

# 56800/E SCI Hands-On Exercise

56800/E Training



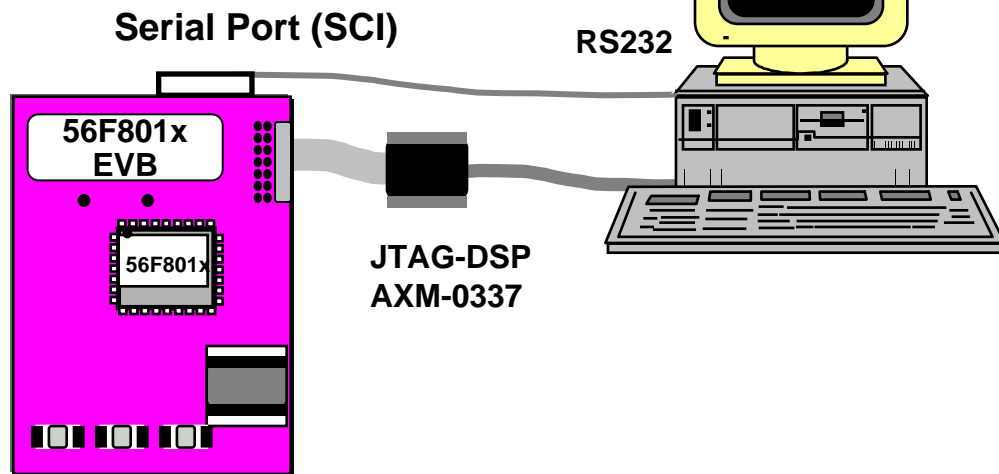
Slide 1

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.  
© Freescale Semiconductor, Inc. 2004

# Task Description

Develop a monitor application that uses SCI in messages between PC and DSC56F801x EVB.

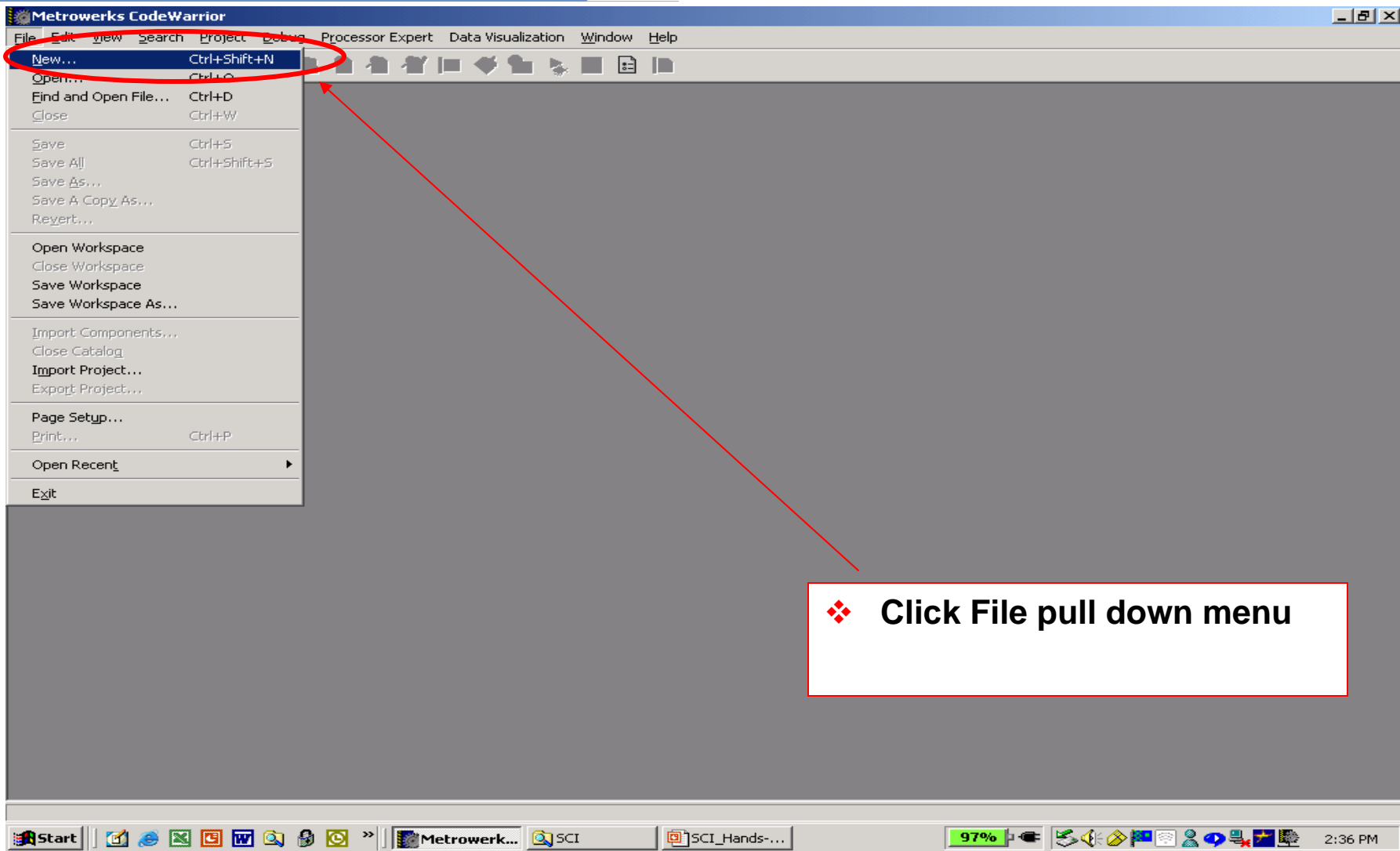
HyperTerminal (Console) for  
 CAN Data Display  
 via RS232,  
 Bits/sec : 9600  
 No parity : none  
 Stop bit : 1  
 Hardware : None

