SNiFF+

Release 3.2
for Windows

Tutorial
for the
Integration of SNiFF+ 3.2 in pRISM+ 2.0

V 1.2

TakeFive Software, a Wind River Company

TakeFive Software GmbH
A Wind River Company
Salzburg, Austria
E-Mail: info@takefive.co.at

Wind River Systems, Inc
Alameda, CA
E-Mail: *info@windriver.com*

1 Introduction

The **Integration of SNiFF+ 3.2 in pRISM+ 2.0** is an upgrade to the integration of **SNiFF**+ 3.0.2 shipped on the **pRISM+** product CD-ROM. Its purpose is to have the latest Version of **SNiFF+** integrated in **pRISM+ 2.0** for Win32 Win9x and WinNT4.0 for the target CPU families **68K**, **ARM**, **MIPS**, **PowerPC** and **x86**.

This Tutorial helps developers of embedded applications using **pRISM+** development environment to use this integration easily and efficiently.

There are several Chapters addressing the needs of the

- **Single user** doing an evaluation as well as for the single developer
- **Small group** of developers
- **large group** of developers
- **BSP** developer

This Tutorial is especially written for the single user doing an evaluation as well as for the single developer. At the end you find a chapter discussing how to add another user to a pRISM+ project.

pRISM+:

pRISM+ is a real time application development environment, which combines the real time operating system **pSOSystem** with additional development tools like the compiler, debugger and editor. **pRISM+** is available for embedded target systems based on **68K**, **ARM**, **MIPS**, **PowerPC** and **x86** processors and has **SNiFF+ 3.0.2** as an optional component in the shipment.

SNiFF+:

SNiFF+ is a Source Code Engineering tool to analyse and navigate through source code. **SNiFF+** is designed for use in a multi-platform, multi-language development team environment. **SNiFF+** can analyse C/C++, Java, Fortran, IDL and ADA and can be extended for more languages through a public open API. **SNiFF+** provides functionality to handle the team make support and the team version and configuration management.

2 Abstract

This Tutorial has been written for your convenience. You should be able to find out about using SNiFF+ from within pRISM+. All steps are shown by adding screenshots, so there is no need to have a running host computer. Although it is highly recommended to follow the tutorial by doing each step on your host computer while you go through it.

You will see how you can:

- Create a new project based on the standard pSOSystem application pdemo
- Browse the Source files to understand the functionality using the:
    - Source Editor (SE)
    - Cross Referencer (CR)
    - Browser (IB)
    - Symbol Browser (SB)
    - Retriever (RE)
- Apply and use the Version Control Tool RCS
- Find out about history and differences of the generated versions of your source files
- Compile and link your application
- Correct compiletime or linktime errors
- Switch the BSP
- Adapting a Custom BSP

3 Assumptions

The product **pRISM+** comes with online help and printed documentation (also available in PDF format). It is highly recommended to have experienced the **pRISM+** Tutorial (**pRISM+ Toolbar menu Help->Helptopics, How To …, pRISM+ Tutorial**) and to read the documentation called "**pRISM+ User's Guide**" and here the chapter "**Using SNiFF+ in pRISM+ Environment**". There is an updated version of this chapter shipped in the **Integration of SNiFF+ 3.2 in pRISM+ 2.0**.

This document contains screenshots and example code of the **pRISM+ for PowerPC** installation. For other targets remarkable differences are stated as a note where the differences occur.

4 SNiFF+ in pRISM+ for the Single User

Starting with a standard pSOSystem application (pdemo):

Preparations

If you already generated the standard application pdemo, delete in the directory of the Private Working Environment **$PSS_USER_PWE** the directories “**apps/pdemo**” and “**.ProjectCache**”, so we can start in a clean environment.

The standard **pSOSystem** applications are designed to help to understand, how application programming is done in **pRISM+** and **pSOSystem**.

It is not designed to start from here a team-based development. The single user may use the standard application as a starting point of his application.

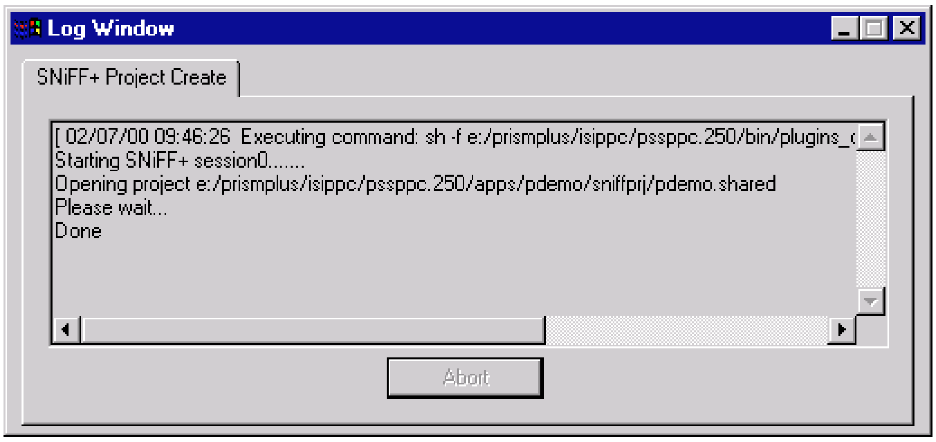
You should **not** do **team development** using **standard application** environment.

Please refer to the topic “**Team development and standard pSOSystem applications**” at the end of this chapter.

Generation of a new project …

Generate the **pdemo** example project using **pRISM+ Toolbar** as described in the **Tutorial** (**File->New->SNiFF+->Next->Starting with a pSOSystem example application ->Next->pdemo->Finish**).

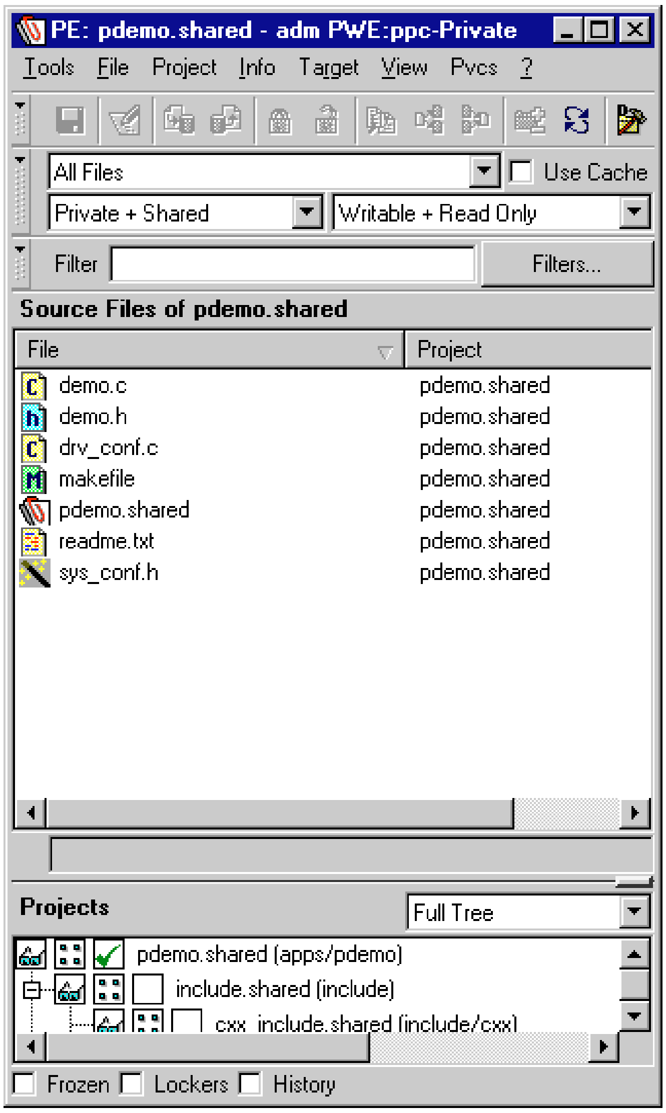
The Log Window of **pRISM+** opens, if **pRISM+** is freshly started or you did not close it in the meantime. You may close it. The text in the window states the progress of the **SNiFF+** open process, which is at an end, when the “**Abort**” button is greyed out and the “**Done**” text is displayed.



```
[ 02/07/00 09:46:26  Executing command: sh -f e:/prismplus/isippc/pssppc.250/bin/plugins_(
Starting SNiFF+ session0.......
Opening project e:/prismplus/isippc/pssppc.250/apps/pdemo/sniffprj/pdemo.shared
Please wait...
Done
```

Project Editor (PE)

Now the SNiFF+ V3.2 **Launch pad** appears and the **PE** (Project Editor) gets displayed:



File	Project
demo.c	pdemo.shared
demo.h	pdemo.shared
drv_conf.c	pdemo.shared
makefile	pdemo.shared
pdemo.shared	pdemo.shared
readme.txt	pdemo.shared
sys_conf.h	pdemo.shared

```
pdemo.shared (apps/pdemo)
  include.shared (include)
    cxx_include.shared (include/cxx)
```

The **PE** has two window sections. The upper one is displaying the files of the project, selected in the lower window, the project window with the checkmark.

The file attributes of all relevant files of **pSOSystem** are set to read-only.

All files in the file window appear in non-bold and non-italic font. This means, that there is no risk to destroy the standard example application by accident.

The project tree may be collapsed or expanded by clicking on the "-" respectively "+" symbol in front of every project. The projects of the standard **pSOSystem** applications come with predefined attributes:

The application and **BSP** (Board Support Package) parts of the project are configured to be private to the user and are marked writable. (See the pencil icon in the first column.)

The other projects are containing code related to **pSOSystem** real time operating system (**RTOS**) and are global to all users (Shared). For that reason, some **pSOSystem** related projects (**sys_os.shared** and **config_std.shared**) are marked **read-only**. (See the "frozen" icon in the first column.)

Version Control (CMVC)

The Version Control Tool by default is **RCS**.

On Windows hosts, the files of the private part of the projects are copied to a private directory, the private workspace. It contains the private working environment. The Shell environment variable **$PSS_USER_PWE** points to that location. On Unix hosts symbolic links are used to reduce the amount of disk space (Windows does not support symbolic links).

It is highly recommended to use version control to help to find out about the history of a file, set of files or the complete project tree. It is easy to use, since the version control is designed in the GUI of **SNiFF+.**

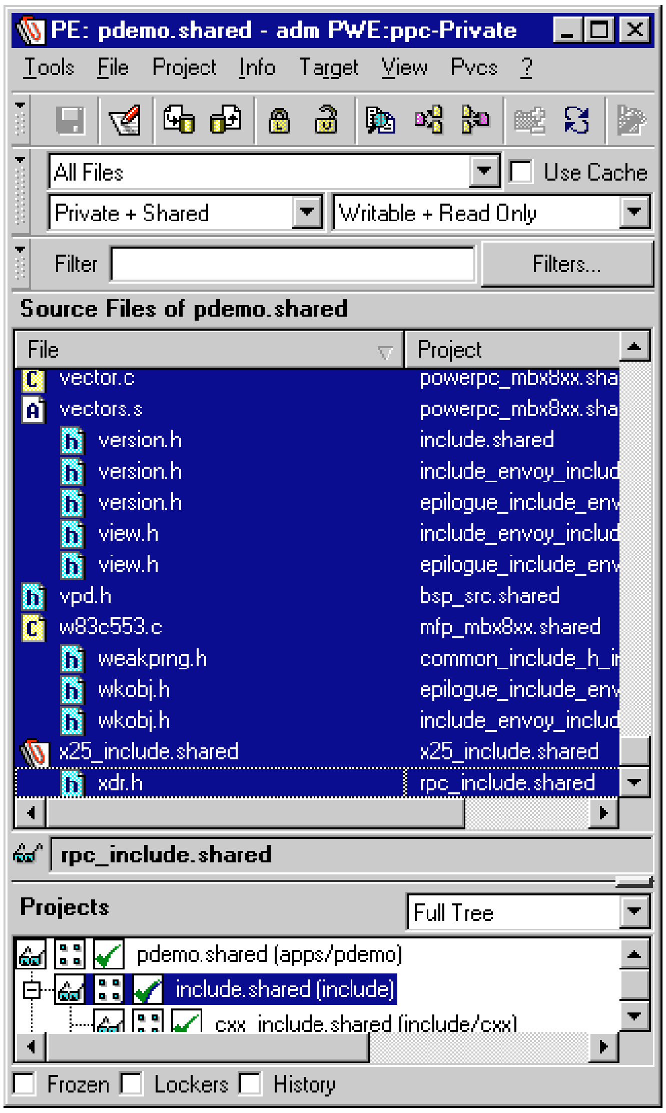
Usually the single developer checks in all files for the first time and checks out the file(s) he/she needs to modify.

When the implementation is tested well enough, the modified version(s) of the file(s) is (are) checked in again with a comment attached. This helps in the maintenance phase of a project.

To find out about the status of each file there is a checkmark in the **PE** to display this information instantly ("**Lockers**"). To also instantly display the history of the file selected in the file window of the **PE** there is the button "**History**" available.

The buttons "**Lockers**" and "**History**" are not intended for permanent use, since they decrease the overall performance of the SNiFF+ development environment.

To check in all files, select a project in the project window and use the context menu to "**Select from All Projects**". Now every project has a green checkmark and a lot of files are displayed in the file window. Select a file in the file window and use the context menu to "**Select All**" files.



PE: pdemo.shared - adm PWE:ppc-Private

All Files

Private + Shared

Writable + Read Only

Filter

Source Files of pdemo.shared

File	Project
vector.c	powerpc_mbx8xx.sha
vectors.s	powerpc_mbx8xx.sha
version.h	include.shared
version.h	include_envoy_includ
version.h	epilogue_include_env
view.h	include_envoy_includ
view.h	epilogue_include_env
vpd.h	bsp_src.shared
w83c553.c	mfp_mbx8xx.shared
weakprng.h	common_include_h_i
wkobj.h	epilogue_include_env
wkobj.h	include_envoy_includ
x25_include.shared	x25_include.shared
xdr.h	rpc_include.shared

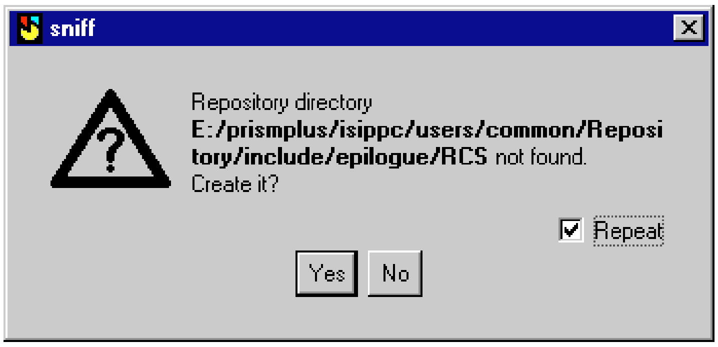
rpc_include.shared

Projects — Full Tree

pdemo.shared (apps/pdemo)
include.shared (include)
cxx_include.shared (include/cxx)

Check in all files by using the context menu “**Check In Files…**”.

You get asked to generate the repository directory tree, which is representing the directory structure of your entire project. Checkmark the “**Repeat**” button and answer with “**Yes**”.



sniff

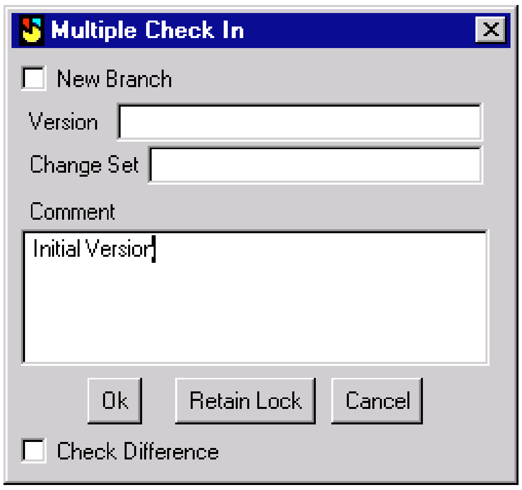
Repository directory **E:/prismplus/isippc/users/common/Repository/include/epilogue/RCS** not found. Create it?

[✓ Repeat]

Yes No

The version control asks for a comment of this initial version. You may also enter a starting version number, default value is **”1.1”**. Acknowledge the dialog and all files get checked in now.

> If you already have checked in the **pSOSystem** related files and the **BSP**, check in only the files of the project “**pdemo.shared**”. The check in of all files takes some time (2-15 minutes).



Multiple Check In

New Branch

Version

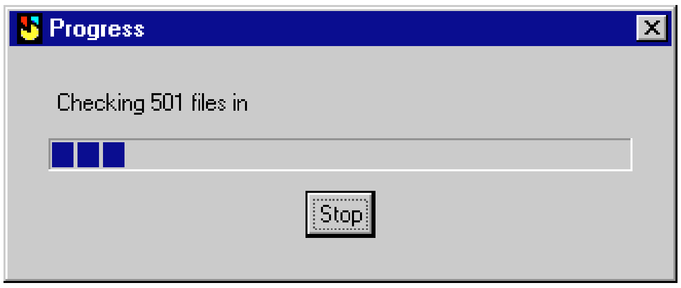
Change Set

Comment

Initial Version

Ok Retain Lock Cancel

Check Difference

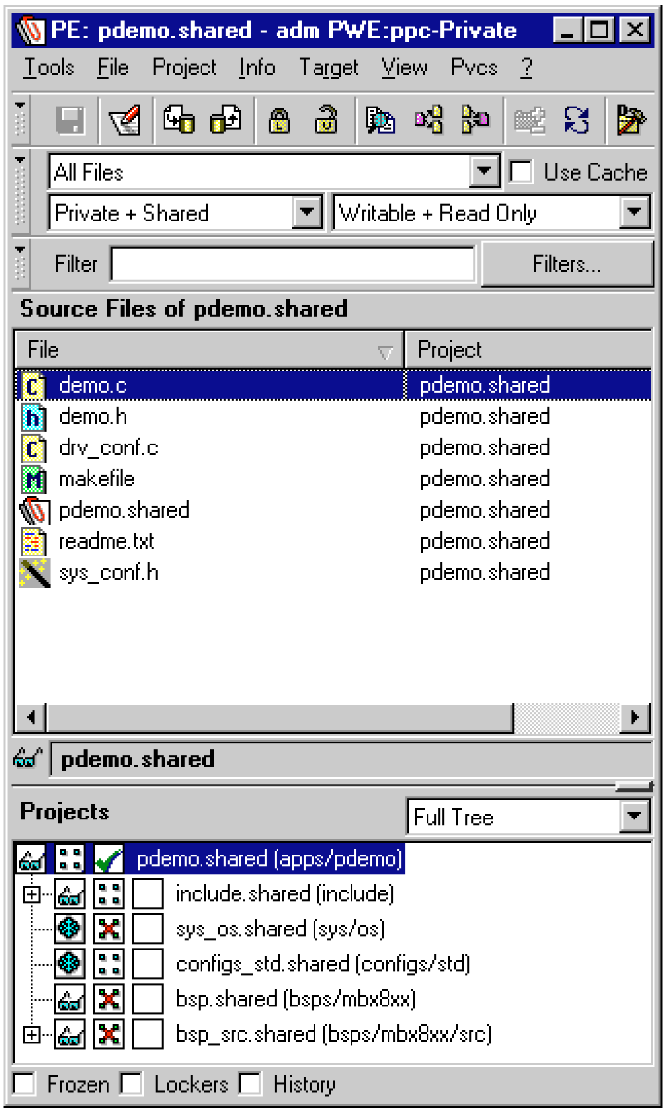


Progress

Checking 501 files in

Stop

To verify the success of the check in, checkmark the “**Lockers**” button at the bottom line of the **PE** window and the file window should contain an additional third column displaying the version control tool you use. For a better demonstration effect, we select the “**pdemo.shared**“ project only (using the context menu) and the file “**demo.c**” by selecting it):


PE: pdemo.shared - adm PWE:ppc-Private

All Files ☐ Use Cache
Private + Shared | Writable + Read Only
Filter | Filters...

