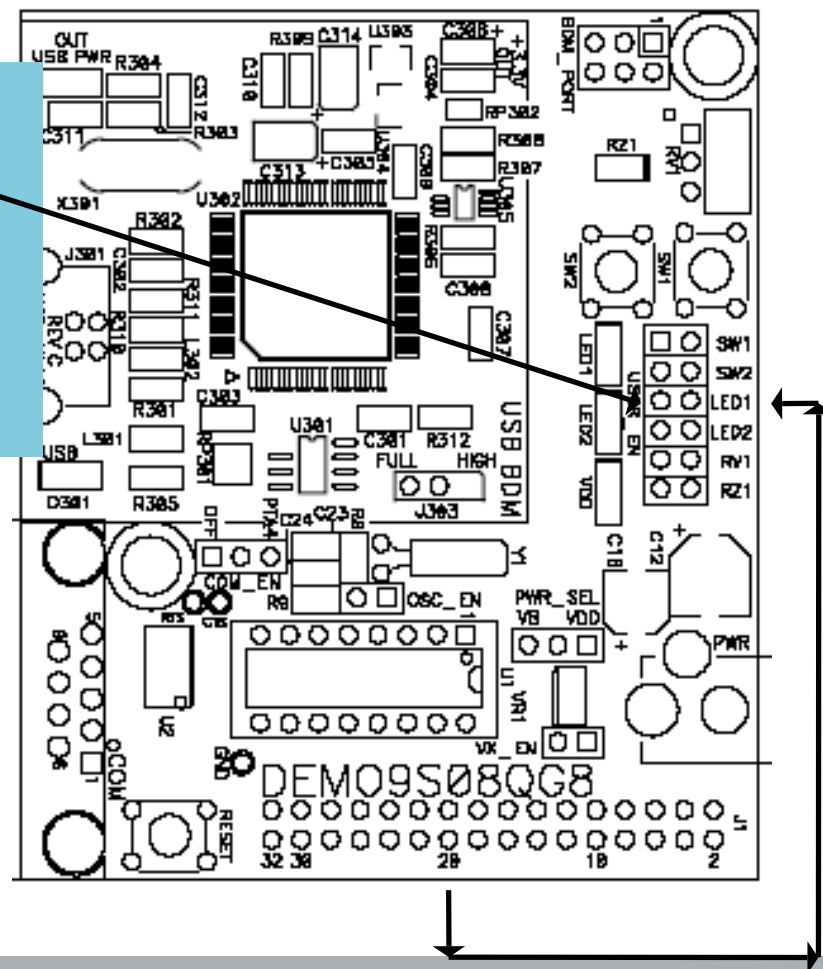
- ▶ Add the Debouce code in KBI ISR
- ▶ Drop and Drag the NegVal to toggle LED1
- ▶ Hit Debug and Run, Press SW2 or Cover up RZ1 to test toggle LED1

LAB3 – ADC & PWM

- ▶ Initialize ADC Channel 0 (PORTA0 –Potentiometer)
- ▶ Initialize PWM for 1mSec period with 64uSec initial on Pulse.
- ▶ Use ADC reading to reconfigure PWM ratio (duty cycle).
- ▶ Re-route PWM output to an LED1 to show dimming effect.

LAB3 – ADC & PWM

- ▶ Remove Jumper on LED1
- ▶ Jumper a wire between J1(23) and LED1(2)
LED1(2) is the pin closer to the edge of the board



LAB3 – ADC & PWM

The screenshot shows the Freescale CodeWarrior IDE interface. The main window displays a project named "BlinkingLed1.mcp" for a target CPU [Cpu:MC9S08QG8CPB]. The Bean Selector window is open, showing a tree view of beans. The "ADC" bean is selected under the "Converter" directory. A "Confirm" dialog box is displayed, asking: "Do you want to enable the bean in all configurations? Select 'No' if you want to use the bean in current configuration only." The dialog has "Yes" and "No" buttons, and a checkbox for "Do not ask again". The Bean Inspector window on the right shows the properties for the selected ADC bean, including "Enable", "Disable", "Start", "Stop", "Measure", and "GetCharValue8".

► From the “Bean Selector” window, expand the “Converter” directory and select “ADC” bean, and click “Yes” to confirm

LAB3 – ADC & PWM

► In the “Bean Inspector”, select the “Properties” tab and click the “Conversion time” box

► Enter a “5.750” uSec value and click “OK”

The screenshot shows the 'Bean Inspector AD1:ADC' window. The 'A/D channels' section is expanded to show 'Channel0'. The 'Conversion time' property is highlighted, and a dialog box is open for its configuration. The dialog box contains the following fields:

- Requested conv. time: 5.750
- unit: μs
- Error allowed: 5
- unit: %
- Units: μs (selected), ms, sec, Xtal ticks

The background window shows the following configuration for 'Channel0':

Property	Value
A/D channel (pin)	PTA0_KBIP0_TPMCH0_ADPC
A/D channel (pin) signal	PTA0_KBIF
A/D prescaler	ADCmode1
A/D resolution	Autoselect
ADC clock source in high speed mode	Auto selected prescaler
Conversion time	5.750 μs
Low-power mode	Disabled
Sample time	short
Automatic Compare	Disabled
Internal trigger	Disabled
Number of conversions	1
Initialization	Enabled in init. code

LAB3 – ADC & PWM

► In the “Bean Inspector”, select the “Properties” tab and toggle the “Start” method to indicate “generate code”

The screenshot shows the 'Bean Inspector AD1:ADC' window with the 'Properties' tab selected. The table below shows the configuration for various methods:

Method	generate code	generate code
<input checked="" type="checkbox"/> Enable	don't generate code	<input type="checkbox"/>
<input checked="" type="checkbox"/> Disable	don't generate code	<input type="checkbox"/>
<input checked="" type="checkbox"/> EnableEvent	don't generate code	<input type="checkbox"/>
<input checked="" type="checkbox"/> DisableEvent	don't generate code	<input type="checkbox"/>
<input checked="" type="checkbox"/> Start	generate code	<input type="checkbox"/>
<input checked="" type="checkbox"/> Stop	don't generate code	<input type="checkbox"/>