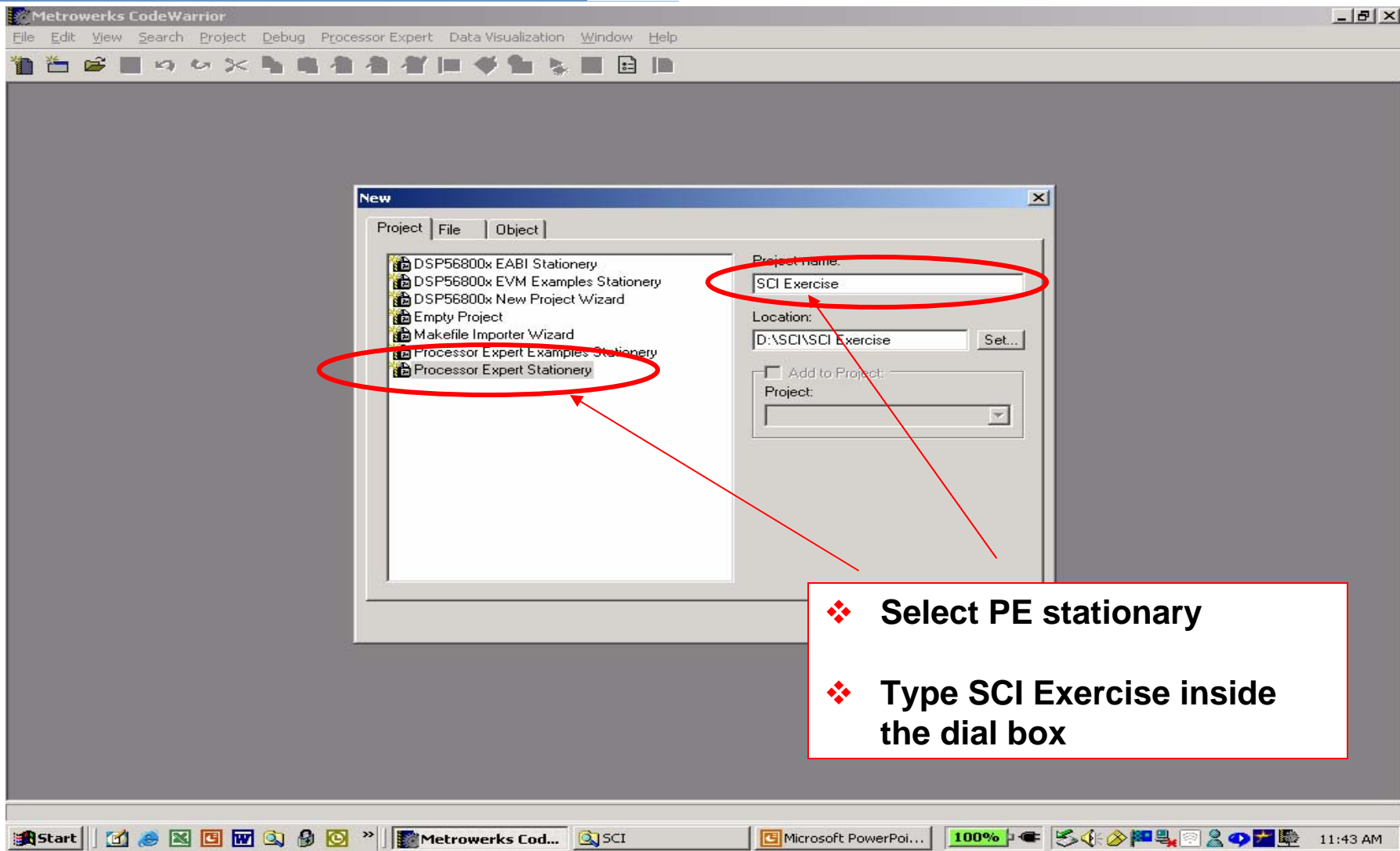


# Approach

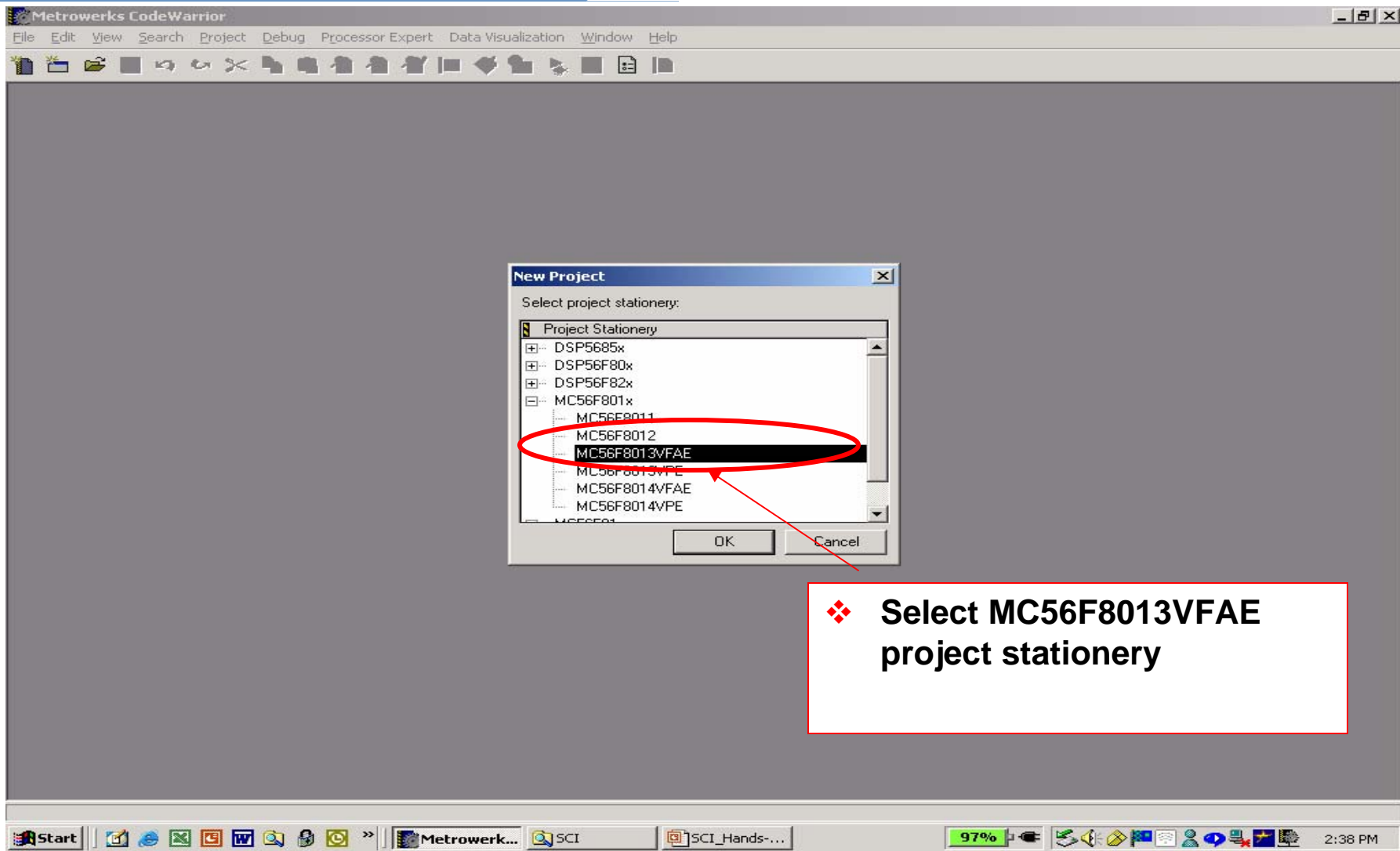
- ❖ **Use Processor Expert Beans to implement Target System application**
  - ✓ **AsynchroSerial**
  - ✓ **Download and Execute on 56F801x Demo Board**

# 56800/E Training





- ❖ **Select PE stationery**
- ❖ **Type SCI Exercise inside the dial box**



❖ **Select MC56F8013VFAE project stationery**

The screenshot shows the Metrowerks CodeWarrior IDE interface. On the left, the 'SCI Exercise.mcp' project is open, showing a file tree with a 'support' folder. The main workspace displays a 3D model of a CPU chip with various pins and connections. Below the workspace, the 'Bean Selector' window is open, showing a tree view of components. The 'CPU' component is selected, and a red arrow points from the 'CPU' entry in the tree to the 3D model. A red-bordered box highlights the text: ❖ PE template is shown on.