Source Files of pdemo.shared

File	Project
demo.c	pdemo.shared
demo.h	pdemo.shared
drv_conf.c	pdemo.shared
makefile	pdemo.shared
pdemo.shared	pdemo.shared
readme.txt	pdemo.shared
sys_conf.h	pdemo.shared

pdemo.shared

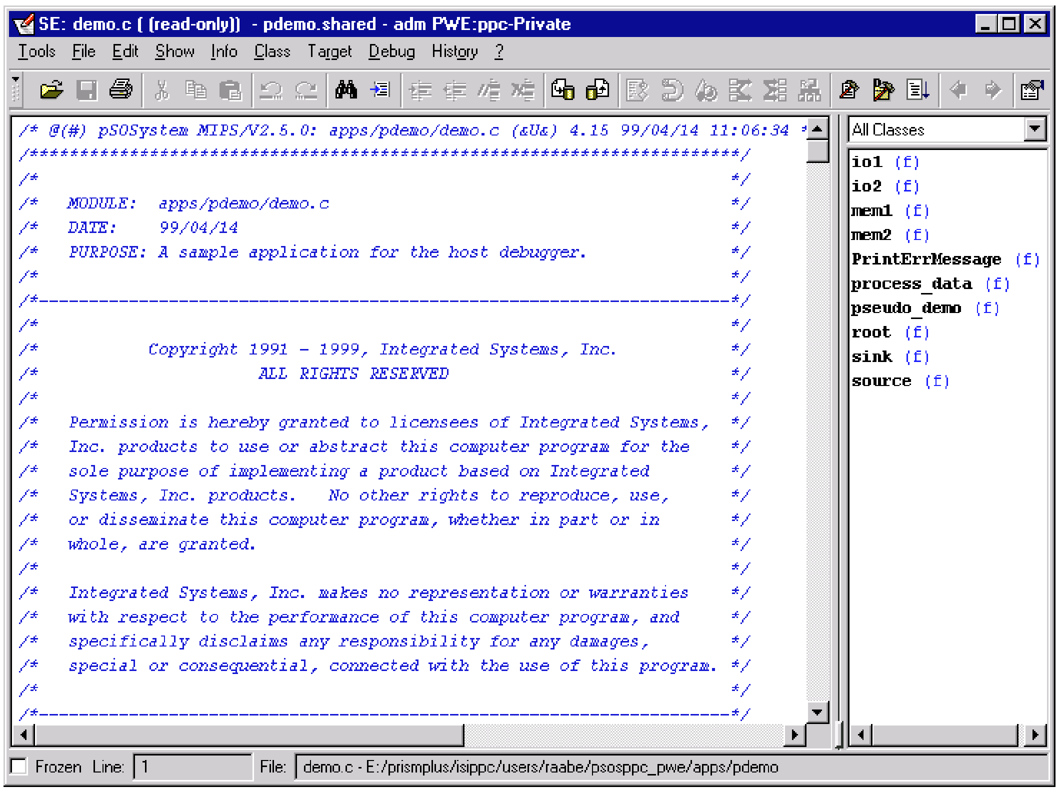
Projects: Full Tree

- pdemo.shared (apps/pdemo)
  - include.shared (include)
  - sys_os.shared (sys/os)
  - configs_std.shared (configs/std)
  - bsp.shared (bsps/mbx8xx)
  - bsp_src.shared (bsps/mbx8xx/src)

☐ Frozen ☐ Lockers ☐ History

To take a closer look at the pdemo application, please refer to the **pSOSystem** documentation, **pRISM+ Online Help** (**Help->Help Topics, How To ...->pRISM+ Tutorial**) and the file “**readme.txt**”.

To find out about the application we start the **SE** (Source Editor) by double clicking the file “**demo.c**”, which contains the root task and other tasks, demonstrating most of the inter task communication mechanisms. The title of the **SE** shows, that the file is read-only and any attempt of adding new characters is refused.


SE: demo.c ((read-only)) - pdemo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes

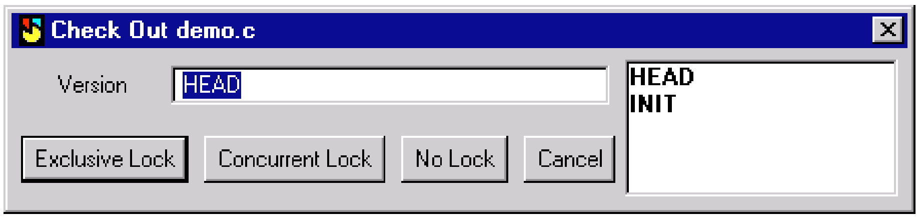
```
/* @(#) pSOSystem MIPS/V2.5.0: apps/pdemo/demo.c (&U&) 4.15 99/04/14 11:06:34 *
/***********************************************************************/
/*                                                                     */
/*   MODULE:  apps/pdemo/demo.c                                        */
/*   DATE:    99/04/14                                                 */
/*   PURPOSE: A sample application for the host debugger.              */
/*                                                                     */
/*---------------------------------------------------------------------*/
/*                                                                     */
/*          Copyright 1991 - 1999, Integrated Systems, Inc.            */
/*                      ALL RIGHTS RESERVED                            */
/*                                                                     */
/*   Permission is hereby granted to licensees of Integrated Systems,  */
/*   Inc. products to use or abstract this computer program for the    */
/*   sole purpose of implementing a product based on Integrated        */
/*   Systems, Inc. products.   No other rights to reproduce, use,      */
/*   or disseminate this computer program, whether in part or in       */
/*   whole, are granted.                                               */
/*                                                                     */
/*   Integrated Systems, Inc. makes no representation or warranties    */
/*   with respect to the performance of this computer program, and     */
/*   specifically disclaims any responsibility for any damages,        */
/*   special or consequential, connected with the use of this program. */
/*                                                                     */
/*---------------------------------------------------------------------*/
```

```
io1 (f)
io2 (f)
mem1 (f)
mem2 (f)
PrintErrMessage (f)
process_data (f)
pseudo_demo (f)
root (f)
sink (f)
source (f)
```

Frozen Line: 1 File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

To make this file writable, you need to check the file out of the version control tool using the context menu “**Check Out File**”.
You get asked, how to check out what version.
Select “**Exclusive Lock**”, so the system grants you the exclusive right to modify the file.

> This is helpful if you attempt to check out again later on, then you get warned, that the file is already checked out.



In our case we get offered to check out the version “**HEAD**”, which is the desired version.

> If you have already checked in more versions, then the Name “**HEAD**” is always the latest version and the version “**INIT**” is the oldest one.

Now the title of the **SE** shows, the file is writable.

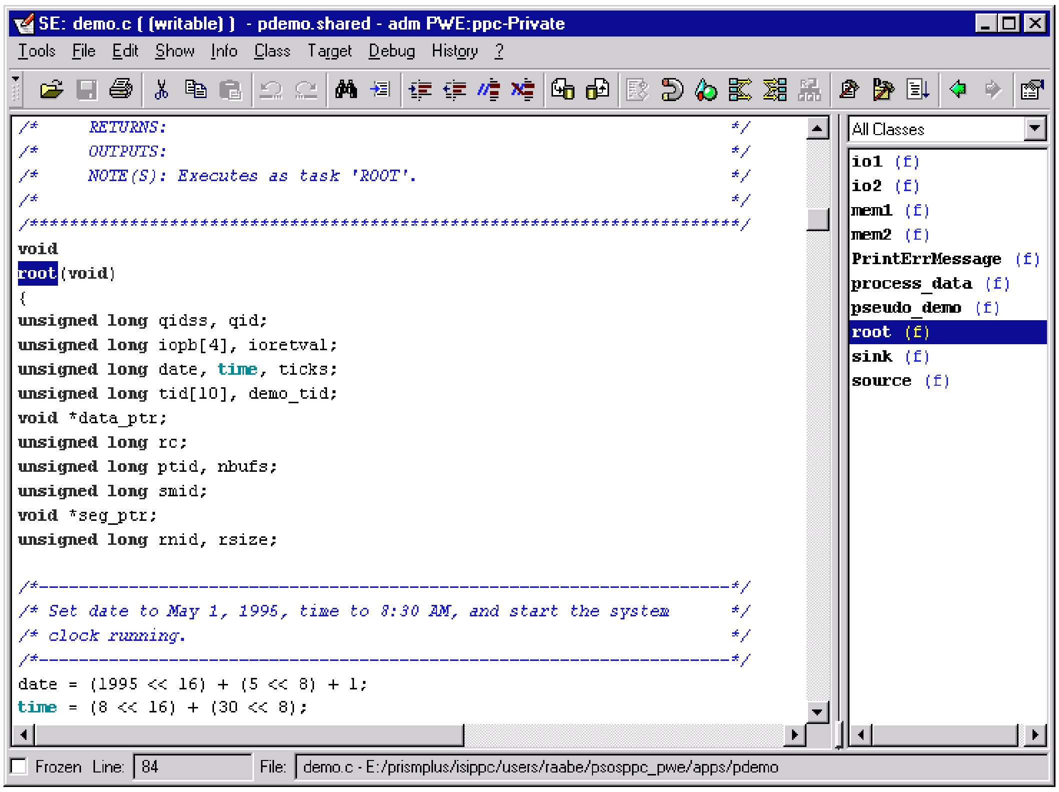
… and Browsing and understanding Source Code

Source Editor (SE)

First we look at the **SE**. There is the source file displayed in the main window displaying the file and there is a smaller window on the right displaying additional information on basis of a symbol table.

The symbol table is generated on creation time of the projects and when a modification is saved. So the information is ready to use whenever you save your file(s).

The right window is the quick navigation window and shows all C-functions in this file. The first function called by **pSOSystem** is "**root ()**". To quickly navigate to **root ()** click on the text **root (f)** in the right window and the code is displayed.



SE: demo.c ((writable)) - pdemo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes

```
/*      RETURNS:                                                          */
/*      OUTPUTS:                                                          */
/*      NOTE(S): Executes as task 'ROOT'.                                 */
/*                                                                        */
/**************************************************************************/
void
root(void)
{
unsigned long qidss, qid;
unsigned long iopb[4], ioretval;
unsigned long date, time, ticks;
unsigned long tid[10], demo_tid;
void *data_ptr;
unsigned long rc;
unsigned long ptid, nbufs;
unsigned long smid;
void *seg_ptr;
unsigned long rnid, rsize;

/*------------------------------------------------------------------------*/
/* Set date to May 1, 1995, time to 8:30 AM, and start the system       */
/* clock running.                                                         */
/*------------------------------------------------------------------------*/
date = (1995 << 16) + (5 << 8) + 1;
time = (8 << 16) + (30 << 8);
```

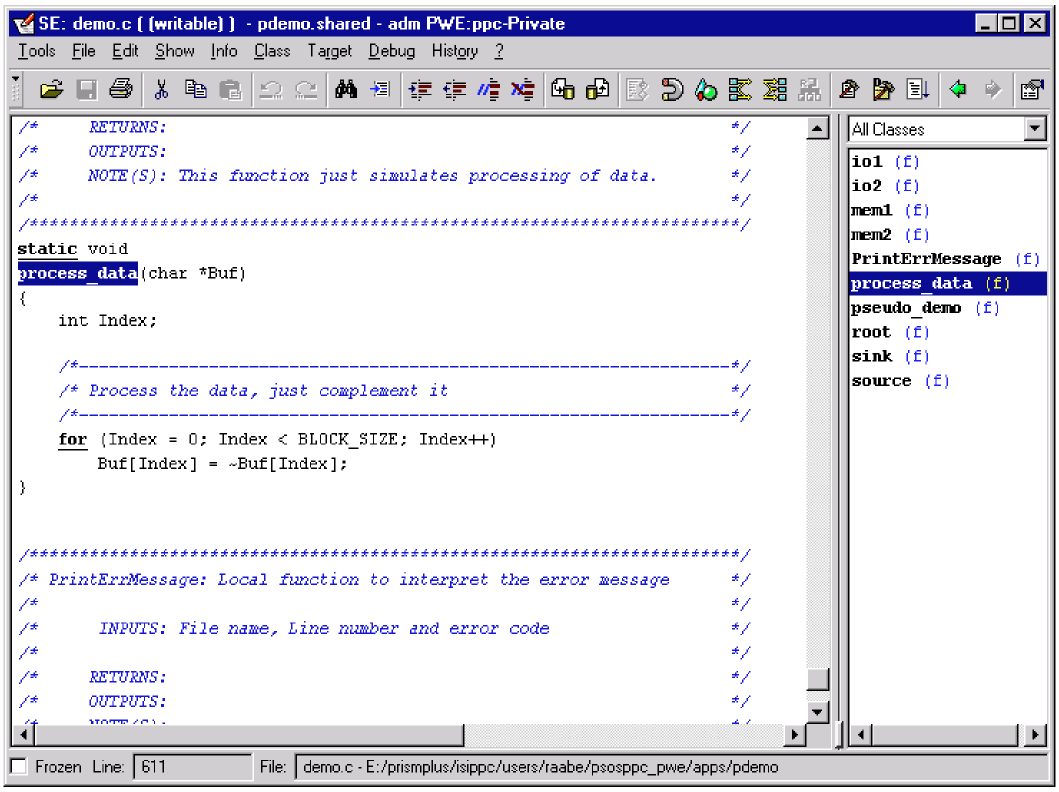
```
io1 (f)
io2 (f)
mem1 (f)
mem2 (f)
PrintErrMessage (f)
process_data (f)
pseudo_demo (f)
root (f)
sink (f)
source (f)
```

Frozen Line: 84 File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

In the quick navigation window you also see there is a function "**process_data ()**" shown. Click on it and in the main window the function name is highlighted.

If you user **pRISM+ for 68K** or **ARM** there is **pSOSystem 2.3.0** included. Here the function "**process_data ()**" is in a different file. So please use" **mem1 ()**" instead for this example!

When you are in the **SE** and have a file open, there is a shortcut (Ctrl-E on Windows) or a menu entry (**Show->Header File** when in the source of an implementation file or **Show->Implementation File** when in the source of an header file) to switch between the implementation and header file back and forth.


SE: demo.c ((writable)) - pdemo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes

```
/*     RETURNS:                                                      */
/*     OUTPUTS:                                                      */
/*     NOTE(S): This function just simulates processing of data.     */
/*                                                                   */
/*********************************************************************/
static void
process_data(char *Buf)
{
    int Index;

    /*--------------------------------------------------------------*/
    /* Process the data, just complement it                         */
    /*--------------------------------------------------------------*/
    for (Index = 0; Index < BLOCK_SIZE; Index++)
        Buf[Index] = ~Buf[Index];
}


/*********************************************************************/
/* PrintErrMessage: Local function to interpret the error message    */
/*                                                                   */
/*      INPUTS: File name, Line number and error code                */
/*                                                                   */
/*     RETURNS:                                                      */
/*     OUTPUTS:                                                      */
```

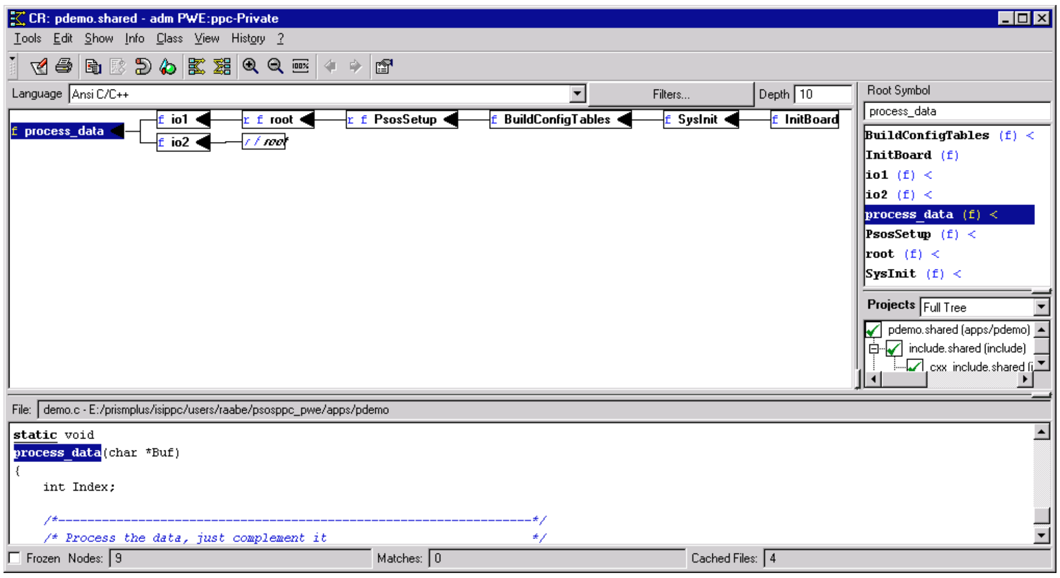
```
io1 (f)
io2 (f)
mem1 (f)
mem2 (f)
PrintErrMessage (f)
process_data (f)
pseudo_demo (f)
root (f)
sink (f)
source (f)
```

Frozen Line: 611 File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

Cross Referencer (CR)

To find out about who calls (refers to) this function ask **SNiFF+** by using the context menu "**process_data Referred-By**".

A new tool window opens containing the **CR** (**Cross Referencer**). Here is shown that **process_data ()** is called by **io1 ()** and **io2 ()** function. By default the depth of nesting references is set to "**1**". Set this to "**10**" and repeat the question to **SNiFF+** by hitting the <**ENTER**> key. Now a more complex call tree is displayed. Here you see that **io1 ()** is referenced (not called) by **root ()** and **io2 ()** as well. To prevent the user to be confused by too much redundant information, the name **root** in the second line is printed in italic font, meaning there is another graph containing **root** and the **root** referencing symbols.

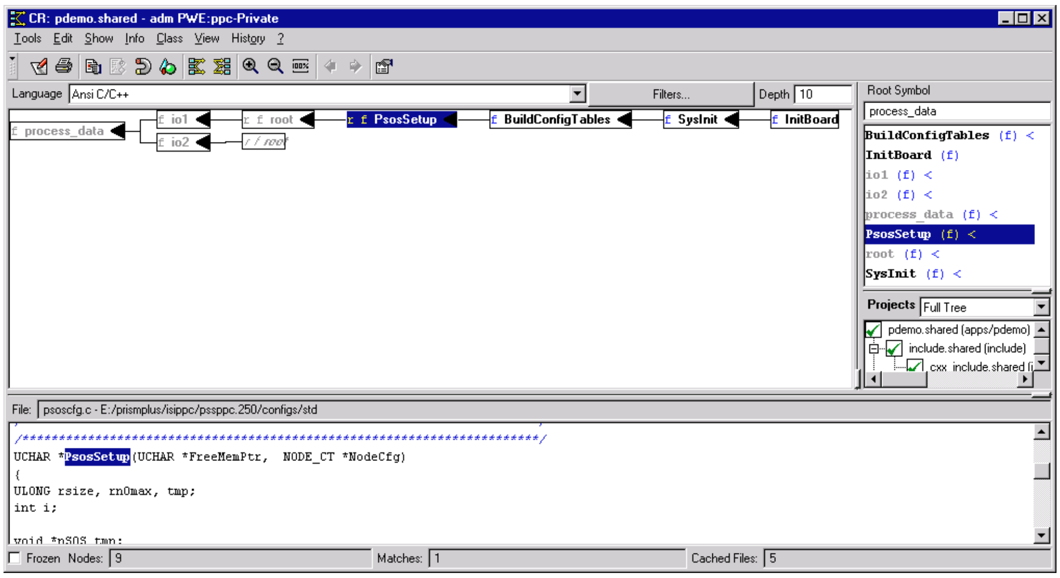


```
BuildConfigTables (f) <
InitBoard (f)
io1 (f) <
io2 (f) <
process_data (f) <
PsosSetup (f) <
root (f) <
SysInit (f) <
```

File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

```
static void
process_data(char *Buf)
{
    int Index;

    /*-----------------------------------------------------------------*/
    /* Process the data, just complement it                            */
```

Frozen Nodes: 9 Matches: 0 Cached Files: 4



```
BuildConfigTables (f) <
InitBoard (f)
io1 (f) <
io2 (f) <
process_data (f) <
PsosSetup (f) <
root (f) <
SysInit (f) <
```

File: psoscfg.c - E:/prismplus/isippc/pssppc.250/configs/std

```
/*********************************************************************/
UCHAR *PsosSetup(UCHAR *FreeMemPtr,  NODE_CT *NodeCfg)
{
ULONG rsize, rn0max, tmp;
int i;

void *pSOS_tmp;
```

Frozen Nodes: 9 Matches: 1 Cached Files: 5

When clicking on one symbol, the source code corresponding to the implementation of the symbol is displayed in the lower window.