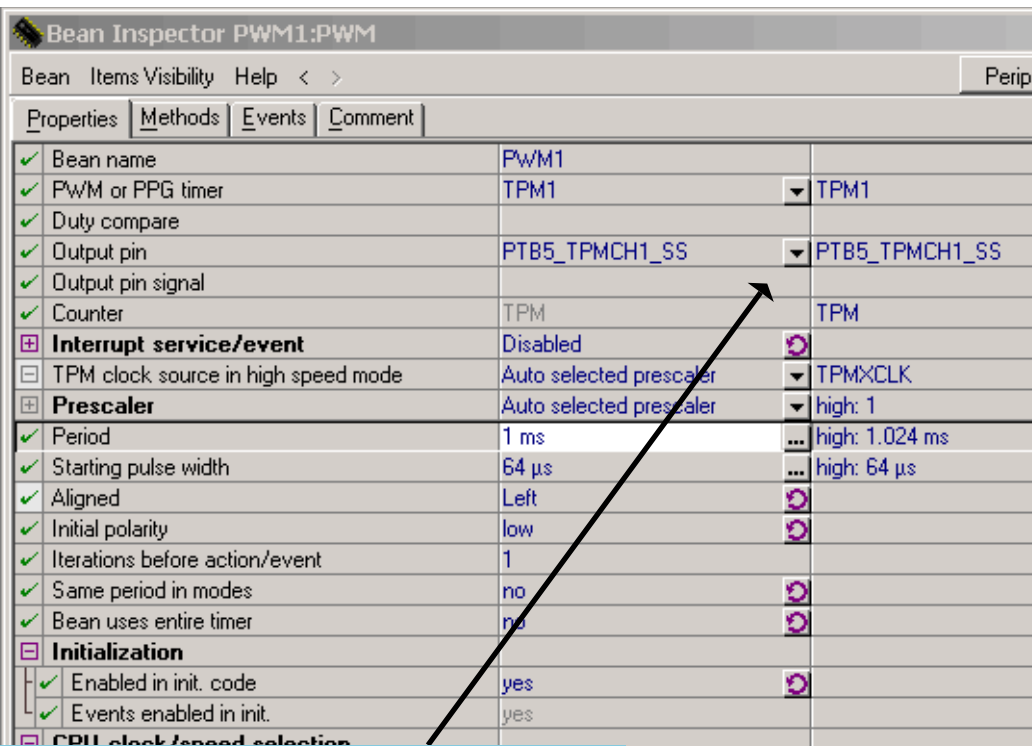
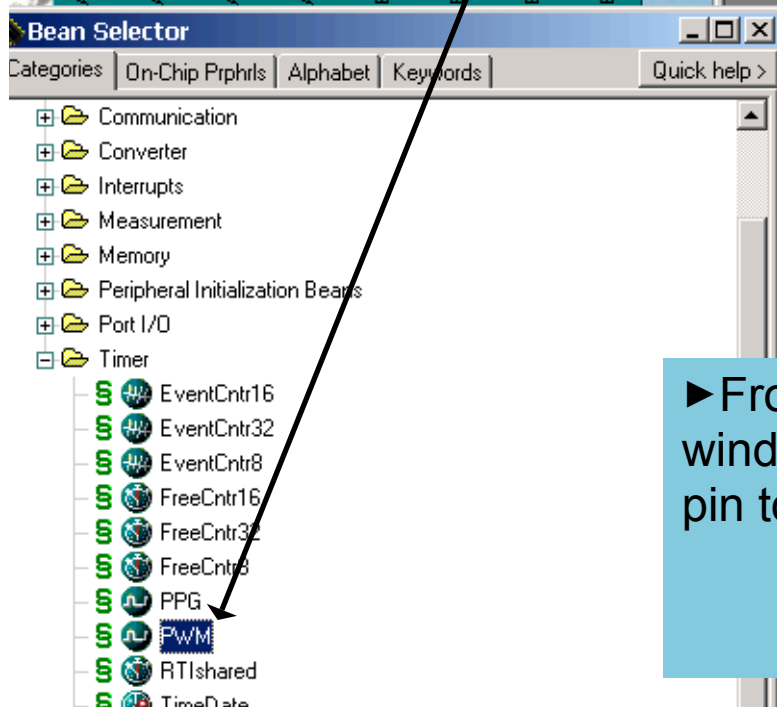
Channels. When each measurement on all channels has finished the [OnEnd](#) event may be invoked. This method is not available if the [interrupt](#) continuous mode. Note: If time of measurement is too short and the instruction clock is too slow then the conversion complete interrupt and its

speed mode

EnableAsynchroClock don't generate code

LAB3 – ADC & PWM

► From the “Bean Selector” window, expand the “Timer” directory and select “PWM” bean and click “Yes” to confirm



► From the “Bean Inspector” window, Change the PWM output pin to “PTB5_TPMC1_SS”

LAB3 – ADC & PWM

► From the “Bean Inspector” window, click on the “Period” box.

► Enter “1”ms and click “OK”

► From the “Bean Inspector” window, click on the “Starting Pulse Width” box.

► Enter “128” or “64” uSec and click “OK”

The screenshot shows the Bean Inspector window for PWM1:PWM. The configuration table is as follows:

✓ Counter	TPM	TPM
⊕ Interrupt service/event	Disabled	TPM
☐ TPM clock source in high speed mode	Auto selected prescaler	TPM
☐ Prescaler	Auto selected prescaler	high
✓ Period		...
! Starting pulse width		Unassigned
✓ Aligned	Left	TPM
✓ Initial polarity	low	TPM

The 'Init. value' dialog box for the Period property is open, showing:

- Requested value: 1
- unit: ms
- Error allowed: 5
- unit: %
- Units: With value
- Speed mode: high
- Adjusted value: 1.024 ms
- Prescaler: 1
- Error: 2.4%

The screenshot shows the Bean Inspector window for PWM1:PWM. The configuration table is as follows:

☐ TPM clock source in high speed mode	Auto selected prescaler	TPMXCLK
☐ Prescaler	Auto selected prescaler	high: not set
✓ Period	1 ms	high: 1.024 ms
✓ Starting pulse width	128 μs	high: 128 μs
✓ Aligned	Left	TPM
✓ Initial polarity	low	TPM
✓ Iterations before action/event	1	TPM
✓ Same period in modes	no	TPM
✓ Bean uses entire timer	no	TPM
⊕ Initialization		TPM

The 'Init. value' dialog box for the Starting pulse width property is open, showing:

- Requested value: 128
- unit: μs
- Error allowed: 5
- unit: %
- Units: With value
- Speed mode: high
- Adjusted value: 128 μs
- Prescaler: 1
- Error: 2.4%