The screenshot shows the Metrowerks CodeWarrior IDE. On the left, the project tree shows 'AS1:AsynchroSerial' selected under 'Beans'. In the center, a hardware diagram of the CPU is shown with various pins connected. On the right, the 'Bean Inspector AS1:AsynchroSerial' window is open, displaying the 'Settings' tab. A red circle highlights the 'Baud rate' dial box, which is currently set to 'Unassigned'. Another red circle highlights the 'AsynchroSerial' bean in the 'Bean Selector' window.

- ❖ Kick AsynchroSerial Bean under Communication PE menu
- ❖ Then, kick Baud rate dial box under the properties of Bean Inspector AS1:AsynchroSerial



**❖ Type 9600 baud rate inside the dial box**

The screenshot shows the Metrowerks CodeWarrior IDE interface. The 'Processor Expert' menu is open, and the 'Generate Code 'SCI Exercise.mcp'' option is highlighted with a red circle. A red arrow points from this option to a callout box containing the following text:

❖ Select Generate Code 'SCI Exercise.mcp' under PE pull down menu to generate the code

The interface also shows the 'Bean Inspector AS1:AsynchroSerial' window on the right, displaying properties for the SCI peripheral, and the 'Bean Selector' window at the bottom, showing a tree view of components under the 'Communication' category.

The screenshot displays the Metrowerks CodeWarrior IDE. The main window shows the 'Bean Inspector AS1:AsynchroSerial' configuration for a target CPU (Cpu:56F8013VFAE). The configuration is set to 'Peripheral Initialization' and includes the following settings:

Property	Value
Bean name	AS1
Channel	SCI
Interrupt service/event	Disabled
<b>Settings</b>	
Parity	none
Width	8 bits
Stop bit	1
SCI output mode	Normal
LIN slave mode	Disabled
<b>Receiver</b>	Enabled
RxD	GPIOB6_RX
RxD pin signal	GPIOB6
<b>Transmitter</b>	Enabled
TxD	GPIOB7_TX
TxD pin signal	GPIOB7
Baud rate	9600 baud
Break signal	Disabled
Wakeup condition	Idle line wak
Transmitter output	Not inverted
Stop in wait mode	no
<b>Initialization</b>	
Enabled in init. code	yes
Events enabled in init.	yes
<b>CPU clock/speed selection</b>	
High speed mode	This bean er
Low speed mode	This bean di
Slow speed mode	This bean di

The 'Code Generation' dialog in the foreground shows the following information:

- Project: SCI\_Exercise
- Module: Cpu
- Current line: 624
- Total lines: 6469
- Errors: 0
- Warnings: 0
- Hints: 0

The screenshot shows the Metrowerks CodeWarrior IDE interface. On the left, the project tree for 'SCI\_Exercise.mcp' is visible, with 'SCI\_Exercise.c:main' highlighted in blue and circled in red. A red arrow points from this circle to a callout box. The main editor window displays the source code for 'SCI\_Exercise.c', which includes headers for 'AS1.h', 'PE\_Types.h', 'PE\_Error.h', 'PE\_Const.h', and 'IO\_Map.h'. The code defines a 'main' function that calls 'PE\_low\_level\_init()' and contains a placeholder for user code. A comment at the bottom states: 'This file was created by UNIS Processor Expert 2.96 [03.1] for the Freescale 56800 series of microcontrollers.' The bottom status bar shows the system tray with a 97% battery level and the time 2:45 PM.

❖ Open up the SCI\_Exercise.c:main file to edit user code.



The screenshot shows the Metrowerks CodeWarrior IDE. On the left, the PE Bean configuration tree is visible under 'AS1:AsynchroSerial'. The 'SendChar' bean is selected and circled in red. A red arrow points from this circle to a text box on the right. The main window displays the source code for 'SCI\_Exercise.c', which includes headers for 'AS1.h', 'PE\_Types.h', 'PE\_Error.h', 'PE\_Const.h', and 'IO\_Map.h'. The code defines a 'main' function that calls 'PE\_low\_level\_init()' and contains a loop for sending characters. The bottom status bar shows '97%' CPU usage and the time '2:45 PM'.