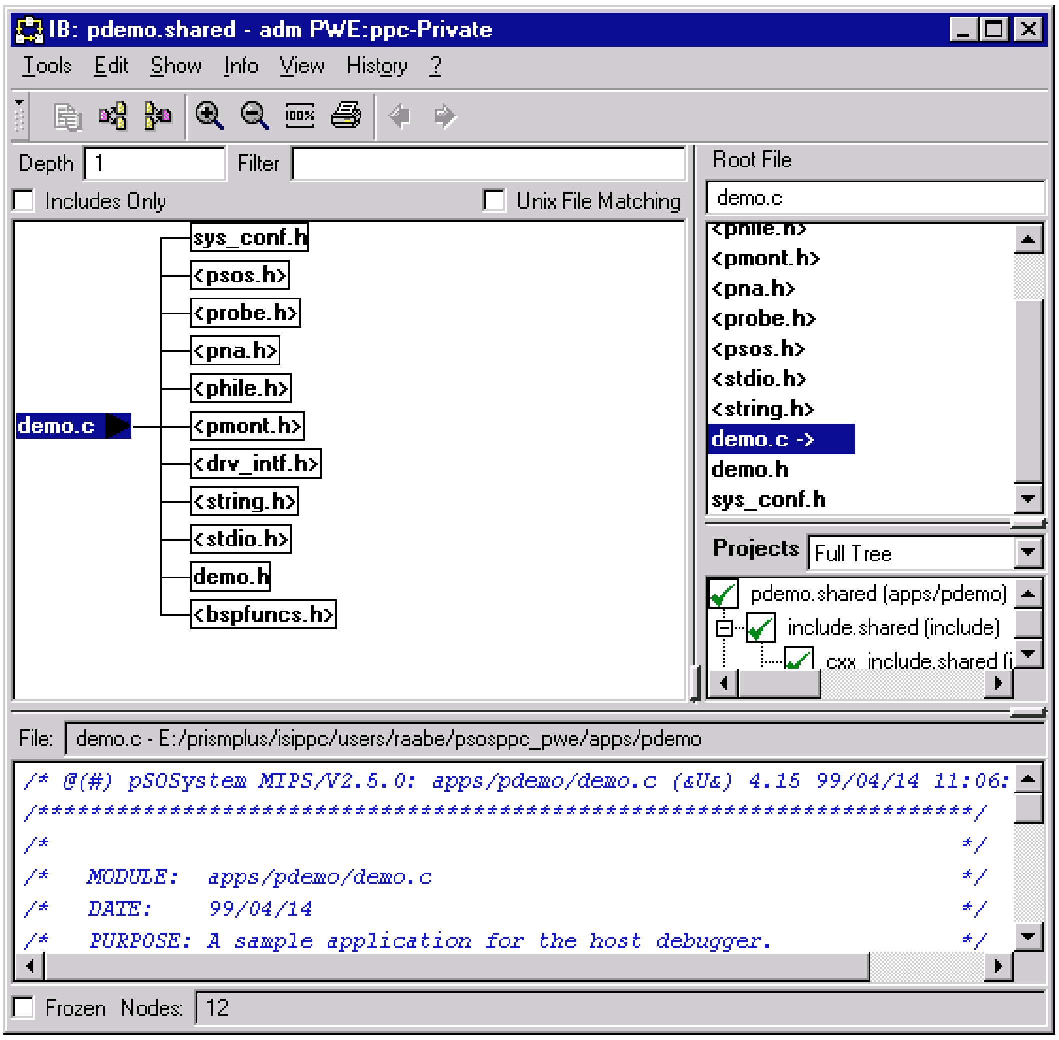
When clicking on one symbol while pressing the left shift key, the source code corresponding to the reference to the symbol is displayed in the lower window.

Double clicking instead of single clicking opens the **SE** tool at the appropriate position.

Include Browser (IB)

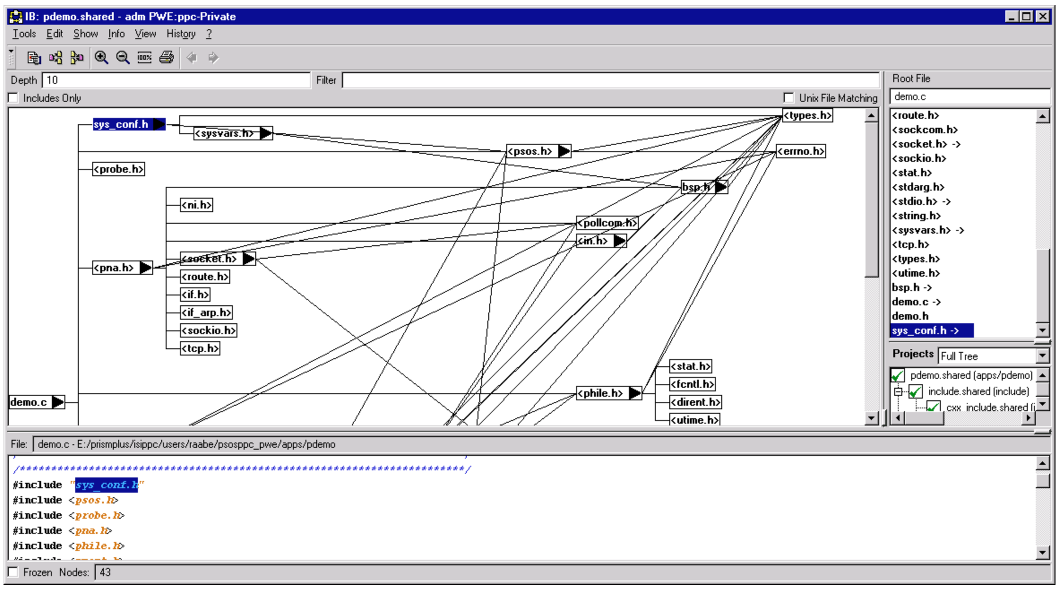
To find out about who includes whom, and who gets included by whom, there is the tool **IB** (**Include Browser**) available. When you are in the **SE** you may ask which include files are included by **demo.c** by using the menu "**Info->demo.c includes**".

Now the **IB** starts and displays all files which are included by a "#include …" statement. As already seen in the **CR**, the depth of nesting includes is set to "**1**" by default. Set it to "**10**" and press the <**ENTER**> key, to see, which include tree is to be seen.



File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

```
/* @(#) pSOSystem MIPS/V2.5.0: apps/pdemo/demo.c (&U&) 4.15 99/04/14 11:06:
/***********************************************************************/
/*                                                                     */
/*   MODULE:  apps/pdemo/demo.c                                        */
/*   DATE:    99/04/14                                                 */
/*   PURPOSE: A sample application for the host debugger.              */
```



Depth 10
Filter
Includes Only
Unix File Matching
Root File: demo.c

```
<route.h>
<sockcom.h>
<socket.h> ->
<sockio.h>
<stat.h>
<stdarg.h>
<stdio.h> ->
<string.h>
<sysvars.h> ->
<tcp.h>
<types.h>
<utime.h>
bsp.h ->
demo.c ->
demo.h
sys_conf.h ->
```

Projects: Full Tree
✓ pdemo.shared (apps/pdemo)
✓ include.shared (include)
✓ cxx include.shared (i

File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

```
/************************************************************************/
#include "sys_conf.h"
#include <psos.h>
#include <probe.h>
#include <pna.h>
#include <phile.h>
```

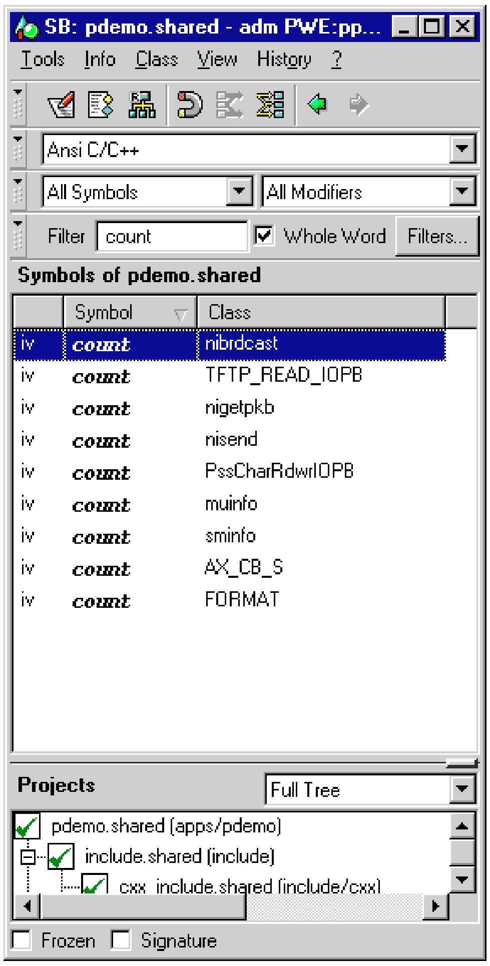
Frozen Nodes: 43

As you can see, the window design is the same as in the **CR**. The Source Code of respectively the reference to the corresponding include file to the highlighted box is displayed in the lower window. Here also double clicking to the box opens the **SE**.

The right hand window is like in **CB** and **SE** the quick navigation window, where a click on the symbol navigates to the symbol in the large window.

Symbol Browser (SB)

To find out more about a symbol with a specific name (here e.g. "**count**") there is the tool **SB** (**Symbol Browser**) available. Start the **SB** in the menu "**Tools->Symbol Browser**". In the project window select "**Select from All Projects**" using the mouse' right click. Here you may enter the string "**count**" and click the checkmark "**Whole Word**" to see there are several **count** symbols in the **pSOSystem** code. What each symbol looks like you may find out by double clicking so that the **SE** opens at the appropriate position.



Filter: count [✓ Whole Word]

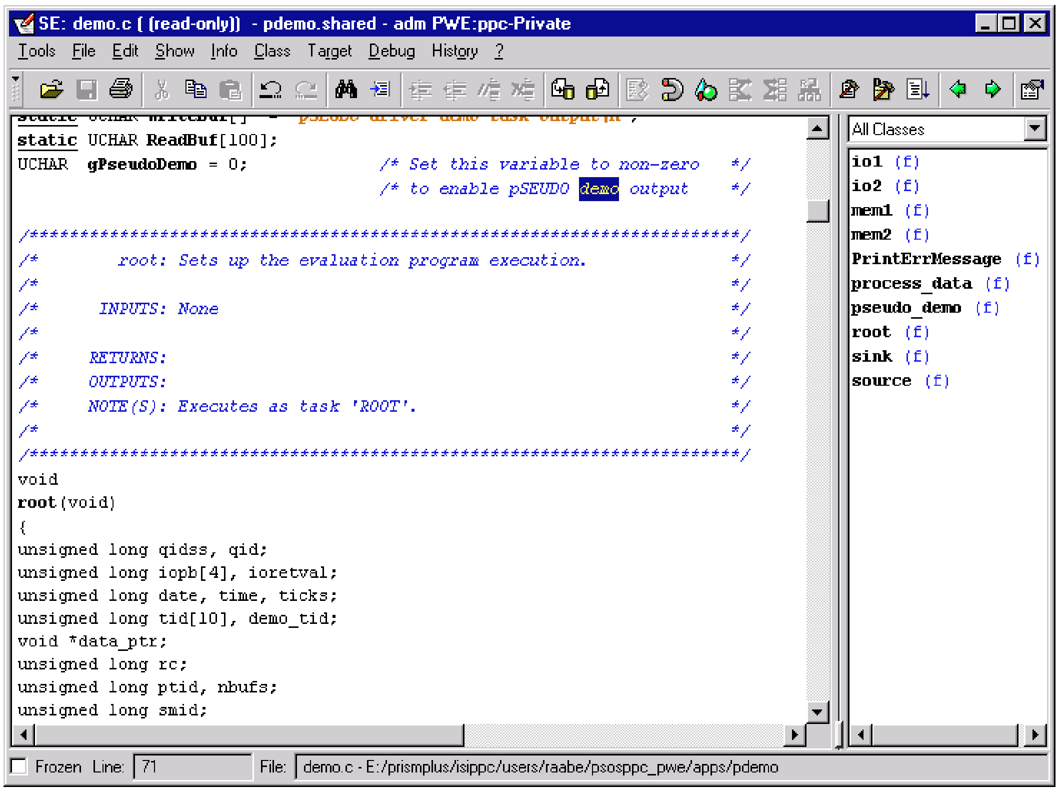
Symbols of pdemo.shared

	Symbol	Class
iv	count	nibrdcast
iv	count	TFTP_READ_IOPB
iv	count	nigetpkb
iv	count	nisend
iv	count	PssCharRdwrIOPB
iv	count	muinfo
iv	count	sminfo
iv	count	AX_CB_S
iv	count	FORMAT

Projects
- pdemo.shared (apps/pdemo)
- include.shared (include)
- cxx_include.shared (include/cxx)

Retriever (RE)

But what if there is no symbol with a specific name. The tool **RE** (**Retriever**) is a “graphical **grep** utility” which finds the occurrence of any string. If you are in the **SE** having “**demo.c**” open and you like to know, if there is a sub string of “**demo**” somewhere in the project, highlight it and ask **SNiFF+** by using the menu “**Info->Retrieve demo from All Projects**”.


SE: demo.c ((read-only)) - pdemo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes

```
static UCHAR ReadBuf[100];
UCHAR  gPseudoDemo = 0;          /* Set this variable to non-zero   */
                                 /* to enable pSEUDO demo output    */

/**********************************************************************/
/*       root: Sets up the evaluation program execution.              */
/*                                                                    */
/*      INPUTS: None                                                  */
/*                                                                    */
/*     RETURNS:                                                       */
/*     OUTPUTS:                                                       */
/*     NOTE(S): Executes as task 'ROOT'.                              */
/*                                                                    */
/**********************************************************************/
void
root(void)
{
unsigned long qidss, qid;
unsigned long iopb[4], ioretval;
unsigned long date, time, ticks;
unsigned long tid[10], demo_tid;
void *data_ptr;
unsigned long rc;
unsigned long ptid, nbufs;
unsigned long smid;
```

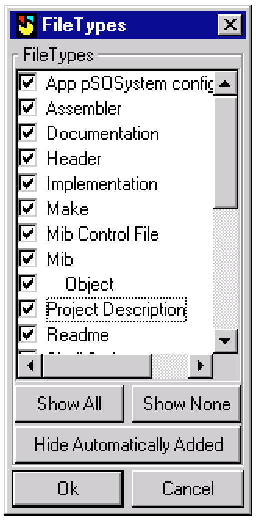
```
io1 (f)
io2 (f)
mem1 (f)
mem2 (f)
PrintErrMessage (f)
process_data (f)
pseudo_demo (f)
root (f)
sink (f)
source (f)
```

Frozen Line: 71 File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

IMPORTANT

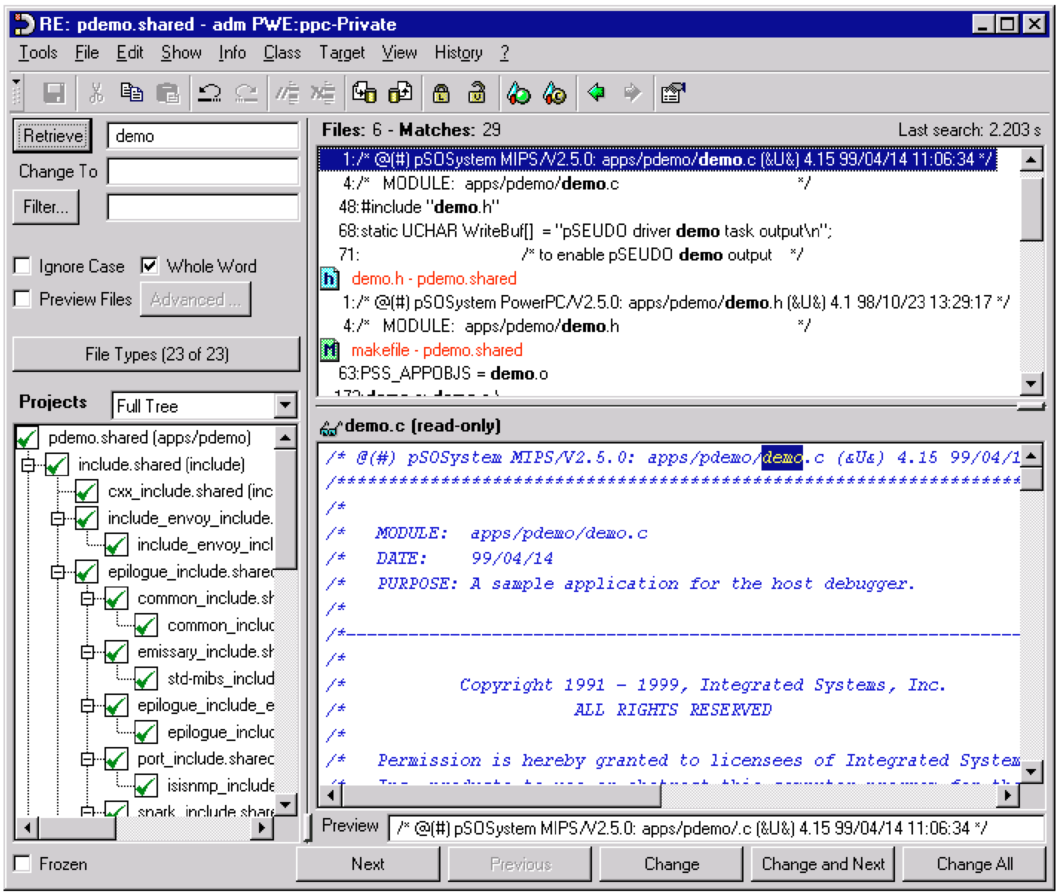
When the **RE** is started, not all file of the project are searched through. There are files of the type “**Added/Removed Automatically**” which are excluded by default. The **pSOSystem** include files are of such file type. To receive all the benefit of the **RE** you need to select all file types in the selector of the **RE**.

Press “**Show All**” and “**OK**” to select.



☑ App pSOSystem config
☑ Assembler
☑ Documentation
☑ Header
☑ Implementation
☑ Make
☑ Mib Control File
☑ Mib
☑ Object
☑ Project Description
☑ Readme

Checkmark “**Whole Word**” and press “**Retrieve**” to update the display.