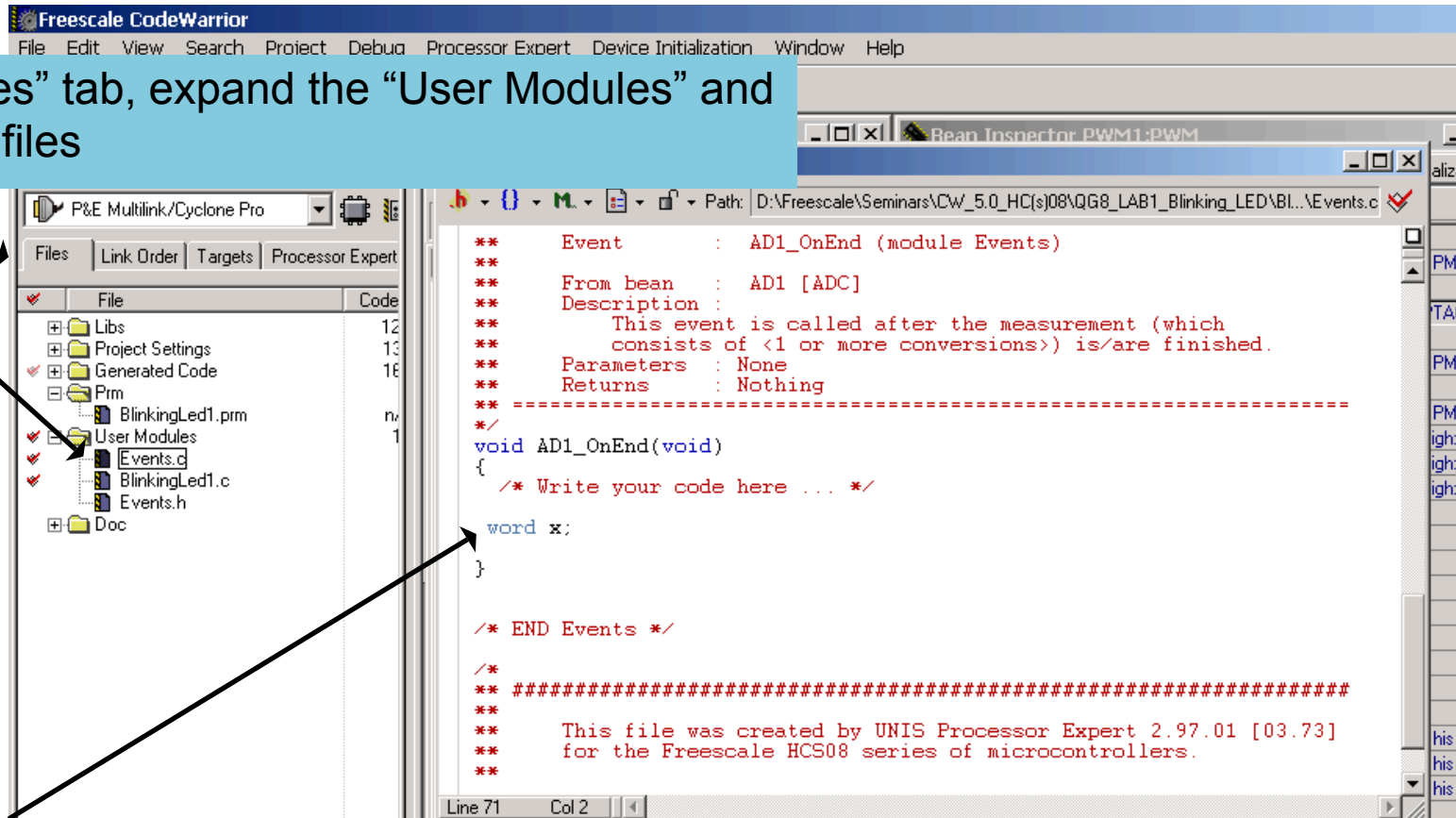
LAB3 – ADC & PWM

From the “Processor Expert” drop down menu, click on “Generate code”

Bean	Items	Visibility	Help
Bean name	PWM1		
PWM or PPG timer	TPM0		TPM0
Duty compare			
Output pin	PTA0_KBIP0_TPMCH0		PTA0_KBIPC
Output pin signal			
Counter	TPM		TPM
Interrupt service/event	Disabled		
TPM clock source in high speed mode	Auto selected prescaler		TPMXCLK
Prescaler	Auto selected prescaler		high: 1
Period	1 ms		high: 1.024 m
Starting pulse width	128 μs		high: 128 μs
Aligned	Left		
Initial polarity	low		
Iterations before action/event	1		
Same period in modes	no		
Bean uses entire timer	no		
Initialization			
Enabled in init. code	yes		
Events enabled in init.	yes		
CPU clock/speed selection			
High speed mode	This bean enabled		This bean is i
Low speed mode	This bean disabled		This bean is i
Slow speed mode	This bean disabled		This bean is i

LAB3 – ADC & PWM

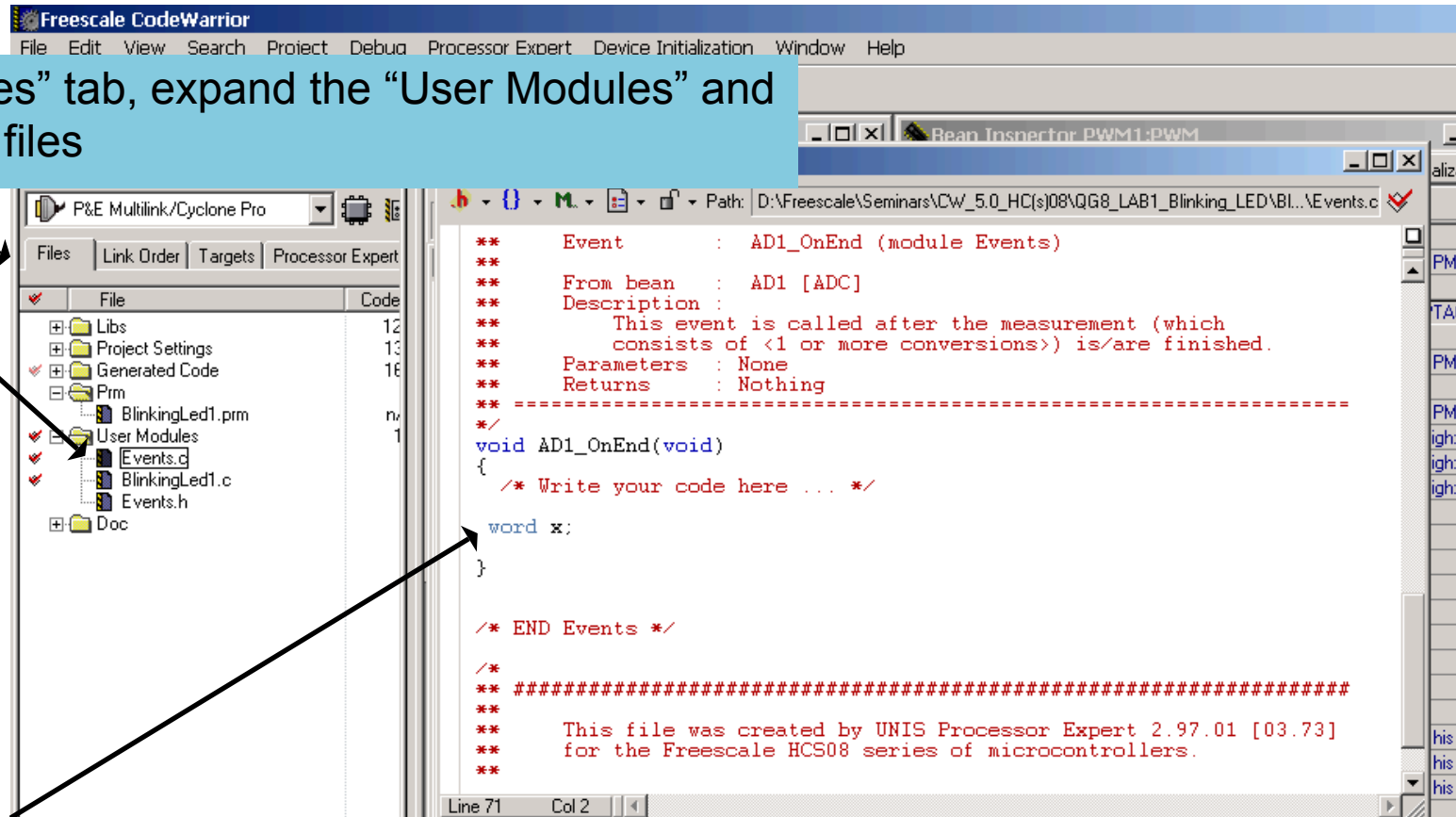
► From the “Files” tab, expand the “User Modules” and open “Event.c” files