**Select PE Bean SendChar for sending out characters**

Metrowerks CodeWarrior

SCI\_Exercise.mcp

smm pROM-xRAM

Files | Link Order | Targets | Processor Expert

- AS1:AsynchroSerial
  - BeforeNewSpeed
  - AfterNewSpeed
  - OnError
  - OnRxChar
  - OnTxChar
  - OnFullRxBuf
  - OnFreeTxBuf
  - OnBreak
  - OnTxComplete
  - Enable
  - Disable
  - EnableEvent
  - DisableEvent
  - RecvChar
  - SendChar**
  - RecvBlock
  - SendBlock
  - ClearRxBuf
  - ClearTxBuf
  - CharsInRxBuf
  - GetCharsInRxBuf
  - CharsInTxBuf
  - GetCharsInTxBuf
  - SetBaudRateMode
  - GetError
  - GetBreak
  - SetBreak
  - TurnTxOn
  - TurnTxOff
  - TurnBxOn

SCI\_Exercise.c

```

#include "AS1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

void main(void)
{
    /*** Processor Expert internal initialization. DON'T REMOVE THIS! ***/
    PE_low_level_init();
    /*** End of Processor Expert internal initialization. ***/

    /* Write your code here */
    for(;;) {AS1_SendChar()}
}

/** END SCI_Exercise */
/**
 * *****
 * This file was created by UNIS Processor Expert 2.96 [03..]
 * for the Freescale 56800 series of microcontrollers.
 * *****
 */
    
```

Line 42 Col 12

Communication

- AsynchroMaster
- AsynchroSerial**
- AsynchroSlave
- FreescaleCAN
- FreescaleH18
- FreescaleSSI
- InternalI2C
- SynchroMaster
- SynchroSlave
- Converter

Events enabled in init.

CPU clock/speed selection

Put PE Bean SendChar to SCI\_Exercise.c:main by Drag-n-Drop Methods.

97%

2:45 PM

The screenshot shows the Metrowerks CodeWarrior IDE. On the left is the Project Explorer showing the 'SCI\_Exercise.mcp' project with various components like 'AS1:AsynchroSerial' and 'AS1:AsynchroSerial' sub-components. The main window displays the source code for 'SCI\_Exercise.c'. The code includes several headers and a 'main' function. A red circle highlights the following code block in the 'main' function:

```

for(;;) {
    AS1_SendChar('H');
    AS1_SendChar('e');
    AS1_SendChar('l');
    AS1_SendChar('l');
    AS1_SendChar('o');
    AS1_SendChar('\r');
    AS1_SendChar('\n');
}

```

A red arrow points from a callout box to this code block. The callout box contains the text:

❖ Made more characters to send out by Copy-n-Paste methods.

The bottom of the IDE shows the Windows taskbar with the Start button, several application icons, and the system tray showing '97%' battery and the time '2:46 PM'.

The screenshot shows the Metrowerks CodeWarrior IDE with the following components:

- Left Panel:** Project tree for 'SCI Exercise.mcp' showing 'AS1:AsynchroSerial' and its various methods like 'BeforeNewSpeed', 'OnError', 'SendChar', etc.
- Center Panel:** Source code for 'SCI\_Exercise.c' showing include statements and a 'main' function that sends characters 'H', 'e', 'l', 'l', 'o', 'r'.
- Right Panel:** 'Bean Inspector AS1:AsynchroSerial' with the 'Methods' tab selected and circled in red. A red arrow points from this tab to the text box below.

- ❖ We need to wait for the 1<sup>st</sup> character send out completely if we want to send out 2<sup>nd</sup> character in SCI communication.
- ❖ kick Methods page of Bean Inspector AS1:AsynchroSerial



**❖ Enable GetRxIdle PE Bean by kicking this dial box**

The screenshot shows the Metrowerks CodeWarrior IDE with the following components:

- Left Panel:** Project tree for 'SCI Exercise.mcp' showing 'AS1:AsynchroSerial' and its various beans like 'BeforeNewSpeed', 'OnRxChar', 'SendChar', etc.
- Center Panel:** Source code for 'SCI\_Exercise.c' showing includes for 'AS1.h', 'PE\_Types.h', and a 'main' function that sends characters 'H', 'e', 'l', 'l', 'o', '\n'.
- Right Panel:** 'Bean Inspector AS1:AsynchroSerial' showing a list of beans. The 'GetRxIdle' bean is circled in red, and its dial box is also circled in red. Other beans include 'Enable', 'Disable', 'SendChar', 'RecvChar', etc.
- Bottom Panel:** System tray showing 'Metrowerks...', 'SCI', and system status (97%, 2:50 PM).

**❖ Enable GetTxComplete PE Bean by kicking this dial box**

```

#include "AS1.h"
/* Include shared modules, which are u
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

void main(void)
{
    /*** Processor Expert internal initi
    PE_low_level_init();
    /*** End of Processor Expert interne

    /* Write your code here */

    for(;;) {
        AS1_SendChar('H');
        AS1_SendChar('e');
        AS1_SendChar('l');
        AS1_SendChar('l');
        AS1_SendChar('o');
        AS1_SendChar('\r');
        AS1_SendChar('\n');
    }
}
    
```