Retrieve: demo

☐ Ignore Case ☑ Whole Word
☐ Preview Files
☐ Frozen

Projects: Full Tree

- ☑ pdemo.shared (apps/pdemo)
  - ☑ include.shared (include)
    - ☑ cxx_include.shared (inc
    - ☑ include_envoy_include.
      - ☑ include_envoy_incl
    - ☑ epilogue_include.sharec
      - ☑ common_include.sh
        - ☑ common_incluc
      - ☑ emissary_include.sh
        - ☑ std-mibs_includ
      - ☑ epilogue_include_e
        - ☑ epilogue_incluc
      - ☑ port_include.sharec
        - ☑ isisnmp_include
      - ☑ snark_include.share

Files: 6 - **Matches:** 29 Last search: 2.203 s

```
1:/* @(#) pSOSystem MIPS/V2.5.0: apps/pdemo/demo.c (&U&) 4.15 99/04/14 11:06:34 */
4:/*   MODULE:  apps/pdemo/demo.c                          */
48:#include "demo.h"
68:static UCHAR WriteBuf[] = "pSEUDO driver demo task output\n";
71:                   /* to enable pSEUDO demo output    */
demo.h - pdemo.shared
1:/* @(#) pSOSystem PowerPC/V2.5.0: apps/pdemo/demo.h (&U&) 4.1 98/10/23 13:29:17 */
4:/*   MODULE:  apps/pdemo/demo.h                          */
makefile - pdemo.shared
63:PSS_APPOBJS = demo.o
```

demo.c (read-only)

```
/* @(#) pSOSystem MIPS/V2.5.0: apps/pdemo/demo.c (&U&) 4.15 99/04/1
/*******************************************************************
/*
/*   MODULE:  apps/pdemo/demo.c
/*   DATE:     99/04/14
/*   PURPOSE: A sample application for the host debugger.
/*
/*------------------------------------------------------------------
/*
/*           Copyright 1991 - 1999, Integrated Systems, Inc.
/*                     ALL RIGHTS RESERVED
/*
/*   Permission is hereby granted to licensees of Integrated System
```

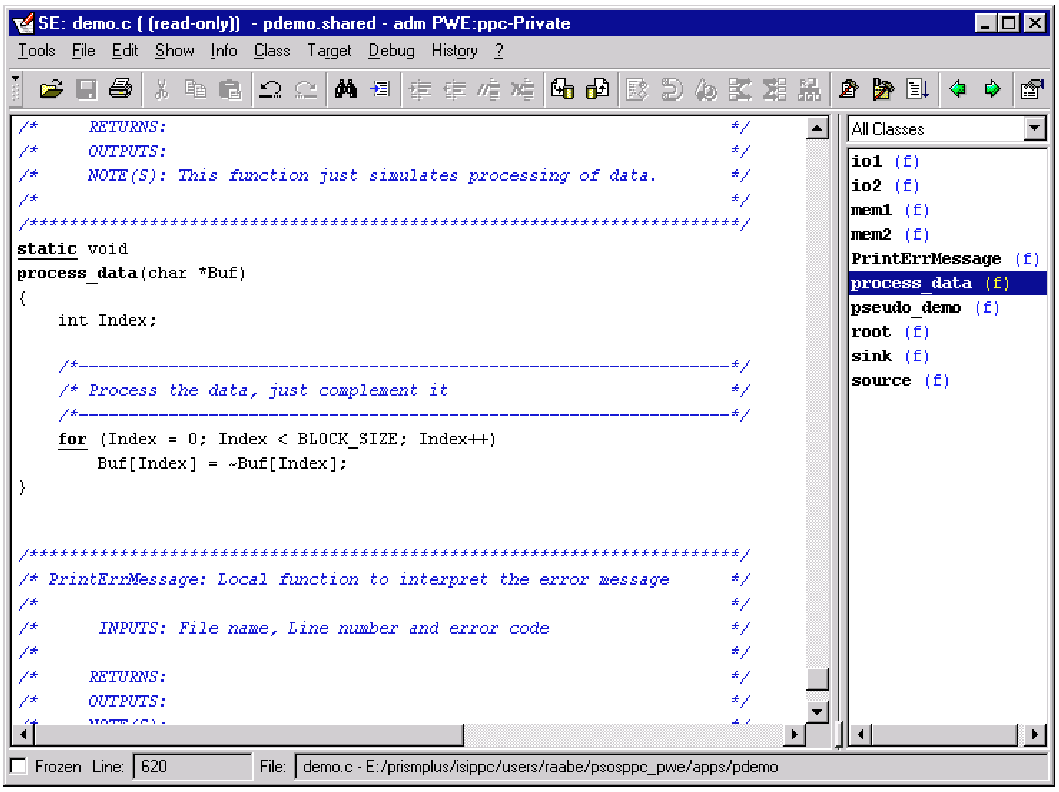
Preview: /* @(#) pSOSystem MIPS/V2.5.0: apps/pdemo/.c (&U&) 4.15 99/04/14 11:06:34 */

Prepare the compile and link run

To prepare for the compile and link run of the application we need to modify the **demo.c** file a bit. The **ESp** (**Embedded System profiler**) for example shows much more interesting information when adding the line “**tm_wkafter (10);**” at the end of the function “**process_data ()**”.

To achieve that, close all windows of **SNiFF+** except **Launch pad** and **PE**.

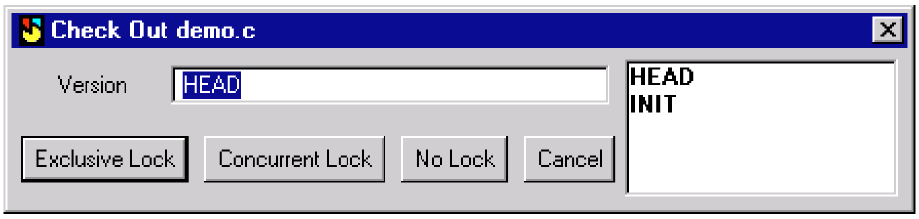
Double click “**demo.c**” in the **PE** to open the **SE**, navigate to “**process_data ()**” and position the cursor at the end of the function. “**demo.c**” is read-only (as to be seen in the **SE**’s title).


SE: demo.c ((read-only)) - pdemo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes
Frozen Line: 620 File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

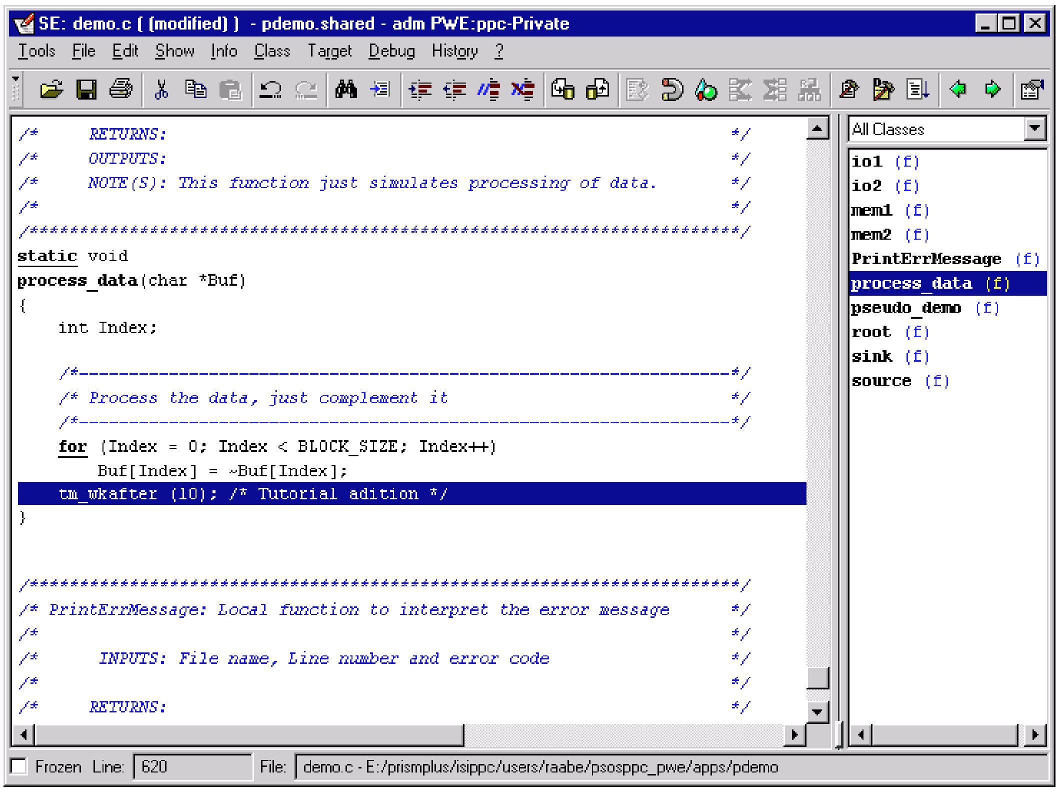
```
/*      RETURNS:                                                        */
/*      OUTPUTS:                                                        */
/*      NOTE(S): This function just simulates processing of data.       */
/*                                                                      */
/************************************************************************/
static void
process_data(char *Buf)
{
    int Index;

    /*------------------------------------------------------------------*/
    /* Process the data, just complement it                             */
    /*------------------------------------------------------------------*/
    for (Index = 0; Index < BLOCK_SIZE; Index++)
        Buf[Index] = ~Buf[Index];
}


/************************************************************************/
/* PrintErrMessage: Local function to interpret the error message       */
/*                                                                      */
/*       INPUTS: File name, Line number and error code                  */
/*                                                                      */
/*      RETURNS:                                                        */
/*      OUTPUTS:                                                        */
```

```
io1 (f)
io2 (f)
mem1 (f)
mem2 (f)
PrintErrMessage (f)
process_data (f)
pseudo_demo (f)
root (f)
sink (f)
source (f)
```

To be able to modify the file, we need to check out the file first by using the menu “**File->Check out…**” and selecting “**Exclusive Lock**” of “**HEAD**”.



Now add the line “ **tm_wkafter (10); /* Tutorial addition */**”.


SE: demo.c ((modified)) - pdemo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes

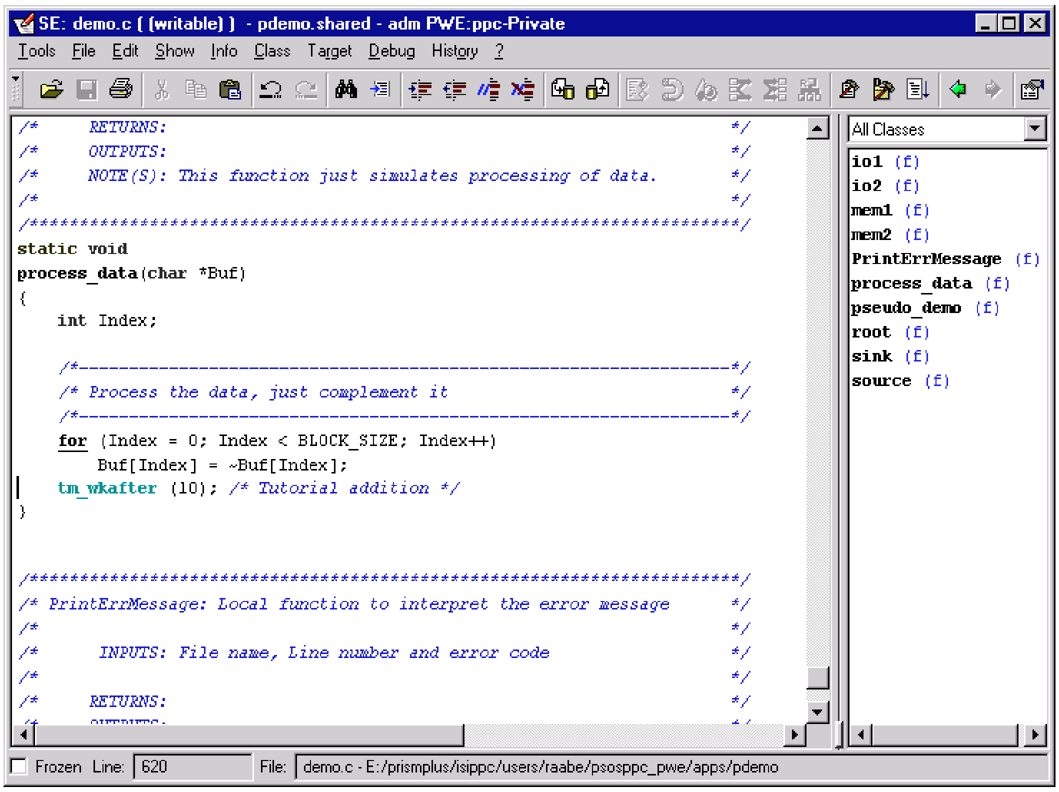
```
/*      RETURNS:                                                        */
/*      OUTPUTS:                                                        */
/*      NOTE(S): This function just simulates processing of data.       */
/*                                                                      */
/************************************************************************/
static void
process_data(char *Buf)
{
    int Index;

    /*------------------------------------------------------------------*/
    /* Process the data, just complement it                             */
    /*------------------------------------------------------------------*/
    for (Index = 0; Index < BLOCK_SIZE; Index++)
        Buf[Index] = ~Buf[Index];
    tm_wkafter (10); /* Tutorial adition */
}


/************************************************************************/
/* PrintErrMessage: Local function to interpret the error message       */
/*                                                                      */
/*       INPUTS: File name, Line number and error code                  */
/*                                                                      */
/*      RETURNS:                                                        */
```

```
io1  (f)
io2  (f)
mem1  (f)
mem2  (f)
PrintErrMessage  (f)
process_data  (f)
pseudo_demo  (f)
root  (f)
sink  (f)
source  (f)
```

Frozen Line: 620   File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

You might see, that the file is stated in the title as “**(modified)**” so save the file to disk by pressing the “**floppy disk**” icon (or on Windows the shortcut **Ctrl-s**).

Now the title states “**(writable)**” and the “**floppy disk**” icon is greyed out. Also the syntax colouring process completed. The comment is coloured in blue and the tm_wkafter () pSOSystem call is coloured in green.


SE: demo.c ((writable)) - pdemo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes
Frozen Line: 620
File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

```
/*      RETURNS:                                                          */
/*      OUTPUTS:                                                          */
/*      NOTE(S): This function just simulates processing of data.         */
/*                                                                        */
/************************************************************************/
static void
process_data(char *Buf)
{
    int Index;

    /*------------------------------------------------------------------*/
    /* Process the data, just complement it                             */
    /*------------------------------------------------------------------*/
    for (Index = 0; Index < BLOCK_SIZE; Index++)
        Buf[Index] = ~Buf[Index];
    tm_wkafter (10); /* Tutorial addition */
}


/************************************************************************/
/* PrintErrMessage: Local function to interpret the error message        */
/*                                                                        */
/*       INPUTS: File name, Line number and error code                    */
/*                                                                        */
/*      RETURNS:                                                          */
```

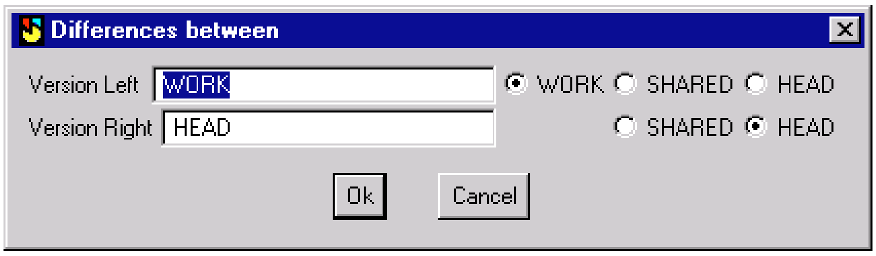
```
io1  (f)
io2  (f)
mem1  (f)
mem2  (f)
PrintErrMessage  (f)
process_data  (f)
pseudo_demo  (f)
root  (f)
sink  (f)
source  (f)
```

All **pSOSystem** API calls are coloured in different flavours of green, depending on the **pSOSystem** component they belong to (like **pSOS+**, **pROBE+**, **pREPC+**, **pNA+**, …)

DiffMerge (DM)

To find out what is the difference between the latest checked in version and the current version there is the **DM** (**DiffMerge Tool**) ready to serve. To activate it, use in the **SE** the menu "**File->Show Differences…**". The "**Differences between**" dialog starts and asks for the versions to compare. The version available for the **SE** (here the checked out one) is called "**WORK**" and the latest checked out is called "**HEAD**".

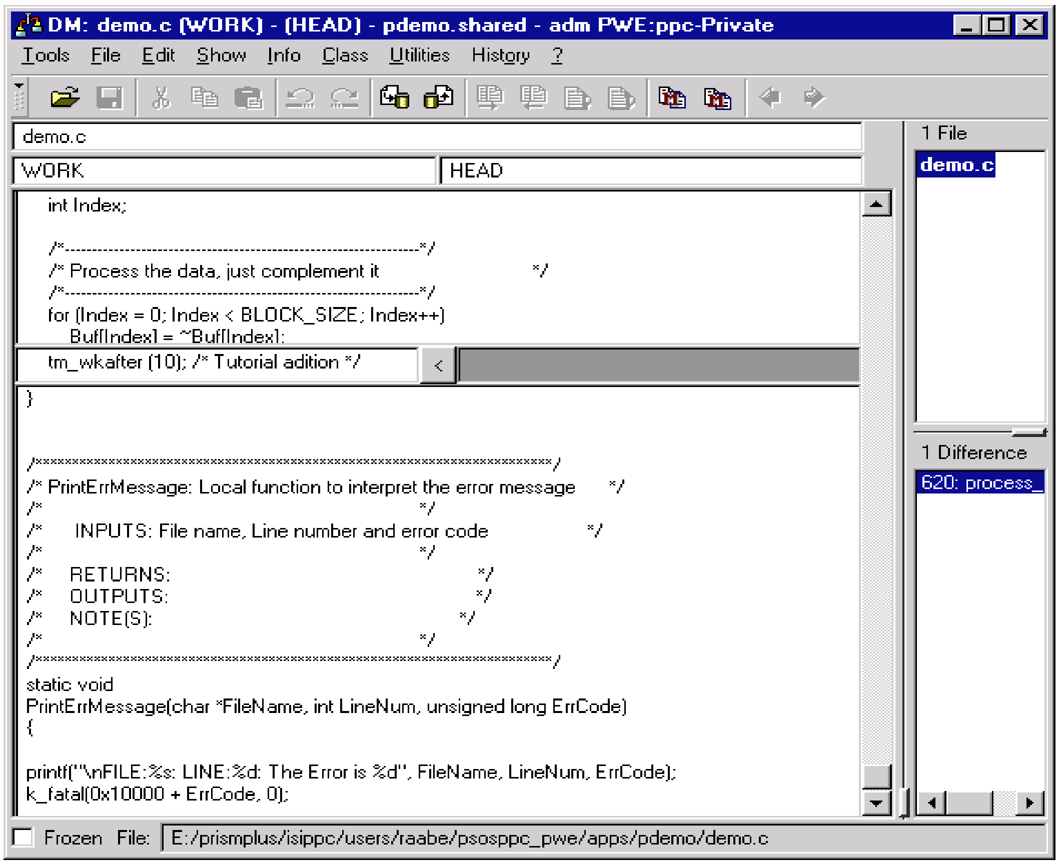
To compare those versions press "**OK**".


Differences between

Version Left: WORK — [● WORK] ○ SHARED ○ HEAD

Version Right: HEAD — ○ SHARED [● HEAD]

Now the **DM** shows all differences (here only one). The large window shows the source, the right windows show the location of the differences and the file names.