► Scroll down to the “void AD1_OnEnd(void)”
► Type “word x;” in the /*Write your code here*/

LAB3 – ADC & PWM

► From the “Files” tab, expand the “User Modules” and open “Event.c” files



► Scroll down to the “void AD1_OnEnd(void)”
► Type “word x;” in the /*Write your code here*/

LAB3 – ADC & PWM

► Select the “Processor Expert” tab

The screenshot shows the Freescale CodeWarrior IDE. The main window is titled "BlinkingLed1.mcp" and has the "Processor Expert" tab selected. The left pane shows a tree view of components under "Beans", with "AD1:ADC" expanded. The right pane shows the "Events.c" file with the following code:

```
/** Event      : AD1_OnEnd (module Events)
** From bean   : AD1 [ADC]
** Description : This event is called
**              consists of <1 or more
** Parameters  : None
** Returns     : Nothing
** -----
*/
void AD1_OnEnd(void)
{
    /* Write your code here ... */

    word x;
    AD1_GetValue16(&x);
}

/* END Events */
```

Below the code editor, there is a "Bean Inspector" window showing a list of beans: PWM, RTIshared, TimeDate, and TimerInt. The "PWM" bean is selected, and its properties are shown: "Low speed mode" and "Slow speed mode", both of which are disabled.

► Expand the “ADC” bean
► Drop and Drag the
“GetValue16” function call
and place it below “word x;”

LAB3 – ADC & PWM

The screenshot shows the Freescale CodeWarrior IDE interface. The 'Debug' menu is highlighted in the top menu bar. The 'Connection Manager' dialog is open, showing the 'Interface' dropdown set to 'USB HCS08/HCS12 Multilink - USB Port'. The 'Erase and Program Flash?' dialog is also open, with the 'Yes' button selected. The 'S08 Processor - Autodetect' dialog is visible in the background, showing the 'Connect' button. The 'Events.c' window shows the following code:

```
*** Event      : AD1_OnEnd (module Events)
*** From bean  : AD1 [ADC]
```

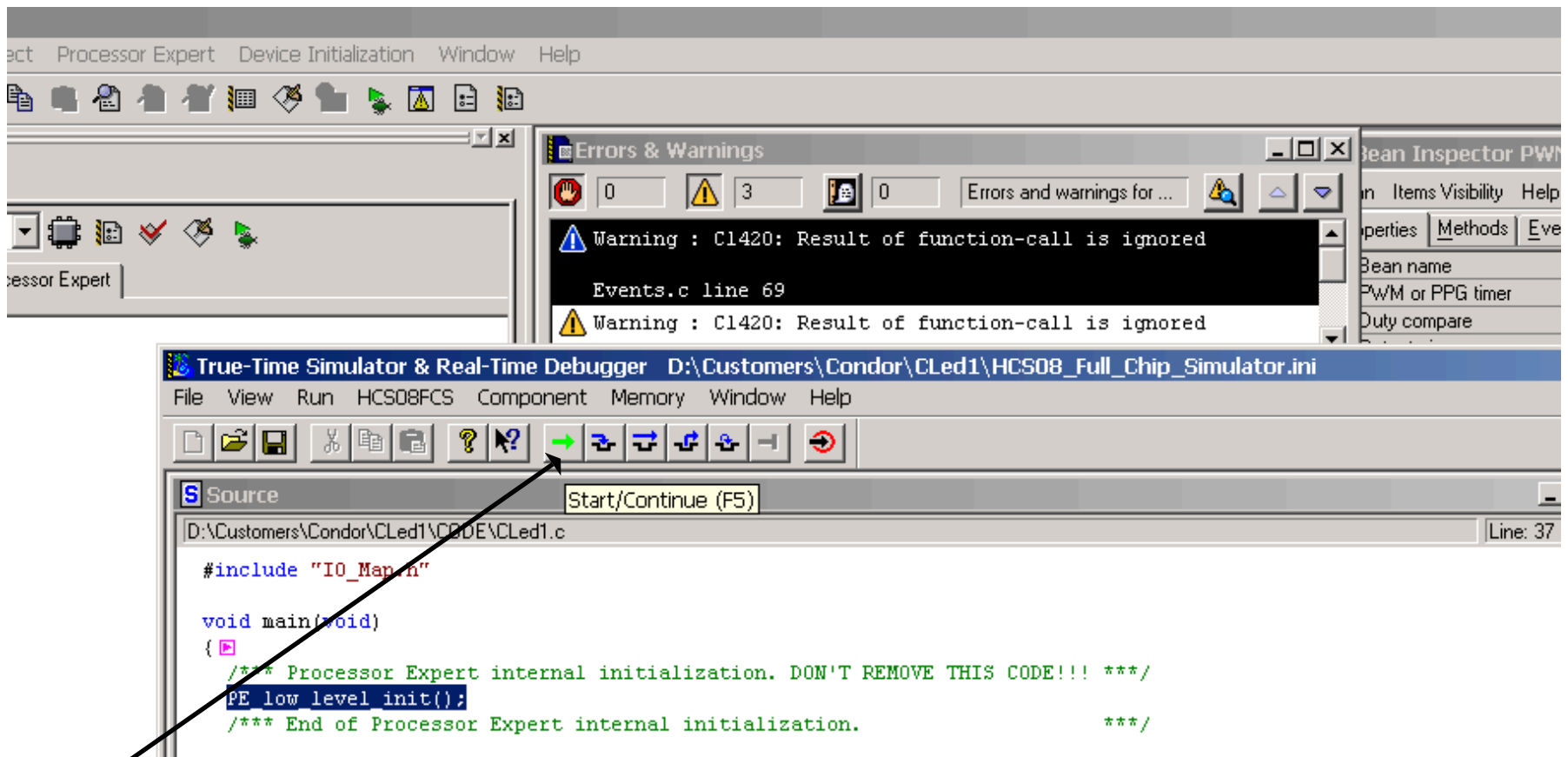
The 'True-Time Simulator & Real-Time Debugger' window is also visible, showing the 'Connect' button.