Bean	Visibility	Help
Enable	don't generate code	
Disable	don't generate code	
EnableEvent	don't generate code	
DisableEvent	don't generate code	
RecvChar	generate code	
SendChar	generate code	
RecvBlock	don't generate code	
SendBlock	don't generate code	
ClearRxBuf	don't generate code	
ClearTxBuf	don't generate code	
CharsInRxBuf	don't generate code	
GetCharsInRxBuf	generate code	
CharsInTxBuf	don't generate code	
GetCharsInTxBuf	generate code	
SetBaudRateMode	don't generate code	
GetError	don't generate code	
GetBreak	don't generate code	
SetBreak	don't generate code	
TurnTxOn	don't generate code	
TurnTxOff	don't generate code	
TurnRxOn	don't generate code	
TurnRxOff	don't generate code	
SetIdle	don't generate code	
LoopMode	don't generate code	
ConnectPin	don't generate code	
GetRxIdle	generate code	
GetTxComplete	don't generate code	

**❖ Scroll up the screen to reach GetTxComplete PE Bean by kick the scroll bar**

```

#include "AS1.h"
/* Include shared modules, which are u
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

void main(void)
{
    /*** Processor Expert internal initi
    PE_low_level_init();
    /*** End of Processor Expert interne

    /* Write your code here */

    for(;;) {
        AS1_SendChar('H');
        AS1_SendChar('e');
        AS1_SendChar('l');
        AS1_SendChar('l');
        AS1_SendChar('o');
        AS1_SendChar('\r');
        AS1_SendChar('\n');
    }
}
    
```

Bean	Visibility	Comment
Enable	don't generate code	
Disable	don't generate code	
EnableEvent	don't generate code	
DisableEvent	don't generate code	
RecvChar	generate code	
SendChar	generate code	
RecvBlock	don't generate code	
SendBlock	don't generate code	
ClearRxBuf	don't generate code	
ClearTxBuf	don't generate code	
CharsInRxBuf	don't generate code	
GetCharsInRxBuf	generate code	
CharsInTxBuf	don't generate code	
GetCharsInTxBuf	generate code	
SetBaudRateMode	don't generate code	
GetError	don't generate code	
GetBreak	don't generate code	
SetBreak	don't generate code	
TurnTxOn	don't generate code	
TurnTxOff	don't generate code	
TurnRxOn	don't generate code	
TurnRxOff	don't generate code	
GetPin	don't generate code	
SetPin	don't generate code	
GetPinMode	generate code	
SetPinMode	generate code	
GetTxComplete	generate code	

The screenshot shows the Metrowerks CodeWarrior IDE with the following components:

- Left Panel (Project Explorer):** Shows the project structure for 'SCI\_Exercise.mcp'. Under 'External Modules', the 'PE Bean' is highlighted. A red circle is drawn around this 'PE Bean' entry.
- Center Panel (Code Editor):** Displays the source code for 'SCI\_Exercise.c'. The code includes headers for 'AS1.h', 'PE\_Types.h', 'PE\_Error.h', 'PE\_Const.h', and 'IO\_Map.h'. The 'main' function contains initialization code and a loop. A red circle highlights the line: `AS1_SendChar('H');AS1_GetTxComplete()`. A red arrow points from this circle to the 'PE Bean' in the project explorer.
- Bottom Panel (Component Browser):** Shows a list of communication components. 'AsynchroSerial' is selected and highlighted with a blue box. A red arrow points from this box to the 'PE Bean' in the project explorer.
- Right Panel (Properties):** Shows various properties for the selected component, including 'SetIdle' and 'LoopMode'.
- Annotation:** A red-bordered box on the right contains the text: **❖ Put PE Bean GetTxComplete to SCI\_Exercise.c:main by Drag-n-Drop Methods.** A red arrow points from this box to the 'PE Bean' in the project explorer.

The screenshot shows the Metrowerks CodeWarrior IDE. The main window displays the source code for `SCI_Exercise.c`. The code includes several headers and a `main` function. A red oval highlights the following code snippet:

```

while(AS1_GetTxComplete()==0){};

```

A red arrow points from a callout box to this code. The callout box contains the text:

❖ **Modify it as while statement.**

The IDE interface also shows a project browser on the left with various modules like `DisableEvent`, `RecvChar`, etc., and a component browser at the bottom listing communication modules like `AsynchroMaster`, `AsynchroSerial`, etc.

The screenshot shows the Metrowerks CodeWarrior IDE with the following components:

- Left Panel:** Project configuration for 'SCI\_Exercise.mcp'. It shows a list of modules with checkboxes, including 'DisableEvent', 'RecvChar', 'SendChar', 'RecvBlock', 'SendBlock', 'ClearRxBuf', 'ClearTxBuf', 'CharsInRxBuf', 'GetCharsInRxBuf', 'CharsInTxBuf', 'GetCharsInTxBuf', 'SetBaudRateMode', 'GetError', 'GetBreak', 'SetBreak', 'TurnTxOn', 'TurnTxOff', 'TurnRxOn', 'TurnRxOff', 'SetIdle', 'LoopMode', 'ConnectPin', 'GetRxIdle', and 'GetTxComplete'. Below this is a tree view for 'User Modules' containing 'SCI\_Exercise.c.main', 'Generated Modules', 'External Modules', 'Documentation', and 'PESL'.
- Center Panel:** Source code for 'SCI\_Exercise.c'. The code includes headers for 'AS1.h', 'PE\_Types.h', 'PE\_Error.h', 'PE\_Const.h', and 'IO\_Map.h'. The 'main' function contains Processor Expert initialization and a loop that sends characters 'H', 'e', 'l', 'l', 'o', '\n' to the SCI peripheral. Each character is sent inside a while loop that waits for 'AS1\_GetTxComplete() == 0'. A red circle highlights these while loops, and a callout box points to them with the text: **❖ Make every while statement for every character sending out.**
- Right Panel:** A tree view of hardware modules under 'Communication', including 'AsynchroMaster', 'AsynchroSerial', 'AsynchroSlave', 'FreescaleCAN', 'FreescaleH18', 'FreescaleSSI', 'Internall2C', 'SynchroMaster', 'SynchroSlave', and 'Converter'. 'SetIdle' and 'LoopMode' are marked as 'don't generate code'.
- Bottom Panel:** The Windows taskbar shows the Start button, several application icons, and the system tray with 'Metrowerks...', 'SCI', 'SCI\_Hands...', '97%' battery, and '2:56 PM'.



The screenshot shows the Metrowerks CodeWarrior IDE interface. The 'Processor Expert' menu is open, with 'Generate Code' selected for the project 'SCI Exercise.mcp'. The code editor displays the initialization code for the SCI module, including comments and function calls like `AS1_SendChar` and `AS1_GetTxComplete`. The component manager on the left shows the 'AsynchroSerial' module selected under the 'Communication' category. The error console at the bottom right shows a hint: 'Hint: File startup\56F80xx\_init.asm copied to the pro...'

The screenshot shows the Metrowerks CodeWarrior IDE interface. On the left is a project browser for 'SCI\_Exercise.mcp' showing various modules like 'DisableEvent', 'RecvChar', 'SendChar', etc. The main window displays the source code for 'SCI\_Exercise.c', which includes headers like 'AS1.h', 'PE\_Types.h', and a 'main' function. A 'Confirm' dialog box is overlaid on the code, asking if the user wants to save the file before code generation. The 'Yes' button is circled in red, and a red arrow points to it from a text box that reads: **❖ Kick Yes to update the code.**





The screenshot shows the Metrowerks CodeWarrior IDE interface. On the left, the 'Processor Expert' tab is active, displaying a list of modules for the 'SCI\_Exercise.mcp' project. A red circle highlights the 'Debug' button in the top toolbar of this tab. A red arrow points from a text box to this button. The main editor window shows the source code for 'SCI\_Exercise.c', which includes headers for 'AS1.h', 'PE\_Types.h', 'PE\_Error.h', 'PE\_Const.h', and 'IO\_Map.h'. The code defines a 'main' function that performs processor expert initialization and then sends a sequence of characters ('H', 'e', 'l', 'l', 'o', '\n') to the AS1 module. The bottom panel shows a tree view of hardware modules, including 'AsynchroSerial' and 'LoopMode'. A text box with a red border contains the instruction: '❖ Kick green arrow to download the code to 56F801xEVB.'

The screenshot shows the Metrowerks CodeWarrior IDE interface. The main window displays the source code for 'SCI\_Exercise.c'. A 'Building SCI\_Exercise.mcp' dialog box is open, showing a table of files being compiled:

File	Task	File Count	Line Count
56F80xx_init.asm	Compiling...	9	0
Totals:		9	29030

The 'Component Selection' tree on the right shows the following components:

- AsynchroMaster
- AsynchroSerial
- AsynchroSlave
- FreescallCAN
- FreescallH18
- FreescallSSI
- Internall2C
- SynchroMaster
- SynchroSlave
- Converter

The 'Properties' window at the bottom right shows the 'Bean Level: High' and 'Errors: 0, warnings: 0, hints: 0'.



The screenshot displays the Metrowerks CodeWarrior IDE with the following components:

- Project Explorer (Left):** Shows a project named "SCI Exercise.mcp" with a target "smm pROM-xRAM". A list of modules is visible, including "DisableEvent", "RecvChar", "SendChar", "ClearRxBuf", "ClearTxBuf", "CharsInRxBuf", "CharsInTxBuf", "GetCharsInRxBuf", "GetCharsInTxBuf", "SetBaudRateMode", "GetError", "GetBreak", "SetBreak", "TurnTxOn", "TurnTxOff", "TurnRxOn", "TurnRxOff", "SetIdle", "LoopMode", "ConnectPin", "GetRxIdle", "GetTxComplete", "User Modules", "SCI\_Exercise.c.main", "Generated Modules", "External Modules", "Documentation", and "PESL".
- Code Editor (Center):** Displays the source code for "SCI\_Exercise.c". The code includes headers for "AS1.h", "PE\_Types.h", "PE\_Error.h", "PE\_Const.h", and "IO\_Map.h". The main function performs internal initialization and contains a loop for sending and receiving characters. A "Downloading..." dialog box is overlaid on the code, indicating 2352 bytes are being downloaded.
- Component Explorer (Bottom Center):** Shows a tree view of communication modules, including "AsynchroMaster", "AsynchroSerial", "AsynchroSlave", "FreescaleCAN", "FreescaleH18", "FreescaleSSI", "Internall2C", "SynchroMaster", "SynchroSlave", and "Converter".
- Properties Window (Bottom Right):** Shows settings for the selected module, with options for "SetIdle", "LoopMode", "ConnectPin", "GetRxIdle", and "GetTxComplete". The "Generate Code" checkbox is checked for "GetTxComplete".
- Status Bar (Bottom):** Shows "97%" and "2:57 PM".

The screenshot displays the Metrowerks CodeWarrior IDE. The main window shows a project named "SCI Exercise.mcp" with a list of modules and targets. The "Processor Expert" tab is active, showing a list of hardware modules like "InternalZC", "SynchroMaster", and "SynchroSlave". The "Target CPU" window shows "Cpu:56F8013VFAE". The "Bean Inspector" window shows "AS1:AsynchroSerial". The "Stack" window shows "Finit\_56800" and "main". The "Variables" window shows "No local variables". The source code editor displays the following code:

```
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

void main(void)
{
    /*** Processor Expert internal initialization. DON'T REMOVE TI
    PE_low_level_init();
    /*** End of Processor Expert internal initialization.

    /* Write your code here */

    for(;;) {
        AS1_SendChar('H'); while(AS1_GetTxComplete()==0){};
        AS1_SendChar('e'); while(AS1_GetTxComplete()==0){};
    }
}
```

The system tray at the bottom shows the Start button, several application icons, the Metrowerks logo, a taskbar with "SCI" and "SCI\_Hands...", a 97% battery indicator, and the time 2:58 PM.

The screenshot shows the Metrowerks CodeWarrior IDE interface. On the left, a project tree for 'SCI\_Exercise.mcp' is visible, with 'sdm\_pROM\_xRAM' selected. The main window displays the 'sdm\_pROM\_xRAM.elf (Thread 0x0)' window, which includes a 'Stack' view and a 'Source' view. The 'Source' view shows the message: 'Program "sdm\_pROM\_xRAM.elf" is executing. Choose Break from the Debug menu to stop it.' A red circle highlights a green arrow icon in the toolbar of the thread window. A red arrow points from a text box to this icon. The text box contains the instruction: 'Kick green arrow to run the code inside 56F801xEVB.'

❖ **Invoke HyperTerminal for getting the characters from 56F801xEVB.**

The screenshot shows the 'New Connection - HyperTerminal' window. A 'Connection Description' dialog box is open, prompting the user to 'Enter a name and choose an icon for the connection:'. The 'Name:' text box contains the text '9600', which is highlighted with a red circle. Below the text box is an 'Icon:' section with several icons. A red arrow originates from the '9600' text and points to a callout box containing the text '❖ Type the name you want'. The background of the HyperTerminal window shows a project tree on the left with folders like 'User Modules', 'Generated Modules', 'External Modules', 'Documentation', and 'PESL'. The main window displays source code for 'exercise.c' with line numbers and column indicators. The Windows taskbar at the bottom shows the Start button, several application icons, and the system tray with a battery level of 97% and the time 3:03 PM.



**❖ Select your PC COM port for communication**

COM1 Properties

Port Settings

Bits per second: 9600

Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

Restore Defaults

OK Cancel Apply

❖ Select 9600 baud rate, 8 data bit, no parity, 1 stop bit, none flow control

❖ Then, kick OK

The screenshot shows a HyperTerminal window titled "9600 - HyperTerminal" with a menu bar (File, Edit, View, Call, Transfer, Help) and a toolbar. The main window displays a list of "Hello" messages. A red circle highlights the text, and a red arrow points from a callout box to it. The callout box contains the text: "❖ 'Hello' is shown on the HyperTerminal screen".

Below the terminal window, a project tree is visible with folders for "User Modules", "Generated Modules", "External Modules", "Documentation", and "PESL". The "PESL" folder is expanded, showing "2 files".

The Windows taskbar at the bottom shows the Start button, several application icons, and a system tray with a battery level indicator at 97% and the time 3:03 PM.

# Summary

56800/E Training

# Summary

- ❖ Understand the hardware and software support available for the 56800E Hybrid Controller product line.
- ❖ Demonstrated the ease of developing applications using CodeWarrior development tools with Processor Expert™ technology.

# 56800E



*Thank You!!!*