DM: demo.c (WORK) - (HEAD) - pdemo.shared - adm PWE:ppc-Private

demo.c

WORK | HEAD

```
   int Index;

   /*--------------------------------------------------------------*/
   /* Process the data, just complement it                         */
   /*--------------------------------------------------------------*/
   for (Index = 0; Index < BLOCK_SIZE; Index++)
      Buf[Index] = ~Buf[Index];
   tm_wkafter (10); /* Tutorial adition */
}


/***************************************************************************/
/* PrintErrMessage: Local function to interpret the error message         */
/*                                                                        */
/*     INPUTS: File name, Line number and error code                     */
/*                                                                        */
/*    RETURNS:                                                            */
/*    OUTPUTS:                                                            */
/*    NOTE(S):                                                            */
/*                                                                        */
/***************************************************************************/
static void
PrintErrMessage(char *FileName, int LineNum, unsigned long ErrCode)
{

printf("\nFILE:%s: LINE:%d: The Error is %d", FileName, LineNum, ErrCode);
k_fatal(0x10000 + ErrCode, 0);
```

1 File: demo.c

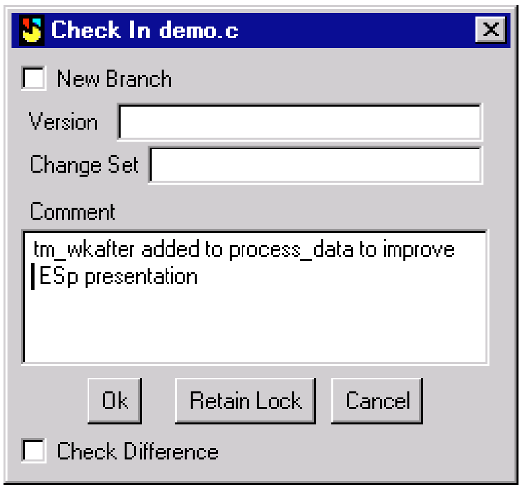
1 Difference: 620: process_

Frozen File: E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo/demo.c

Here you can see, that there was no previous text (greyed box) where now our additional line is.

The **DM** is also capable of merging, so if you want to have the version of the right source copied to the left source, you simply click the separating "**<**" button to merge!

To check in the file again, close the **DM** and use the **SE** menu "**File->Check In…**" to make a new version permanently stored in the Version Control subsystem (here **RCS**).



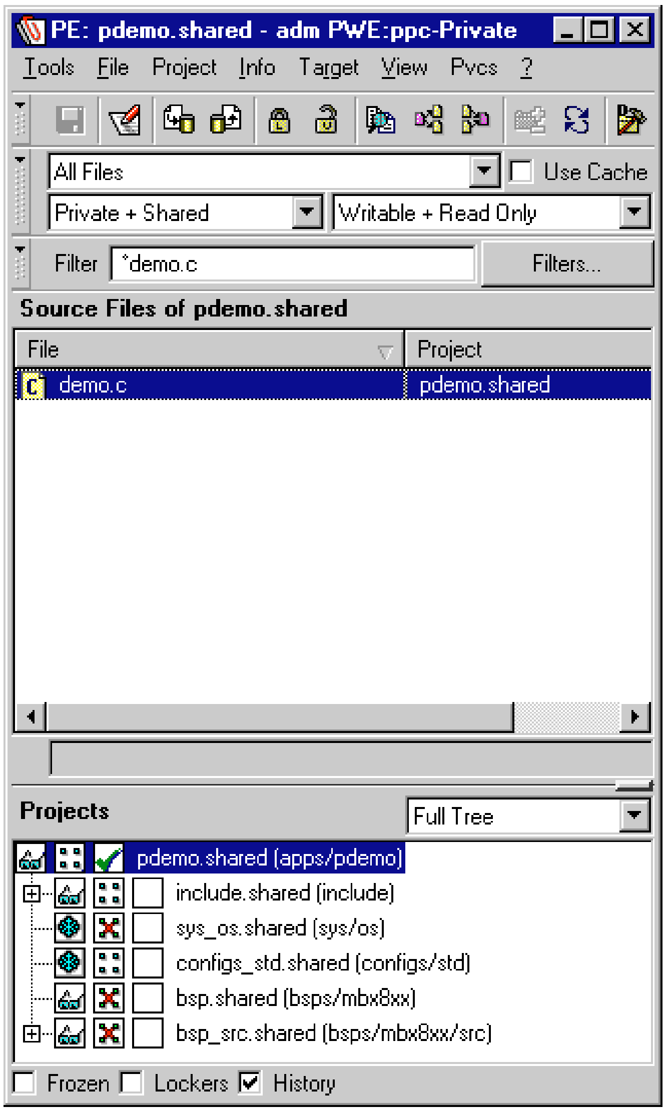
tm_wkafter added to process_data to improve
ESp presentation

You may enter a comment to have documented the reason for the changes.

The version number is increased in the less significant number by one per default, so leaving the **Version** field free, results in the version "1.2".

History (in PE)

To find out what versions have been created in the past use the **SE** menu “**File-Show History Info**”. The **PE** appears again on the screen with a helper window displaying the **history** information retrieved from the version control subsystem (here **RCS**).


PE: pdemo.shared - adm PWE:ppc-Private

All Files
Use Cache
Private + Shared
Writable + Read Only
Filter *demo.c

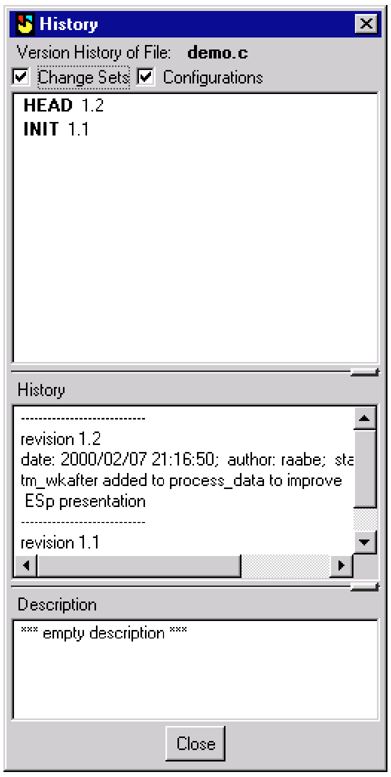
Source Files of pdemo.shared

File	Project
demo.c	pdemo.shared

Projects
Full Tree

pdemo.shared (apps/pdemo)
include.shared (include)
sys_os.shared (sys/os)
configs_std.shared (configs/std)
bsp.shared (bsps/mbx8xx)
bsp_src.shared (bsps/mbx8xx/src)

☐ Frozen ☐ Lockers ☑ History



HEAD 1.2
INIT 1.1

History

```
----------------------------
revision 1.2
date: 2000/02/07 21:16:50;  author: raabe;  sta
tm_wkafter added to process_data to improve
 ESp presentation
----------------------------
revision 1.1
```

Description

*** empty description ***

To remove this display, uncheck the "**History**" checkmark or press the "**close**" button of the "**History**" window. To see all files in the project, delete all characters in the "**Filter**" field and press the <**ENTER**> key. Now all files of "**pdemo.shared**" project are shown again.

Compile and Link run

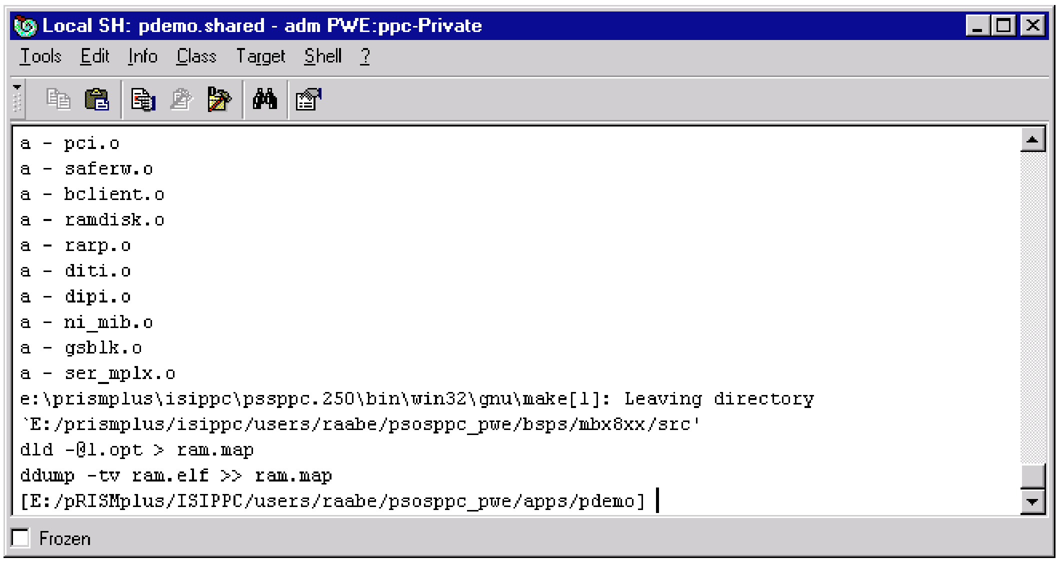
To generate a downloadable **pSOSystem** binary, start the compile and link run in the **SE** by the menu "**Target->Make->ram.elf**". A new tool opens up, the "**Local SH**" or **SH**. Here are all make-utility based activities executed. The make run is invoked by "**psosmake ram.elf**" which compiles all necessary files and links them to the binary "**ram.elf**". A link run results in a file called "**ram.map**" which containing the linker's symbol table.

The executables for other targets than PowerPC are:

68K and MIPS: ram.elf

ARM: ram.axf

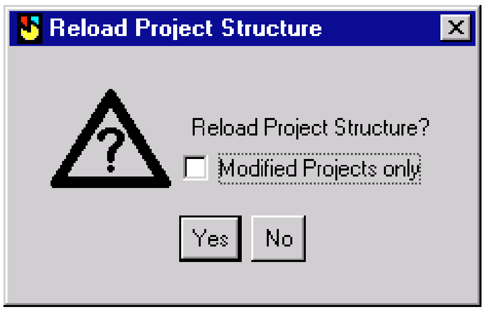
X86: ram.abs

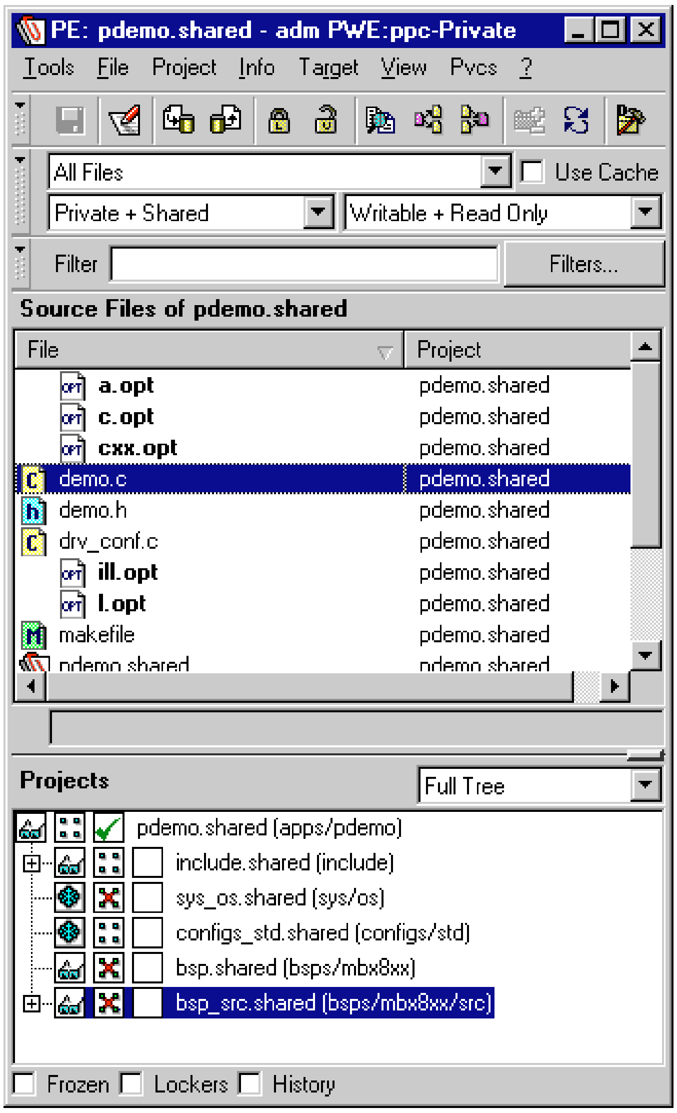


Local SH: pdemo.shared - adm PWE:ppc-Private
Tools Edit Info Class Target Shell ?

```
a - pci.o
a - saferw.o
a - bclient.o
a - ramdisk.o
a - rarp.o
a - diti.o
a - dipi.o
a - ni_mib.o
a - gsblk.o
a - ser_mplx.o
e:\prismplus\isippc\pssppc.250\bin\win32\gnu\make[1]: Leaving directory
`E:/prismplus/isippc/users/raabe/psosppc_pwe/bsps/mbx8xx/src'
dld -@l.opt > ram.map
ddump -tv ram.elf >> ram.map
[E:/pRISMplus/ISIPPC/users/raabe/psosppc_pwe/apps/pdemo]
```

To review this file, you should reload the project pdemo in the **PE** by check marking only "**pdemo.shared**" in the project sub window and using the menu "**Project->Reload Project-> In Current Working Environment**". A dialog asks you whether you want to reload all projects, you answer (without check marking) a "**YES**".




PE: pdemo.shared - adm PWE:ppc-Private

Tools File Project Info Target View Pvcs ?

All Files | Use Cache

Private + Shared | Writable + Read Only

Filter | Filters...

Source Files of pdemo.shared

File	Project
a.opt	pdemo.shared
c.opt	pdemo.shared
cxx.opt	pdemo.shared
demo.c	pdemo.shared
demo.h	pdemo.shared
drv_conf.c	pdemo.shared
ill.opt	pdemo.shared
l.opt	pdemo.shared
makefile	pdemo.shared
pdemo.shared	pdemo.shared

Projects | Full Tree

- pdemo.shared (apps/pdemo)
  - include.shared (include)
  - sys_os.shared (sys/os)
  - configs_std.shared (configs/std)
  - bsp.shared (bsps/mbx8xx)
  - bsp_src.shared (bsps/mbx8xx/src)

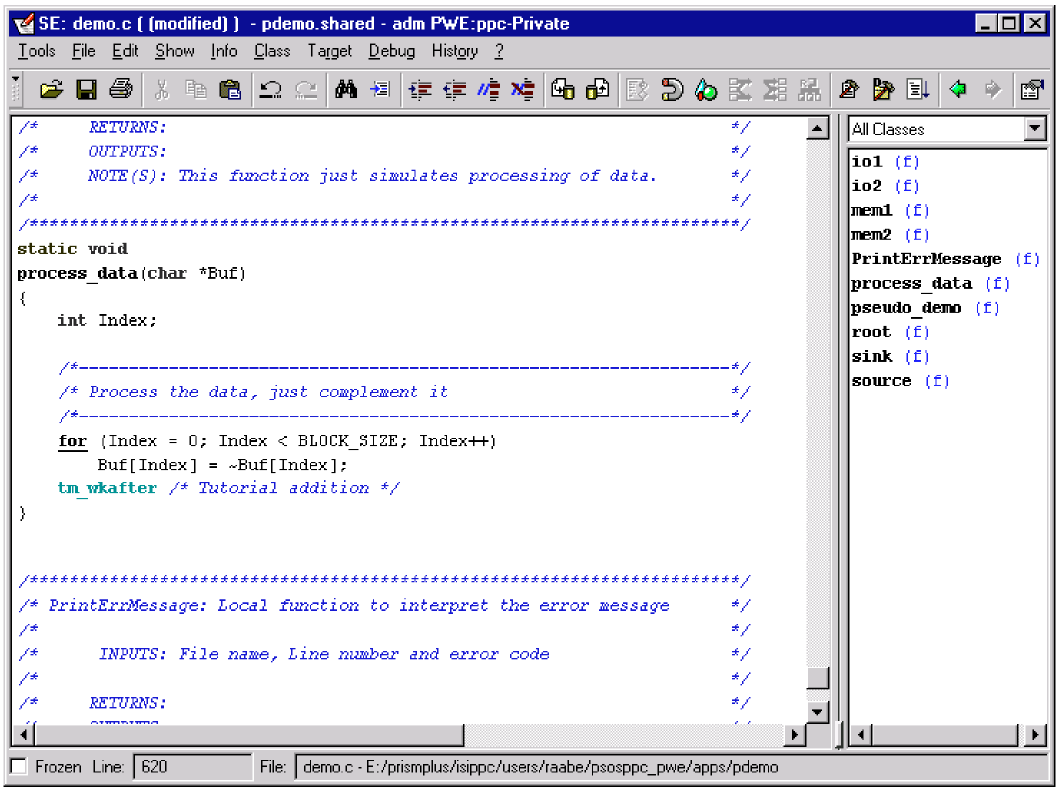
Frozen | Lockers | History

Now the **PE** contains new files (which are placed indented in the file window). There is "**ram.map**", containing the symbol table, but there is also the linker option file "**l.opt**". To review double-click the files you are interested in.

If you need more than one of the same tool open (e.g. two **SE**s), there is a "**Frozen**" checkmark at the lower border. When this checkmark is selected the tool opens a new window when a new tool invocation takes place.

Compile time Errors

If there is an error in the source Code you can easily jump to the erroneous location, correct it and recompile it. Go in the SE and modify the recently added line. (**First check out** and then modify!).

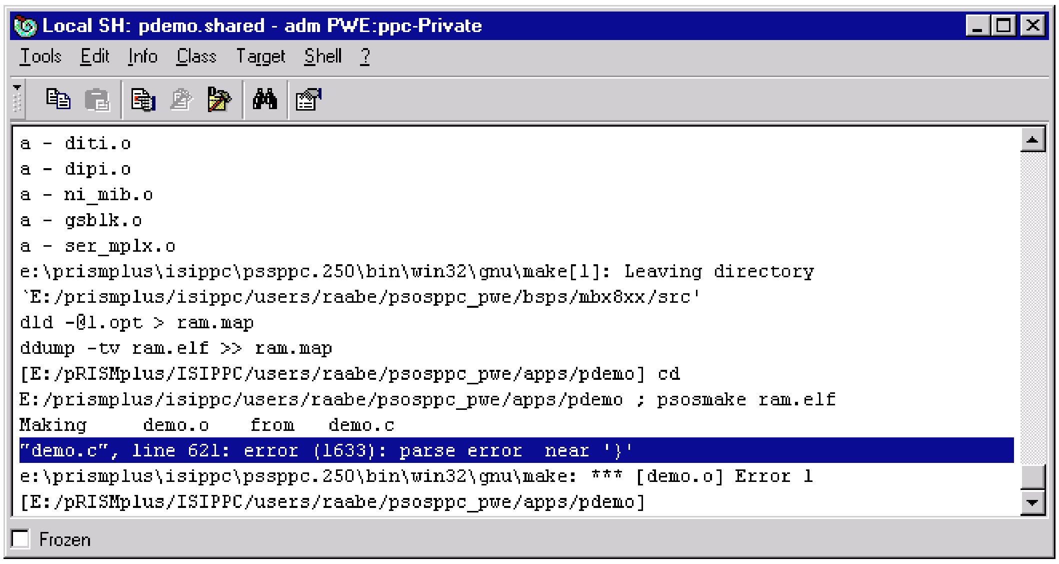

SE: demo.c ((modified)) - pdemo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes

```
/*     RETURNS:                                                          */
/*     OUTPUTS:                                                          */
/*     NOTE(S): This function just simulates processing of data.         */
/*                                                                       */
/*************************************************************************/
static void
process_data(char *Buf)
{
    int Index;

    /*-------------------------------------------------------------------*/
    /* Process the data, just complement it                              */
    /*-------------------------------------------------------------------*/
    for (Index = 0; Index < BLOCK_SIZE; Index++)
        Buf[Index] = ~Buf[Index];
    tm_wkafter /* Tutorial addition */
}


/*************************************************************************/
/* PrintErrMessage: Local function to interpret the error message        */
/*                                                                       */
/*      INPUTS: File name, Line number and error code                    */
/*                                                                       */
/*     RETURNS:                                                          */
```

```
io1 (f)
io2 (f)
mem1 (f)
mem2 (f)
PrintErrMessage (f)
process_data (f)
pseudo_demo (f)
root (f)
sink (f)
source (f)
```

Frozen Line: 620 File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

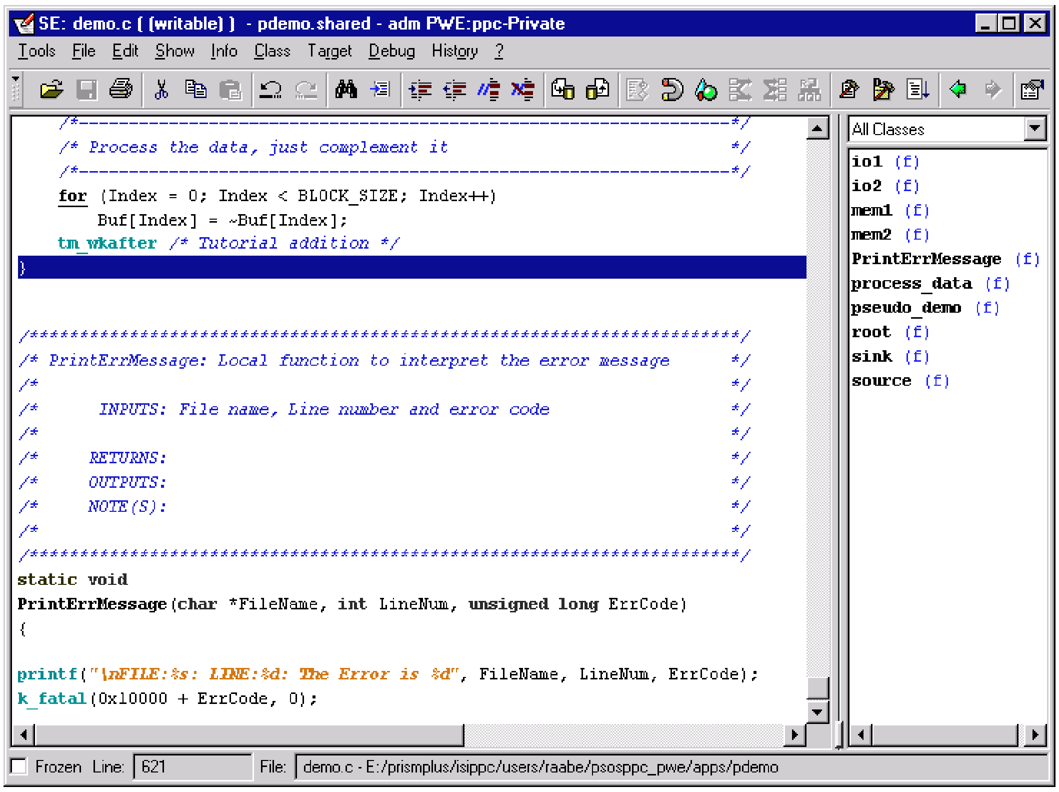
After saving and making the executable (by using the menu **Target->Make->ram.elf**) the **SH** displays an error.