▶ Click on “Debug” to compile.

▶ Click on “Connect” to engage debugger

▶ Click on “Yes” to download new image

LAB3 – ADC & PWM



- ▶ Hit “Start” icon to run the program.
- ▶ Roll the Thumb wheels POT on the Demo board to see the LED1 dimmed or brighten

LAB4 – SCI Communication

Create a brand new project and set bus clk to 8Mhz.

Initialize SCI(Serial) Comm to 9600 Baud rate

Set Interrupt upon received of Character.

Set Hyperterminal on PC to 9600 baud.

Write application to echo received character back to PC

LAB4 – SCI Communication

► Change the Internal Osc Frequency to 31.25Khz.

► Change the Internal BusClock to 8.0 Mhz

Bean Inspector Cpu:MC9S08QG8CPB

Bean	Items	Visibility	Help	<	>
Properties	Methods	Events	Build options	Used	Comment
✓	Bean name		Cpu		
✓	CPU type		MC9S08QG8CPB		
+	Clock settings				
+	Internal clock				
✓	Internal oscillator frequency [kHz]		31.25		31.25 kHz
+	Initialize trim value		yes		
✓	Trim value address		FFAF		
✓	Fine trim value address		FFAE		
+	External clock		Disabled		
+	Low-power modes settings				
+	STOP instruction enabled		no		
✓	Initialization interrupt priority		interrupts enabled		1
+	Internal peripherals				This property enables the STOP instruction.
+	CPU interrupts				
+	Enabled speed modes				
+	High speed mode		Enabled		
✓	High speed clock		Internal Clock		31.25 kHz
+	Bus freq. divider		Auto select		1
✓	Internal bus clock		8.0		8.0 MHz
✓	Fixed freq. clock clk src.		Divided FLL reference clock		
✓	Fixed frequency clock		0.015625		0.015625 MHz
+	FLL mode		Engaged		

LAB4 – SCI Communication

► In the “Bean Selector” select “Asynchronous Serial” bean.

► Enable the “Interrupt”

► Set “Baud rate” to 9600

The screenshot displays the IDE interface for configuring an Asynchronous Serial (SCI) bean. The Bean Selector on the left shows the hierarchy: CPU > CPU Internal Peripherals > Communication > AsynchroSerial. The Bean Inspector on the right shows the configuration for the selected bean, AS1. The configuration is as follows:

Property	Value
Bean name	AS1
Channel	SCI
Interrupt service/event	Enabled
Settings	
Parity	none
Width	8 bits
Stop bit	1
SCI output mode	Normal
Receiver	Enabled
RxD	PTB0_KBIP4_RxD_ADP4
RxD pin signal	PTB0_KBIP4
Transmitter	Enabled
TxD	PTB1_KBIP5_TxD_ADP5
TxD pin signal	PTB1_KBIP5
Baud rate	9600 baud
Break signal	Disabled
Wakeup condition	Idle line wakeup
Transmitter output	Not inverted
Stop in wait mode	no

The Timing - Baud rate section shows the following settings:

Runtime setting	Init. value
Runtime setting type: fixed value	Requested baud rate: 9600 baud
	Error allowed: 1.5 %
	Speed mode Adjusted value: 9615.385 baud Prescaler: 52

LAB4 – SCI Communication

► In the “Events” Tab, generate code only for “OnRXChar”

► Select Don’t Generate Code” for everything else such as “OnError” or “OnTxChar”

► Click “Generate Code”

The screenshot displays the IDE interface with three main windows:

- Target CPU [Cpu:MC9S08QG8CPB]**: Shows a background image of a desert landscape.
- Bean Inspector AS1:AsynchroSerial**: Shows the 'Events' tab with the following configuration:

Event	Generate Code
Event module name	Events
BeforeNewSpeed	don't generate code
AfterNewSpeed	don't generate code
OnError	don't generate code
OnRXChar	generate code
Event procedure name	AS1_OnRXChar
Priority	same as interrupt
OnRXCharExt	don't generate code
OnTxChar	don't generate code
OnFullRxBuf	don't generate code
OnFreeTxBuf	don't generate code
OnBreak	don't generate code
OnIdle	don't generate code
OnTxComplete	don't generate code
- Bean Selector**: Shows a tree view with 'AsynchroSerial' selected under 'Communication'.
- Freescall CodeWarrior**: Shows a context menu with 'Generate Code 'SCInSPI_Com1.mcp'' selected.