Local SH: pdemo.shared - adm PWE:ppc-Private
Tools Edit Info Class Target Shell ?
Frozen

```
a - diti.o
a - dipi.o
a - ni_mib.o
a - gsblk.o
a - ser_mplx.o
e:\prismplus\isippc\psppc.250\bin\win32\gnu\make[1]: Leaving directory
`E:/prismplus/isippc/users/raabe/psosppc_pwe/bsps/mbx8xx/src'
dld -@l.opt > ram.map
ddump -tv ram.elf >> ram.map
[E:/pRISMplus/ISIPPC/users/raabe/psosppc_pwe/apps/pdemo] cd
E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo ; psosmake ram.elf
Making     demo.o    from    demo.c
"demo.c", line 621: error (1633): parse error  near '}'
e:\prismplus\isippc\psppc.250\bin\win32\gnu\make: *** [demo.o] Error 1
[E:/pRISMplus/ISIPPC/users/raabe/psosppc_pwe/apps/pdemo]
```

To mark in the SH output, you only need to place the cursor in the appropriate line. The highlighting is for demonstration purposes only!

The **SE** pops up and places the cursor to the appropriate location.

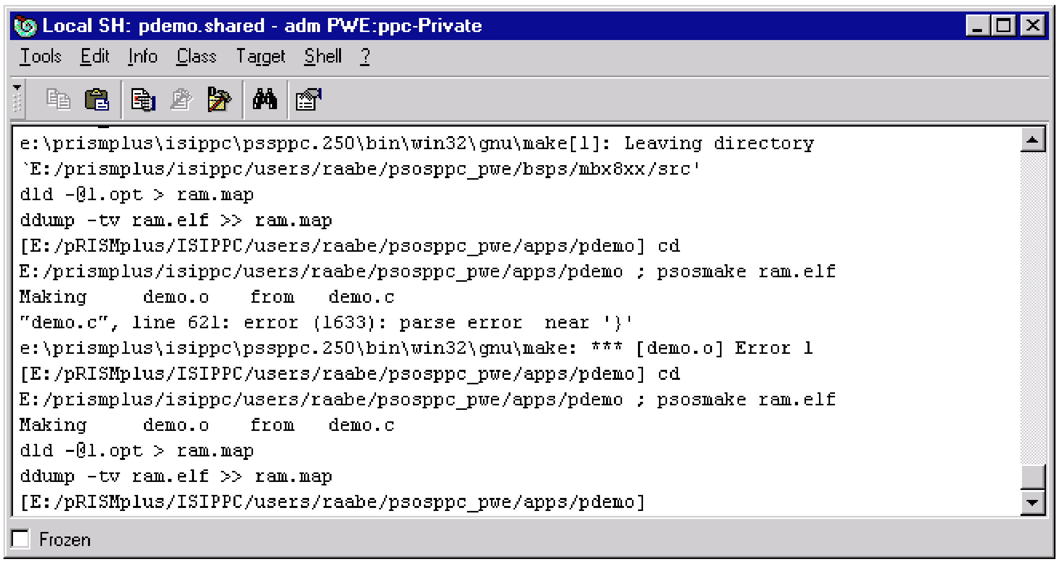

SE: demo.c ([writable]) - pdemo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes

```
    /*----------------------------------------------------------------*/
    /* Process the data, just complement it                           */
    /*----------------------------------------------------------------*/
    for (Index = 0; Index < BLOCK_SIZE; Index++)
        Buf[Index] = ~Buf[Index];
    tm_wkafter /* Tutorial addition */
}


/************************************************************************/
/* PrintErrMessage: Local function to interpret the error message      */
/*                                                                      */
/*       INPUTS: File name, Line number and error code                  */
/*                                                                      */
/*      RETURNS:                                                        */
/*      OUTPUTS:                                                        */
/*      NOTE(S):                                                        */
/*                                                                      */
/************************************************************************/
static void
PrintErrMessage(char *FileName, int LineNum, unsigned long ErrCode)
{

printf("\nFILE:%s: LINE:%d: The Error is %d", FileName, LineNum, ErrCode);
k_fatal(0x10000 + ErrCode, 0);
```

```
io1 (f)
io2 (f)
mem1 (f)
mem2 (f)
PrintErrMessage (f)
process_data (f)
pseudo_demo (f)
root (f)
sink (f)
source (f)
```

Frozen Line: 621 File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo

After correcting the error, you compile and link the executable again (as before).


Local SH: pdemo.shared - adm PWE:ppc-Private
Tools Edit Info Class Target Shell ?
Frozen

```
e:\prismplus\isippc\pssppc.250\bin\win32\gnu\make[1]: Leaving directory
`E:/prismplus/isippc/users/raabe/psosppc_pwe/bsps/mbx8xx/src'
dld -@l.opt > ram.map
ddump -tv ram.elf >> ram.map
[E:/pRISMplus/ISIPPC/users/raabe/psosppc_pwe/apps/pdemo] cd
E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo ; psosmake ram.elf
Making     demo.o    from   demo.c
"demo.c", line 621: error (1633): parse error  near '}'
e:\prismplus\isippc\pssppc.250\bin\win32\gnu\make: *** [demo.o] Error 1
[E:/pRISMplus/ISIPPC/users/raabe/psosppc_pwe/apps/pdemo] cd
E:/prismplus/isippc/users/raabe/psosppc_pwe/apps/pdemo ; psosmake ram.elf
Making     demo.o    from   demo.c
dld -@l.opt > ram.map
ddump -tv ram.elf >> ram.map
[E:/pRISMplus/ISIPPC/users/raabe/psosppc_pwe/apps/pdemo]
```

Windows shortcuts for the quick edit turnaround:
Ctrl-s (save)
Ctrl-m (make the default target)
arrow up key (locate the cursor to the error code of the compiler/linker)
Ctrl-g (Show error to jump into the **SE**)

Ctrl-m also automatically saves the file.

Further tools and details of SNiFF+

For further information about using **SNiFF+** tools please refer to the **SNiFF+** product documentation.

Switching the BSP:

Preparations

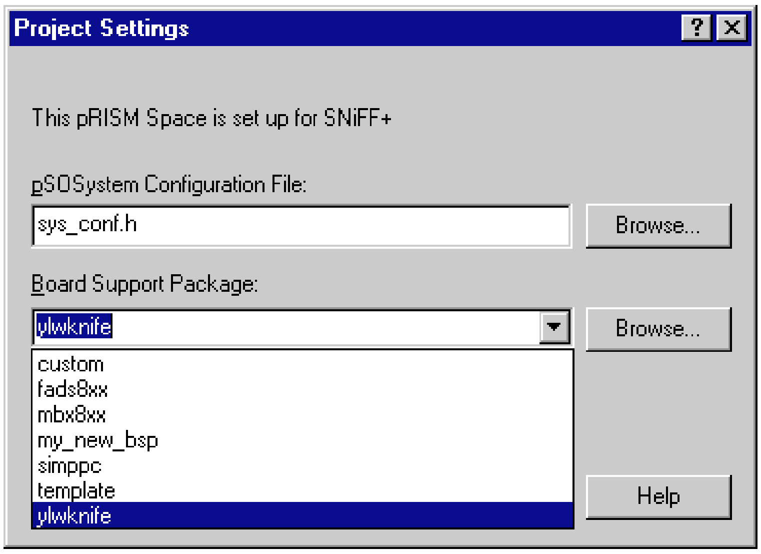
Have the pdemo application generated like in the previous chapter.

Closing SNiFF+

Close the **SNiFF+** Tool chain by using the menu (of any **SNiFF+** tool) “**Tools->Quit SNiFF+**”.

Selecting a new BSP

Select a new **BSP** in the **pRISM+** toolbar using the menu “**PrismSpace->Settings…**”.
(Here “**ylwknife**”). Press ”**OK**”.



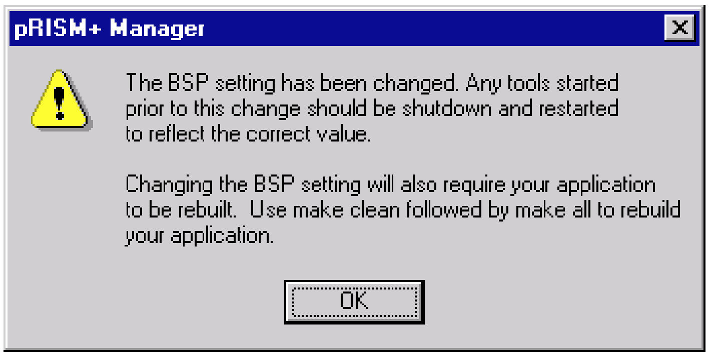
Project Settings

This pRISM Space is set up for SNiFF+

pSOSystem Configuration File: sys_conf.h

Board Support Package: ylwknife

An alert box opens up, reminding you to close any tool started from within the **pRISM+** toolbar like **SNiFF+** or **pWizard**. Press “**OK**”.



The BSP setting has been changed. Any tools started prior to this change should be shutdown and restarted to reflect the correct value.

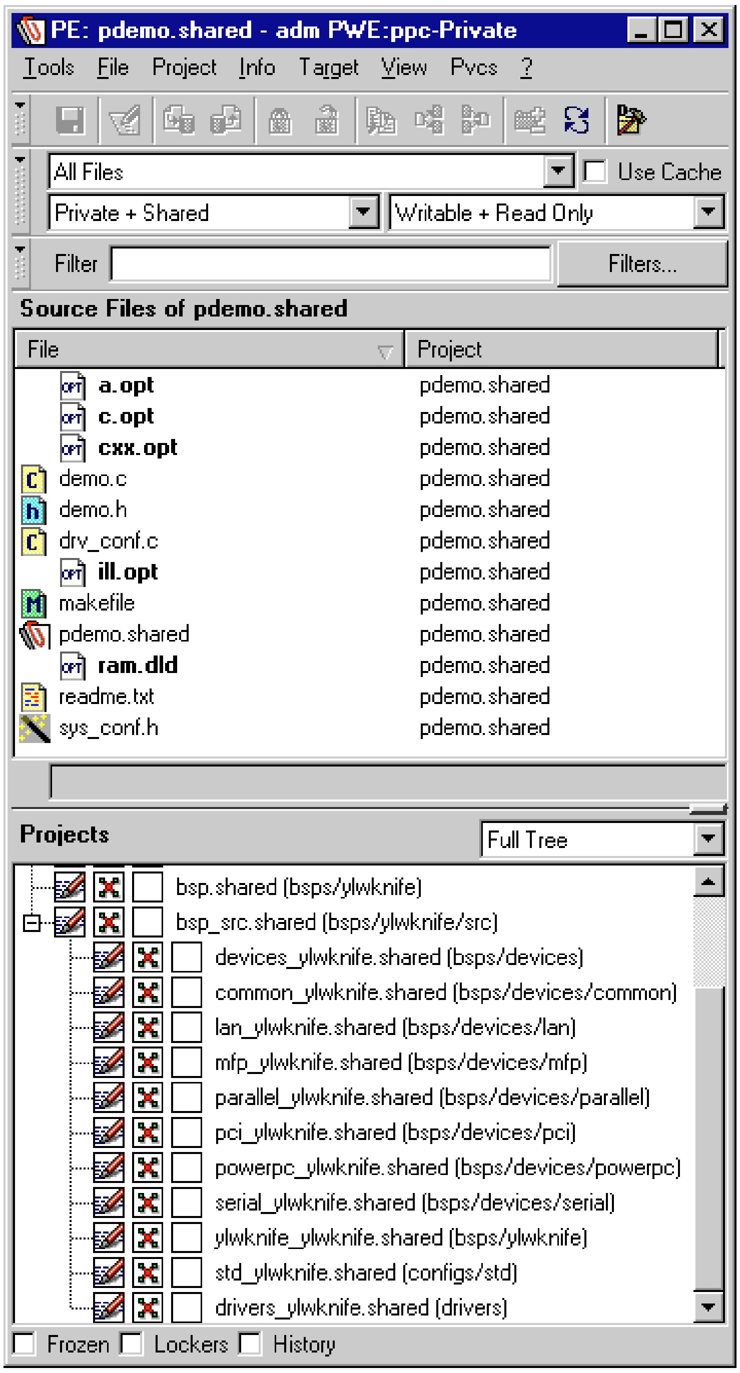
Changing the BSP setting will also require your application to be rebuilt. Use make clean followed by make all to rebuild your application.

Start **SNiFF+** again using the menu “**Tools->Project Editor**” or press the “**Activates SNiFF+**” button. The **PE** opens where you can see the “**bsp.shared**” and “**bsp_src.shared**” project containing the new **BSP** name.

To finish the process, regenerate the application by:

- Selecting the project bsp_src.shared
- Cleaning up the project by using the menu **Target->Make->clean**
- Selecting the project pdemo.shared
- Cleaning up the project by using the menu **Target->Make->clean**
- Compiling and linking the application by using the menu **Target->Make->ram.elf**

Now the **BSP** is switched as you can see in the **PE**'s project window.


PE: pdemo.shared - adm PWE:ppc-Private
Tools File Project Info Target View Pvcs ?
Filters...

All Files ☐ Use Cache
Private + Shared | Writable + Read Only
Filter

Source Files of pdemo.shared

File	Project
a.opt	pdemo.shared
c.opt	pdemo.shared
cxx.opt	pdemo.shared
demo.c	pdemo.shared
demo.h	pdemo.shared
drv_conf.c	pdemo.shared
ill.opt	pdemo.shared
makefile	pdemo.shared
pdemo.shared	pdemo.shared
ram.dld	pdemo.shared
readme.txt	pdemo.shared
sys_conf.h	pdemo.shared

Projects — Full Tree

- bsp.shared (bsps/ylwknife)
- bsp_src.shared (bsps/ylwknife/src)
  - devices_ylwknife.shared (bsps/devices)
  - common_ylwknife.shared (bsps/devices/common)
  - lan_ylwknife.shared (bsps/devices/lan)
  - mfp_ylwknife.shared (bsps/devices/mfp)
  - parallel_ylwknife.shared (bsps/devices/parallel)
  - pci_ylwknife.shared (bsps/devices/pci)
  - powerpc_ylwknife.shared (bsps/devices/powerpc)
  - serial_ylwknife.shared (bsps/devices/serial)
  - ylwknife_ylwknife.shared (bsps/ylwknife)
  - std_ylwknife.shared (configs/std)
  - drivers_ylwknife.shared (drivers)

☐ Frozen ☐ Lockers ☐ History

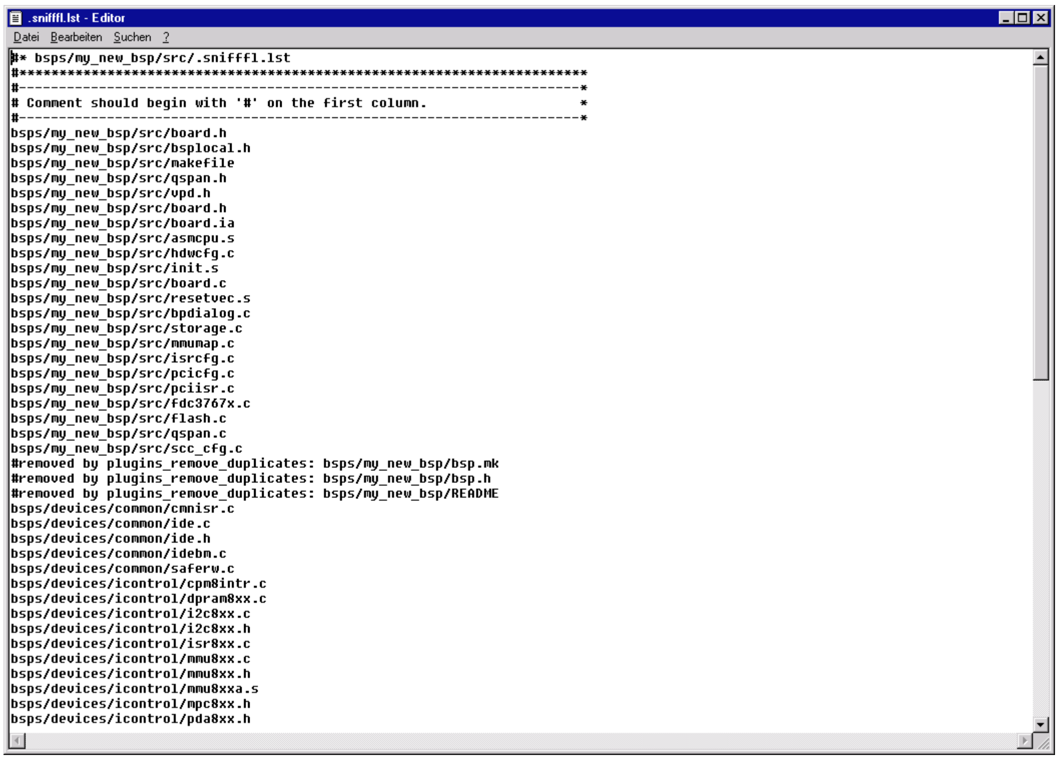
Adapting a custom BSP

To introduce an own **BSP**, there is the script "**plugins_create_bsp**". It is command-line based and needs to find a list called "**.snifffl.lst**" (read as **SNiFF File List**) of all files involved with this **BSP**. The **BSP** has to reside in a subdirectory of **$PSS_ROOT/bsps.** To edit the list use any editor. Place the list in the custom **BSP**'s "**src**" directory. Each file needed at compile time has to be named except the files in the custom **BSP**'s root directory (these files are included by default).