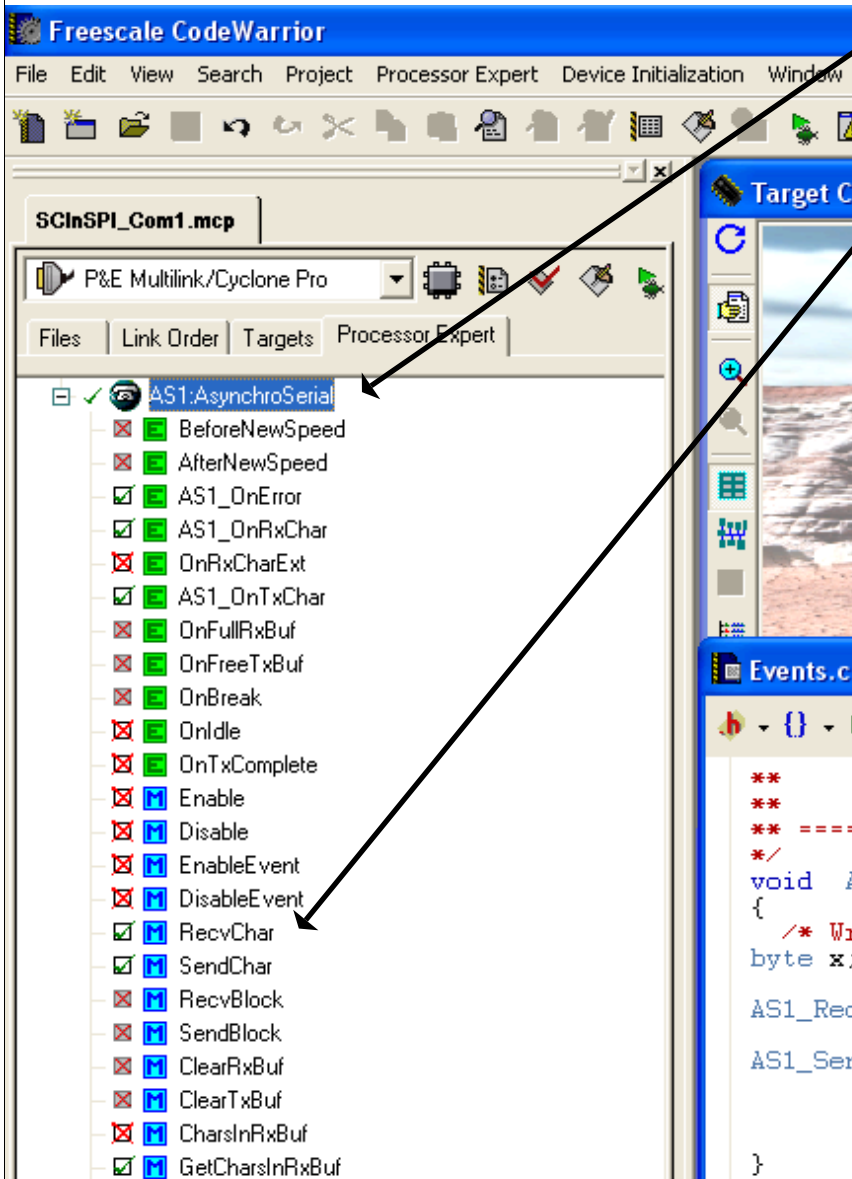
LAB4 – SCI Communication

► Under “Processor Expert” Tab, expand the “AS1 AsynchronousSerial” bean

► Drop and Drag “RecvChar” and “SendChar” bean to Event.c and add the modification below:

► Byte x;
AS1_RecvChar(&x);
AS1_SendChar(x+2);

Hit Debug and Run



```
Events.c
Path: D:\Freescale\Seminars\CW_5.0_HC(s)08\SCInSF

** Parameters : None
** Returns : Nothing
** -----
*/
void AS1_OnRxChar(void)
{
    /* Write your code here ... */
    byte x;

    AS1_RecvChar(&x);

    AS1_SendChar(x+2);
}
```

LAB4 – SCI Communication

► Setting up Hyperterminal for 9600 Baud rate as below:

The image displays three overlapping windows from the Hyperterminal application, illustrating the configuration for a 9600 Baud rate connection.

- COM1 Properties (Port Settings):**
 - Bits per second: 9600
 - Data bits: 8
 - Parity: None
 - Stop bits: 1
 - Flow control: None
- DFAE_9600_Com1 Properties (Settings):**
 - Function, arrow, and ctrl keys act as: Terminal keys, Windows keys
 - Backspace key sends: Ctrl+H, Del, Ctrl+H, Space, Ctrl+H
 - Emulation: VT100 (selected from a dropdown menu)
- ASCII Setup:**
 - ASCII Sending:
 - Send line ends with line feeds
 - Echo typed characters locally
 - Line delay: 0 milliseconds
 - Character delay: 0 milliseconds
 - ASCII Receiving:
 - Append line feeds to incoming line ends
 - Force incoming data to 7-bit ASCII
 - Wrap lines that exceed terminal width

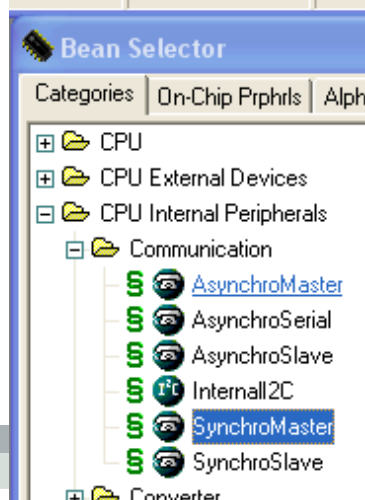
LAB5 – SPI Communication

- ▶ To send communication from PC to PC via an QG8 SCI/SPI converter
- ▶ Continuation with LAB4 (SCI Comm).
- ▶ Initialize SPI for MasterMode
- ▶ Initialize SPI for Slave Mode
- ▶ Write SCI/SPI Converter code
- ▶ Use Hyperterminal to test and application.

LAB5 – SPI Communication(Master)

SPI Master Configuration

- ▶ In the “Bean Selector” window, select Synchronous Master.
- ▶ Enable “Slave Select pin” and set Direction to Output
- ▶ Set “Clock rate” to 4Mhz



Property	Value
Bean name	SM1
Channel	SPI
Interrupt service/event	Enabled
Interrupt	Vspi
Interrupt from input	
Interrupt input priority	medium priority
Interrupt from output	
Interrupt output priority	medium priority
Input buffer size	0
Output buffer size	0
Settings	
Bidirectional mode	Disabled
Input pin	Enabled
Pin	PTB4_MISO
Pin signal	
Output pin	Enabled
Pin	PTB3_KBIP7_MOSI_ADP7
Pin signal	
Clock pin	
Pin	PTB2_KBIP6_SPSCK_ADP6
Pin signal	
Slave select pin	Enabled
Pin	PTB5_TPMCH1_SS
Pin signal	
Direction	Output
Clock edge	rising or falling edge
Shift clock rate	4 MHz
Empty character	0
Ignore empty char.	no
Send MSB first	no
Shift clock idle polarity	Low

LAB5 – SPI Communication(Master)

The screenshot shows the Processor Expert IDE with the project 'SCInSPI_Com1.mcp'. The 'Processor Expert' tab is active, displaying a tree of components. The 'SM1:SynchroMaster' component is selected, and its 'SendChar' property is highlighted. The 'Events.c' source code is open in the editor, showing the implementation of the 'AS1_OnRxChar' event handler. The code includes headers for 'Cpu.h' and 'Events.h', and defines the 'AS1_OnRxChar' event handler function. The function calls 'AS1_RecvChar' to receive a character and 'SM1_SendChar' to send it to the SPI master.

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

/*
** -----
**      Event      : AS1_OnRxChar
**      From bean   : AS1 [Asynchro
**      Description :
**                  This event is called aft
**                  received.
**                  DMA mode:
**                  If DMA controller is ava
**                  the receiver is configur
**                  this event is disabled.
**                  used in DMA mode.
**      Parameters  : None
**      Returns     : Nothing
** -----
*/

void AS1_OnRxChar(void)
{
    /* Write your code here ... */
    byte x;

    AS1_RecvChar(&x);

    //AS1_SendChar(x+2);
    SM1_SendChar(x+2);
}
```

► Under “Processor Expert” Tab, expand the “SM1:SynchronousMaster” bean

► Drop and Drag “SendChar” and modify as below:
SM1_SendChar(x+2);

► Hit Debug and Run

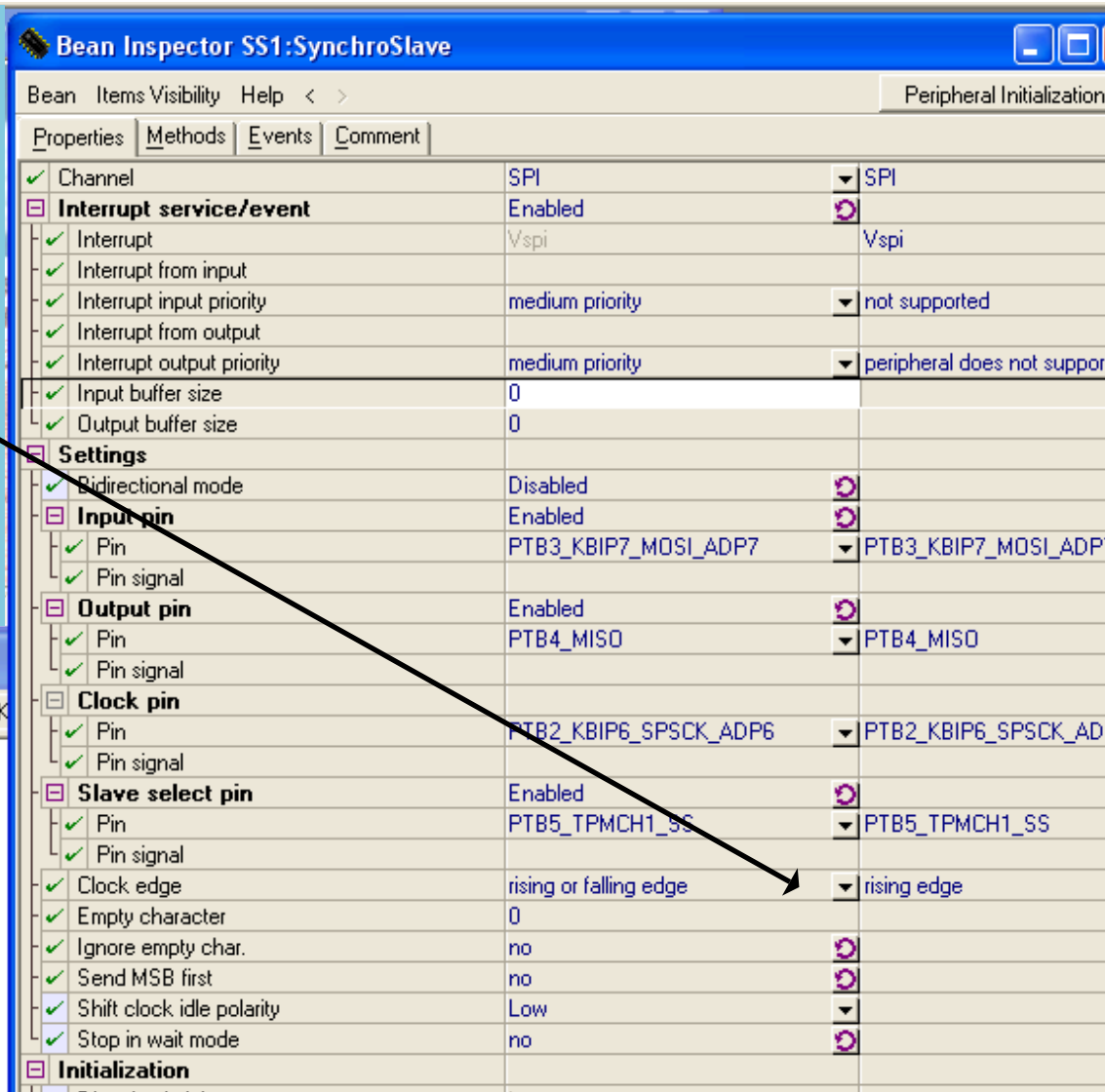
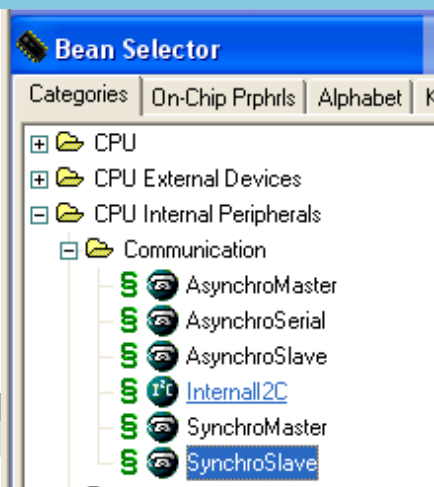
► Use Hyper-terminal to test Communication



LAB5 – SPI Communication(Slave)

SPI Slave Configuration

- ▶ In the “Bean Selector” window, select SynchronousSlave.
- ▶ Set “Clock Edge” to rising edge to match up with the Master mode.
- ▶ Generate Code



LAB5 – SPI Communication(Slave)

▶ In the Event.c, add the code as shown below:

- ▶ Byte y;
- ▶ SS1_RecvChar(&y);
- ▶ AS1_SendChar(y-2);

▶ or Drop&drag the routines from their respected bean as shown by the arrows

▶ Debug and Run and Test with Hyper-Terminal

The screenshot displays the P&E Multilink/Cyclone Pro IDE interface. On the left, a tree view shows the project structure with 'AS1:AsynchroSerial' and 'SS1:SynchroSlave' expanded. Under 'SS1:SynchroSlave', various event routines are listed with checkboxes and icons. On the right, another tree view shows 'AS1:AsynchroSerial' with its own set of event routines. In the center, the 'Events.c' file is open, showing a code snippet for the 'SS1_OnRxChar' event handler. The code includes comments and function calls: 'SS1_RecvChar(&y);' and 'AS1_SendChar(y-2);'. Two black arrows point from the text in the blue box to the 'RecvChar' and 'SendChar' routines in the left tree view, and to the corresponding function calls in the code editor.

LAB6 – I2C Communication

- ▶ Modify HW to add 4.7K resistors to SCL and SDA signals
- ▶ Initialize I2C HW
- ▶ Write I2C application

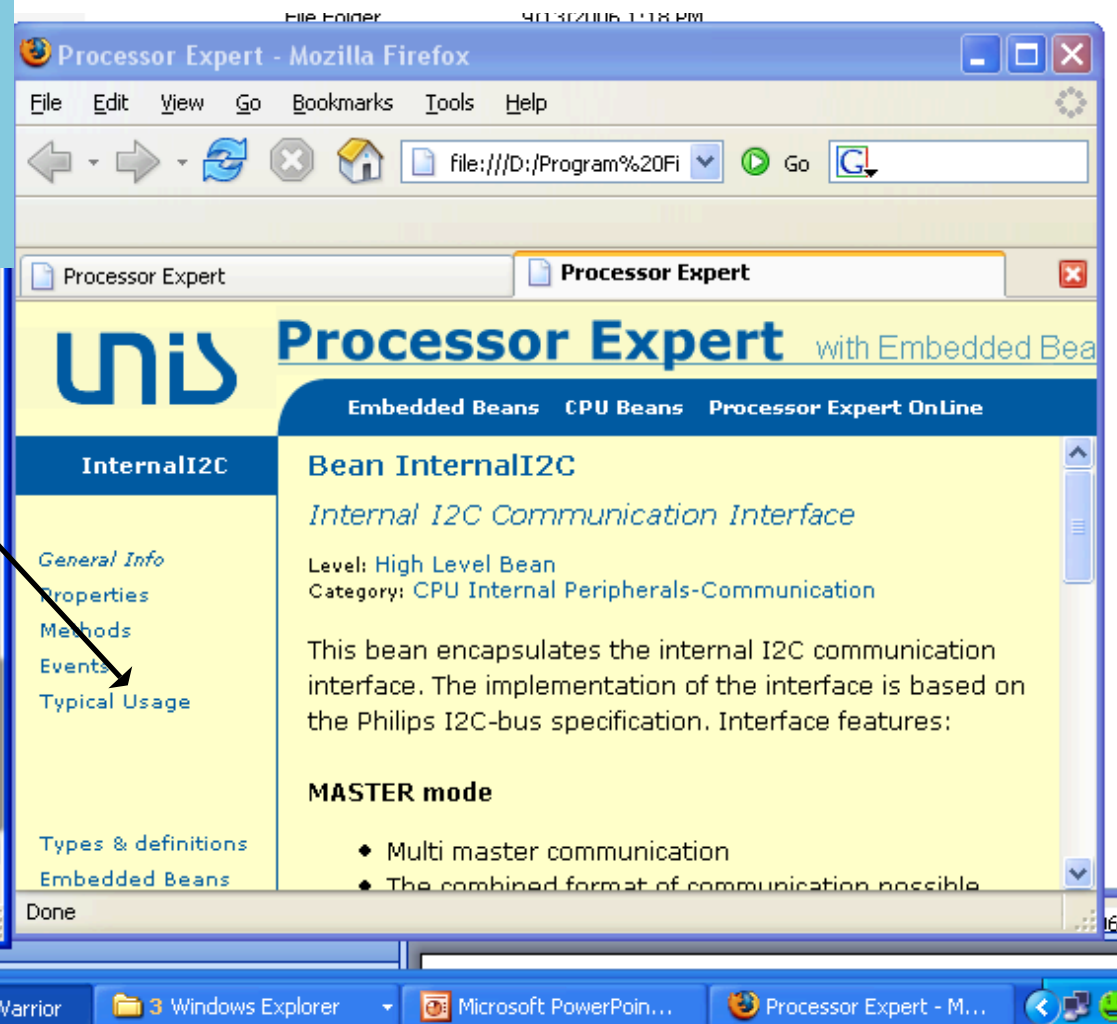
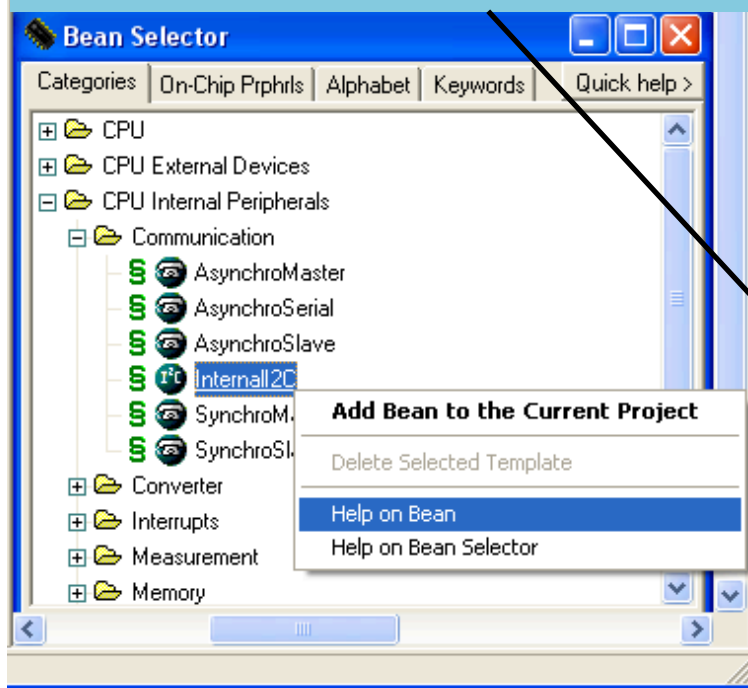
LAB6 – I2C Communication

► Modify I2C Bean Configuration as below:

Property	Value	Additional Info
Bean name	I2C1	
I2C channel	IIC	IIC
Mode selection	MASTER - SLAVE	
Interrupt service/event	Enabled	
Interrupt	Viic	Viic