The file references need to be relative to the **$PSS_ROOT** directory (the directory where **pSOSystem** resides). You need write access to these directories! The first reference has to be in the **BSP**'s src directory. Any further file reference may be unsorted.

A good example may be to copy an existing **BSP** in **$PSS_ROOT/bsps** and to modify the file references in the file "**.snifffl.lst**".

After editing this list, open a Shell (not a SNiFF+ shell).



```
#* bsps/my_new_bsp/src/.snifffl.lst
#************************************************************************
#-----------------------------------------------------------------------*
# Comment should begin with '#' on the first column.                    *
#-----------------------------------------------------------------------*
bsps/my_new_bsp/src/board.h
bsps/my_new_bsp/src/bsplocal.h
bsps/my_new_bsp/src/makefile
bsps/my_new_bsp/src/qspan.h
bsps/my_new_bsp/src/vpd.h
bsps/my_new_bsp/src/board.h
bsps/my_new_bsp/src/board.ia
bsps/my_new_bsp/src/asmcpu.s
bsps/my_new_bsp/src/hdwcfg.c
bsps/my_new_bsp/src/init.s
bsps/my_new_bsp/src/board.c
bsps/my_new_bsp/src/resetvec.s
bsps/my_new_bsp/src/bpdialog.c
bsps/my_new_bsp/src/storage.c
bsps/my_new_bsp/src/mmumap.c
bsps/my_new_bsp/src/isrcfg.c
bsps/my_new_bsp/src/pcicfg.c
bsps/my_new_bsp/src/pciisr.c
bsps/my_new_bsp/src/fdc3767x.c
bsps/my_new_bsp/src/flash.c
bsps/my_new_bsp/src/qspan.c
bsps/my_new_bsp/src/scc_cfg.c
#removed by plugins_remove_duplicates: bsps/my_new_bsp/bsp.mk
#removed by plugins_remove_duplicates: bsps/my_new_bsp/bsp.h
#removed by plugins_remove_duplicates: bsps/my_new_bsp/README
bsps/devices/common/cmnisr.c
bsps/devices/common/ide.c
bsps/devices/common/ide.h
bsps/devices/common/idebm.c
bsps/devices/common/saferw.c
bsps/devices/icontrol/cpm8intr.c
bsps/devices/icontrol/dpram8xx.c
bsps/devices/icontrol/i2c8xx.c
bsps/devices/icontrol/i2c8xx.h
bsps/devices/icontrol/isr8xx.c
bsps/devices/icontrol/mmu8xx.c
bsps/devices/icontrol/mmu8xx.h
bsps/devices/icontrol/mmu8xxa.s
bsps/devices/icontrol/mpc8xx.h
bsps/devices/icontrol/pda8xx.h
```

For Windows use the Korn Shell form the system menu **Start->Programs->pRISM+->Utilities**.

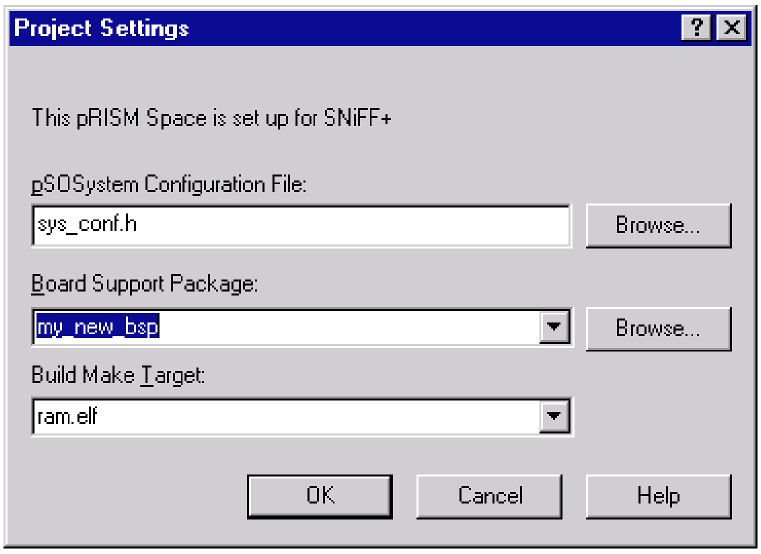
There should not be any directory called "**.sniffdir**" or "**sniffprj**" in the **BSP**'s subdirectories. If there are those directories, delete them. They get now regenerated. **Caution:** The standard **BSP** files have the file attribute "**read-only**" set. To edit the "**.snifffl.lst**", you need to write enable the attributes by using "**chmod +w .snifffl.lst**".

Start the script “**plugins_create_bsp** <**BSP’s pathname**>” where <**BSP’s pathname**> is the absolute path to the **BSP**’s root directory. (E.g. **plugins_create_bsp $PSS_ROOT/bsps/my_new_bsp**)

SNiFF+ is started and shut down automatically by the script.

```
[e:/pRISMplus/ISIPPC] plus/isippc/pssppc.250/bsps/my_new_bsp
Starting SNiFF+ session0.....
#-----------------------------------------------------------------------*
Creating project bsp_src.shared
                Creating uniq dir list
                Creating sniff project for bsps/devices
                Creating sniff project for bsps/devices/common
                Creating sniff project for bsps/devices/icontrol
                Creating sniff project for bsps/devices/lan
                Creating sniff project for bsps/devices/mfp
                Creating sniff project for bsps/devices/pci
                Creating sniff project for bsps/devices/powerpc
                Creating sniff project for bsps/devices/serial
                Creating sniff project for drivers
                Creating main_proj under bsps/my_new_bsp/src
                        It may take a while, please be patient..
Project bsp_src.shared for my_new_bsp.. done
Creating bsp.shared for my_new_bsp.. done
#-----------------------------------------------------------------------*
Shutting SNiFF+ down..
[e:/pRISMplus/ISIPPC]
```

Now you may select the new **BSP** in the **pRISM+ Tool bar**.


Project Settings

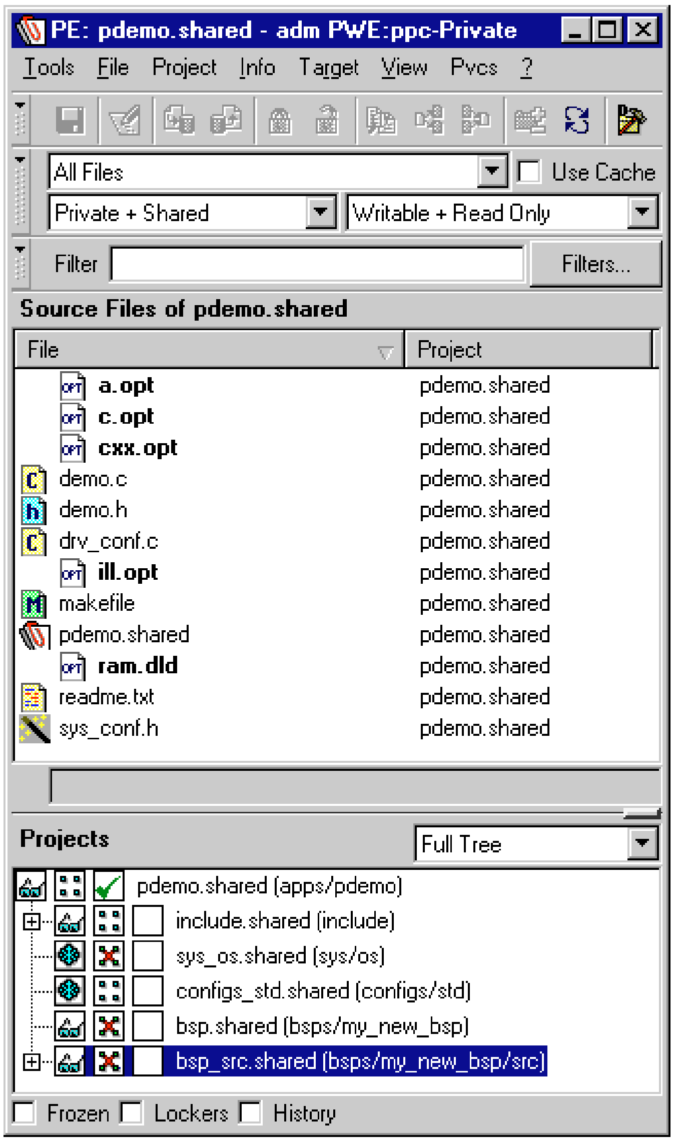
This pRISM Space is set up for SNiFF+

pSOSystem Configuration File: sys_conf.h

Board Support Package: my_new_bsp

Build Make Target: ram.elf

Start **SNiFF+** and compile and link the executable as described in the preceding sections!


PE: pdemo.shared - adm PWE:ppc-Private

All Files | Use Cache
Private + Shared | Writable + Read Only
Filter | Filters...

Source Files of pdemo.shared

File	Project
a.opt	pdemo.shared
c.opt	pdemo.shared
cxx.opt	pdemo.shared
demo.c	pdemo.shared
demo.h	pdemo.shared
drv_conf.c	pdemo.shared
ill.opt	pdemo.shared
makefile	pdemo.shared
pdemo.shared	pdemo.shared
ram.dld	pdemo.shared
readme.txt	pdemo.shared
sys_conf.h	pdemo.shared

Projects | Full Tree

```
pdemo.shared (apps/pdemo)
  include.shared (include)
  sys_os.shared (sys/os)
  configs_std.shared (configs/std)
  bsp.shared (bsps/my_new_bsp)
  bsp_src.shared (bsps/my_new_bsp/src)
```

Frozen | Lockers | History

Since the some existing BSPs do not mention all files needed to compile in .snifffl.lst, especially those files located in the directories drivers and devices, there is a switch to refer to those files based either on **$PSS_ROOT** or **$PSS_USER_PWE**. This switch is selectable in the file **$PSS_ROOT/bin/win32/psosmake.ksh**. For further information refer to the comments in this file and the **ReleaseNotes.txt**.

Team development and standard pSOSystem applications

You should not do team development using standard application environment.

The **pSOSystem** files reside in a write-protected area where updating of this Shared Source Working Environment is prohibited.

To use the standard **pSOSystem** applications for team development,

- Copy the application from **$PSS_ROOT/apps/<application_to_copy>** to the area of the User's Shared Source Working Environment (**$PSS_USER_SSWE**) into a new directory.
- Delete the files "**.snifffl.lst**", "**drv_conf.h**", "**sys_conf.h**" and "**makefile**".
- Open the pRISM+ project as described in the following chapter (Starting with existing code base).

If there is a special setting in "**drv_conf.h**" or "**sys_conf.h**", copy the files after generation of the project into the "**pss_main**" project, force the reparse of the project and recompile.

Starting with an existing codebase (user_pdemo):

Preparations

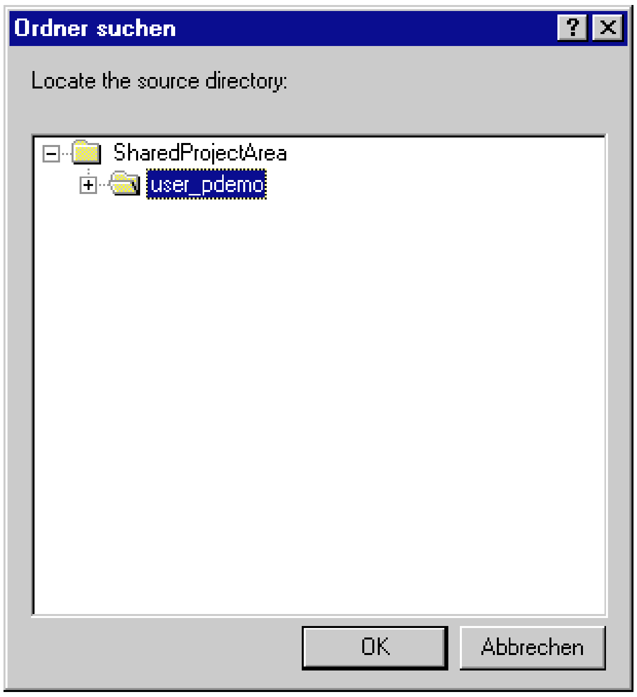
If you already generated the application user_pdemo, delete in the directory of the Private Working Environment **$PSS_USER_PWE** the directories "**user_pdemo**" and "**.ProjectCache**". Also delete in the directory of the Shared Source Working Environment **$PSS_USER_SSWE** the directories "**user_pdemo**" and "**.ProjectCache**". Now we can start in a clean environment.

The demo application **user_pdemo** is designed to help to understand, how to handle existing codebase. It is derived from the standard application **pdemo**.

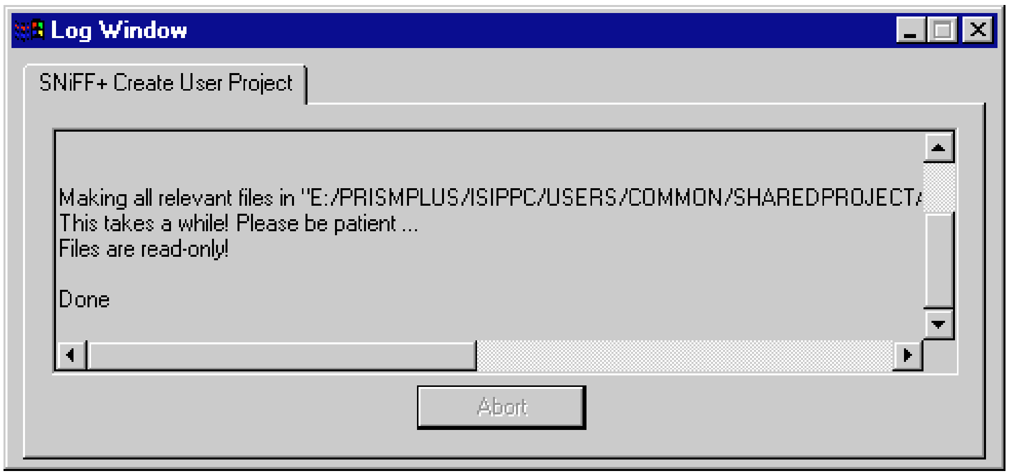
It is not designed to start from here a team-based development.

Generation of a new project …

Generate the user_pdemo example project using **pRISM+ Toolbar** as described in the **Tutorial** (**File->New->SNiFF+->Next->Starting with an existing codebase->Next->Browse->user_pdemo->OK->Next->Next->Finish**).



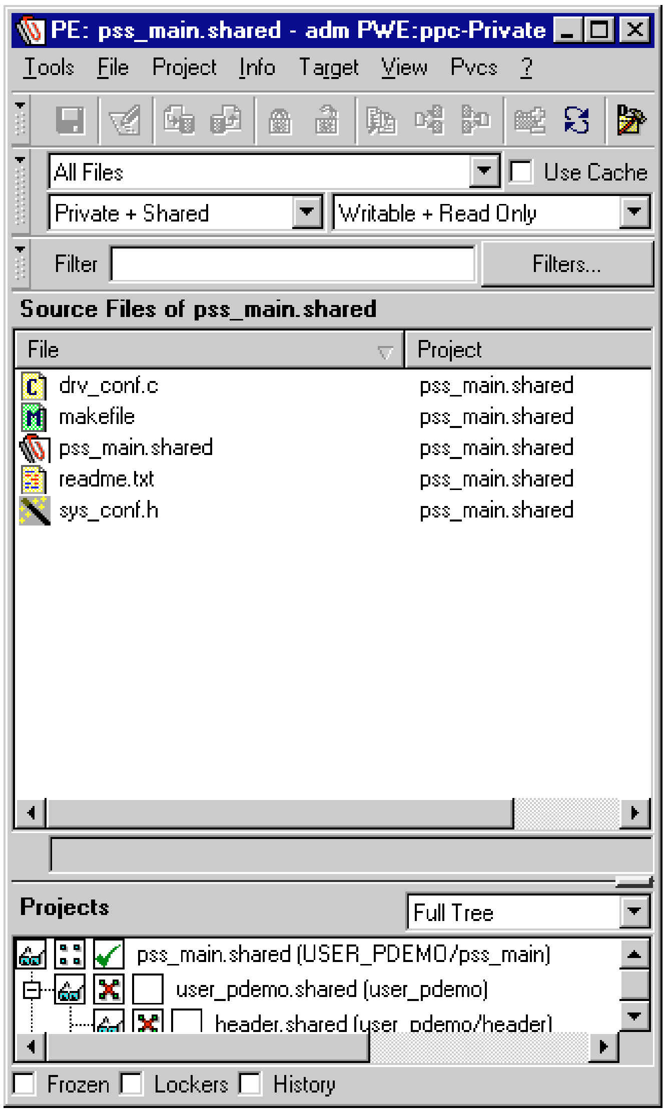
The Log Window of **pRISM+** opens, if **pRISM+** is freshly started or you did not close it in the meantime. You may close it. The text in the window states the progress of the **SNiFF+** open process, which is at an end, when the "**Abort**" button is greyed out and the "**Done**" text is displayed.



```
Making all relevant files in "E:/PRISMPLUS/ISIPPC/USERS/COMMON/SHAREDPROJECT/
This takes a while! Please be patient ...
Files are read-only!

Done
```

Project Editor (PE)

Now the SNiFF+ V3.2 **Launch pad** appears and the **PE** (Project Editor) gets displayed:


PE: pss_main.shared - adm PWE:ppc-Private

All Files ☐ Use Cache
Private + Shared | Writable + Read Only
Filter | Filters...

Source Files of pss_main.shared

File	Project
drv_conf.c	pss_main.shared
makefile	pss_main.shared
pss_main.shared	pss_main.shared
readme.txt	pss_main.shared
sys_conf.h	pss_main.shared

Projects | Full Tree

```
✓ pss_main.shared (USER_PDEMO/pss_main)
☐ user_pdemo.shared (user_pdemo)
  ☐ header.shared (user_pdemo/header)
```

☐ Frozen ☐ Lockers ☐ History

The **PE** has two window sections. The upper one is displaying the files of the project, selected in the lower window, the project window with the checkmark.

The file attributes of all relevant files of the existing codebase are set to read-only.

The project tree may be collapsed or expanded by clicking on the "-" respectively "+" symbol in front of every project. The projects of the standard **pSOSystem** applications come with predefined attributes:

The application and **BSP** (Board Support Package) parts of the project are configured to be private to the user and are marked writable. (See the pencil icon in the first column.)

The other projects are containing code related to **pSOSystem** real time operating system (**RTOS**) and are global to all users (Shared). For that reason, some **pSOSystem** related projects (**sys_os.shared** and **config_std.shared**) are marked **read-only**. (See the "frozen" icon in the first column.)

Version Control (CMVC)

The Version Control Tool by default is **RCS**.

On Windows hosts, the files of the private part of the projects are copied to a private directory, the private workspace. It contains the private working environment. The Shell environment variable **$PSS_USER_PWE** points to that location. On Unix hosts symbolic links are used to reduce the amount of disk space (Windows does not support symbolic links).