Interrupt priority	medium priority	not supported
Buffers for SLAVE mode	Enabled	
Input buffer size	8	
Output buffer size	8	
MASTER mode	Enabled	
Polling trials	2000	
Automatic stop condition	yes	
Initialization		
Slave address init	8	
SLAVE mode	Enabled	
Slave address	0	
General call address	Disabled	
Empty character	0	
Data and Clock		
SDA pin	PTB6_SDA_EXTAL	PTB6_SDA_EXTAL
SDA pin signal		
SCL pin	PTB7_SCL_EXTAL	PTB7_SCL_EXTAL
SCL pin signal		
Internal frequency (multiplier factor)	8 MHz	high: 8 MHz
Bits 0-2 of Frequency divider register	000	
Bits 3-5 of Frequency divider register	000	
SCL frequency	400 kHz	high: 400 kHz
SDA Hold	0.875 us	high: 0.875 us
Initialization		
Enabled in init code	yes	
Events enabled in init	yes	

LAB6 – I2C Communication

- ▶ Right hand click on “Internal I2c” bean and select Help on Bean from Drop down menu.
- ▶ Select Typical Usage



LAB6 – I2C Communication

Processor Expert - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

file:///D:/Program%20Files/Freescale/CodeWarrior%20for%20HC08%20...

Processor Expert Processor Expert

UNI Processor Expert with Embedded Beans

Embedded Beans CPU Beans Processor Expert OnLine

- Method code in the bean
- Interrupt handler code in the bean
- User application code in the event module, typically called from the
- I2C peripheral module sends or receives a byte

← Call of a method or occurrence of an event

- - → Return from a method, event or interrupt

← Interrupt

Typical settings and usage of InternalI2C bean

- ◆ (1) Sending data in the MASTER mode
- ◆ (2) Receiving data in the MASTER mode
- ◆ (3) Sending data in the SLAVE mode
- ◆ (4) Receiving data in the SLAVE mode
- ◆ (5) Sending data in the MASTER mode without interrupts
- ◆ (6) Receiving data in the MASTER mode without interrupts

HomeWork:

Complete an I2C application using the Typical Usage Help Menu

Remember to add 4.7k resistors to SDA and SCL line before calling He My FAE.

CodeWarriors & Processor Expert Seminar

Thank You