It is highly recommended to use version control to help to find out about the history of a file, set of files or the complete project tree. It is easy to use, since the version control is designed in the GUI of **SNiFF+.**

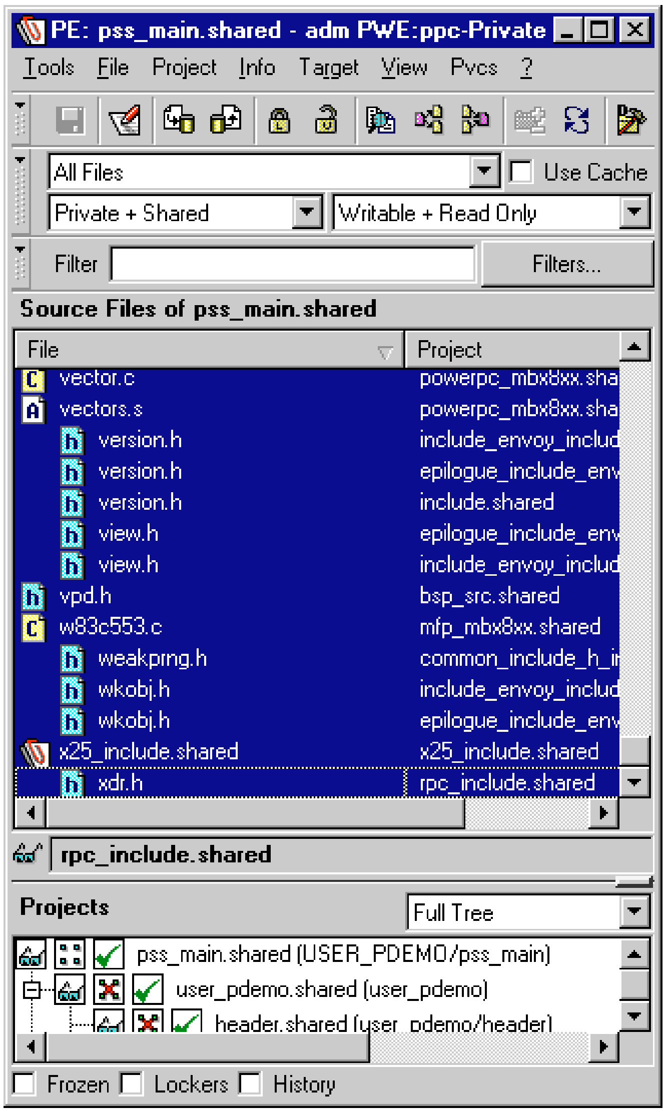
Usually the single developer checks in all files for the first time and checks out the file(s) he/she needs to modify.

When the implementation is tested well enough, the modified version(s) of the file(s) is (are) checked in again with a comment attached. This helps in the maintenance phase of a project.

To find out about the status of each file there is a checkmark in the **PE** to display this information instantly ("**Lockers**"). To also instantly display the history of the file selected in the file window of the **PE** there is the button "**History**" available.

The buttons "**Lockers**" and "**History**" are not intended for permanent use, since they decrease the overall performance of the SNiFF+ development environment.

To check in all files, select a project in the project window and use the context menu to "**Select from All Projects**". Now every project has a green checkmark and a lot of files are displayed in the file window. Select a file in the file window and use the context menu to "**Select All**" files.



PE: pss_main.shared - adm PWE:ppc-Private

All Files · Use Cache: ☐
Private + Shared · Writable + Read Only
Filter: · Filters...

Source Files of pss_main.shared

File	Project
vector.c	powerpc_mbx8xx.sha
vectors.s	powerpc_mbx8xx.sha
version.h	include_envoy_includ
version.h	epilogue_include_env
version.h	include.shared
view.h	epilogue_include_env
view.h	include_envoy_includ
vpd.h	bsp_src.shared
w83c553.c	mfp_mbx8xx.shared
weakprng.h	common_include_h_i
wkobj.h	include_envoy_includ
wkobj.h	epilogue_include_env
x25_include.shared	x25_include.shared
xdr.h	rpc_include.shared

rpc_include.shared

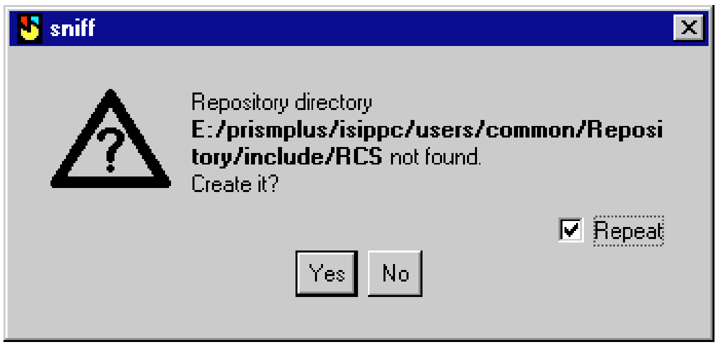
Projects · Full Tree

pss_main.shared (USER_PDEMO/pss_main)
user_pdemo.shared (user_pdemo)
header.shared (user_pdemo/header)

☐ Frozen ☐ Lockers ☐ History

Check in all files by using the context menu “**Check In Files…**”.

You get asked to generate the repository directory tree, which is representing the directory structure of your entire project. Checkmark the "**Repeat**" button and answer with "**Yes**".

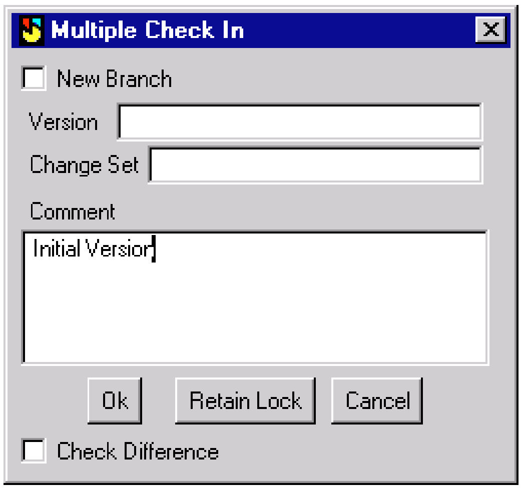

sniff

Repository directory **E:/prismplus/isippc/users/common/Repository/include/RCS** not found.
Create it?

[✓] Repeat

The version control asks for a comment of this initial version. You may also enter a starting version number, default value is **"1.1"**. Acknowledge the dialog and all files get checked in now.

> If you already have checked in the **pSOSystem** related files and the **BSP**, check in only the files of the projects "**pss_main.shared**" and "**user_pdemo.shared**". The check in of all files takes some time (2-15 minutes).


Multiple Check In

[ ] New Branch

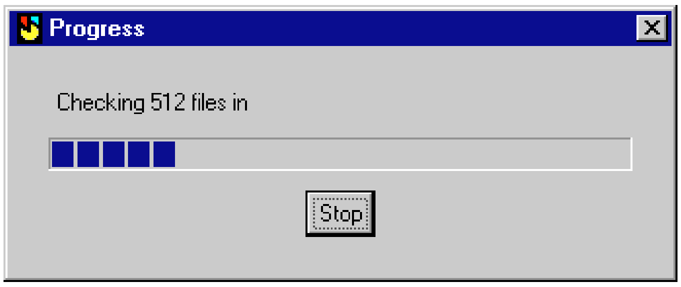
Version

Change Set

Comment

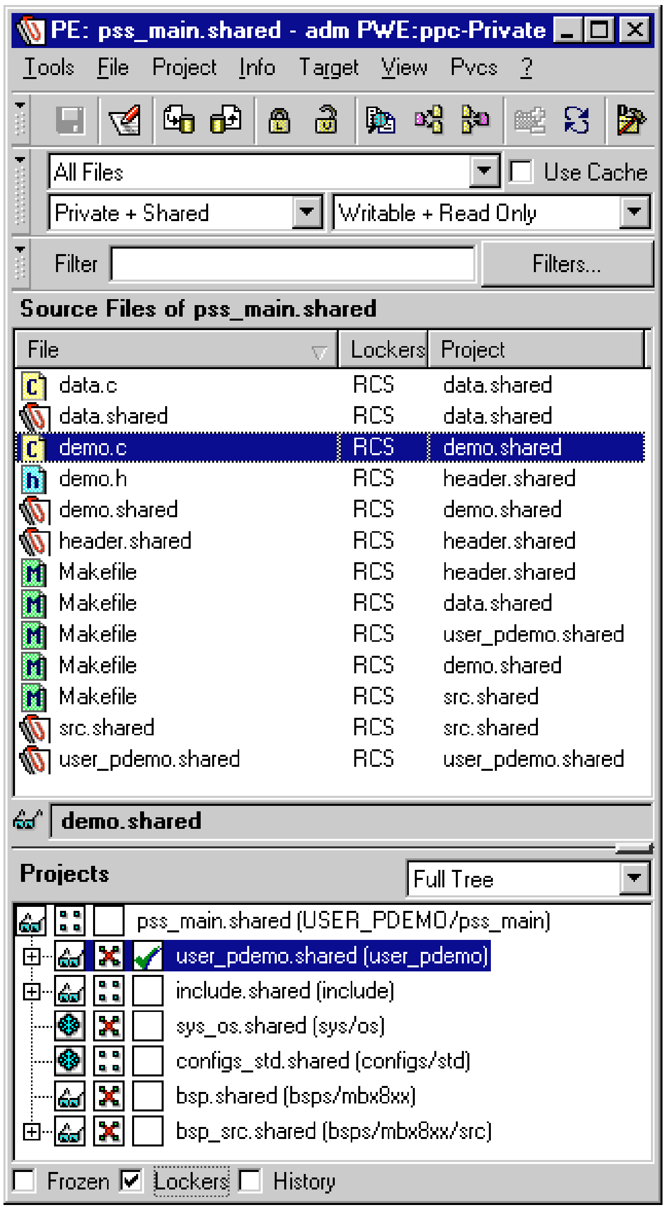
Initial Version

[ ] Check Difference


Progress

Checking 512 files in

To verify the success of the check in, checkmark the “**Lockers**” button at the bottom line of the **PE** window and the file window should contain an additional third column displaying the version control tool you use. For a better demonstration effect, we select the “**user_demo.shared**“ project only (using the context menu) and the file “**demo.c**” by selecting it):



PE: pss_main.shared - adm PWE:ppc-Private

Source Files of pss_main.shared

File	Lockers	Project
data.c	RCS	data.shared
data.shared	RCS	data.shared
demo.c	RCS	demo.shared
demo.h	RCS	header.shared
demo.shared	RCS	demo.shared
header.shared	RCS	header.shared
Makefile	RCS	header.shared
Makefile	RCS	data.shared
Makefile	RCS	user_pdemo.shared
Makefile	RCS	demo.shared
Makefile	RCS	src.shared
src.shared	RCS	src.shared
user_pdemo.shared	RCS	user_pdemo.shared

demo.shared

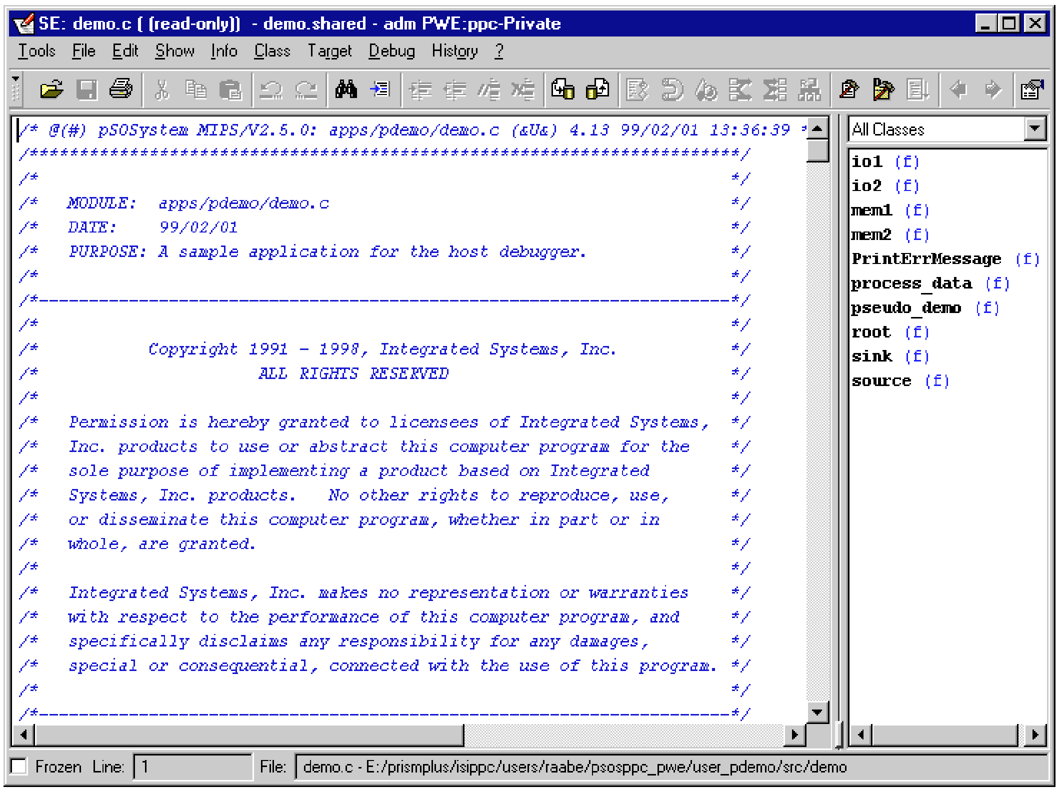
Projects — Full Tree

- pss_main.shared (USER_PDEMO/pss_main)
  - [✓] user_pdemo.shared (user_pdemo)
  - include.shared (include)
  - sys_os.shared (sys/os)
  - configs_std.shared (configs/std)
  - bsp.shared (bsps/mbx8xx)
  - bsp_src.shared (bsps/mbx8xx/src)

Frozen [✓ Lockers] History

To take a closer look at the **user_pdemo** application, please refer to the **pSOSystem** documentation, **pRISM+ Online Help** (**Help->Help Topics, How To …->pRISM+ Tutorial**) and the file “**readme.txt**”.

To find out about the application we start the **SE** (Source Editor) by double clicking the file “**demo.c**”, which contains the root task and other tasks, demonstrating most of the inter task communication mechanisms. The title of the **SE** shows, that the file is read-only and any attempt of adding new characters is refused. To make this file writable, you need to check the file out of the version control tool using the context menu “**Check Out File**”.


SE: demo.c ([read-only]) - demo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes

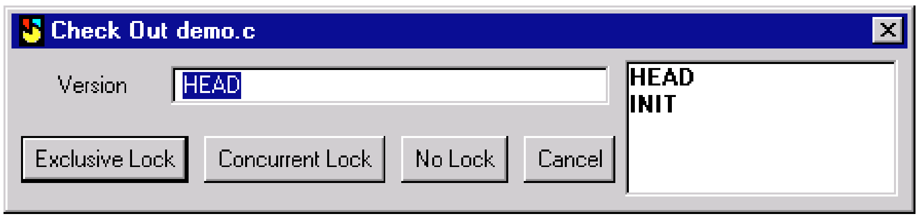
```
/* @(#) pSOSystem MIPS/V2.5.0: apps/pdemo/demo.c (&U&) 4.13 99/02/01 13:36:39 *
/***************************************************************************/
/*                                                                         */
/*   MODULE:  apps/pdemo/demo.c                                            */
/*   DATE:     99/02/01                                                    */
/*   PURPOSE: A sample application for the host debugger.                  */
/*                                                                         */
/*-------------------------------------------------------------------------*/
/*                                                                         */
/*          Copyright 1991 - 1998, Integrated Systems, Inc.                */
/*                      ALL RIGHTS RESERVED                                */
/*                                                                         */
/*   Permission is hereby granted to licensees of Integrated Systems,      */
/*   Inc. products to use or abstract this computer program for the        */
/*   sole purpose of implementing a product based on Integrated            */
/*   Systems, Inc. products.   No other rights to reproduce, use,          */
/*   or disseminate this computer program, whether in part or in           */
/*   whole, are granted.                                                   */
/*                                                                         */
/*   Integrated Systems, Inc. makes no representation or warranties        */
/*   with respect to the performance of this computer program, and         */
/*   specifically disclaims any responsibility for any damages,            */
/*   special or consequential, connected with the use of this program.     */
/*                                                                         */
/*-------------------------------------------------------------------------*/
```

```
io1 (f)
io2 (f)
mem1 (f)
mem2 (f)
PrintErrMessage (f)
process_data (f)
pseudo_demo (f)
root (f)
sink (f)
source (f)
```

Frozen Line: 1 File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/user_pdemo/src/demo

You get asked, how to check out what version.
Select “**Exclusive Lock**”, so the system grants you the exclusive right to modify the file.

This is helpful if you attempt to check out again later on, then you get warned, that the file is already checked out.



In our case we get offered to check out the version “**HEAD**”, which is the desired version.

If you have already checked in more versions, then the Name “**HEAD**” is always the latest version and the version “**INIT**” is the oldest one.

Now the title of the **SE** shows, the file is writable.

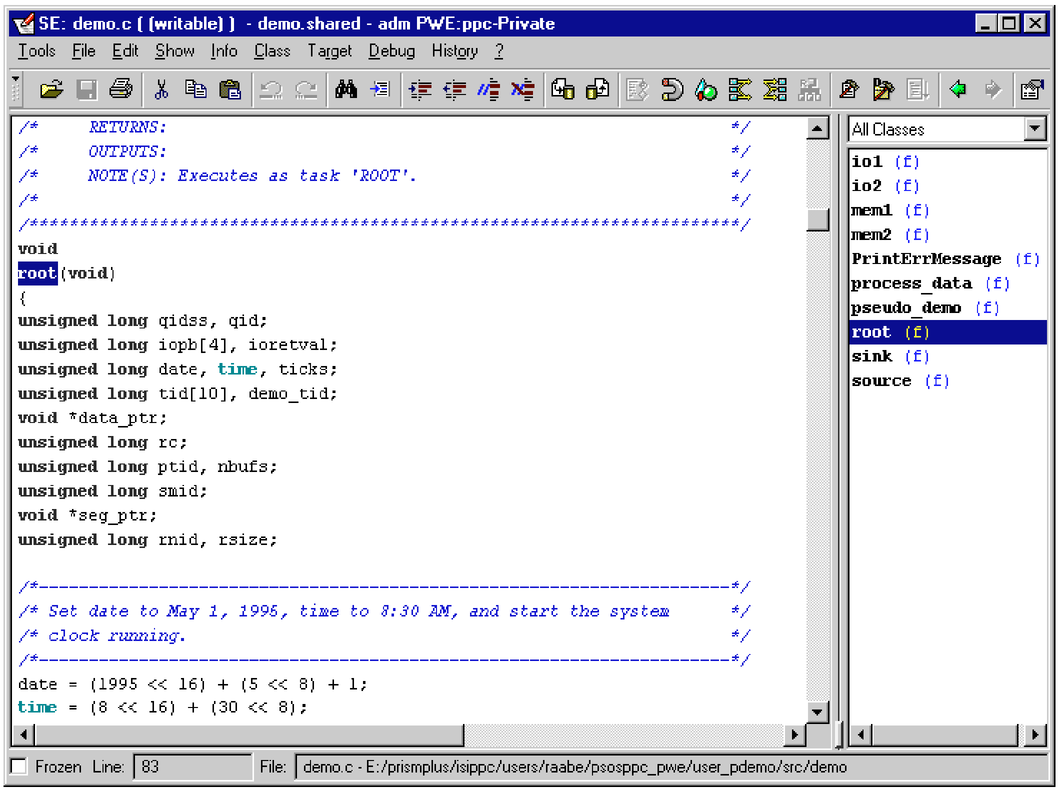
… and Browsing and understanding Source Code

Source Editor (SE)

First we look at the **SE**. There is the source file displayed in the main window displaying the file and there is a smaller window on the right displaying additional information on basis of a symbol table.

The symbol table is generated on creation time of the projects and when a modification is saved. So the information is ready to use whenever you save your file(s).

The right window is the quick navigation window and shows all C-functions in this file. The first function called by **pSOSystem** is “**root ()**”. To quickly navigate to **root ()** click on the text **root (f)** in the right window and the code is displayed.

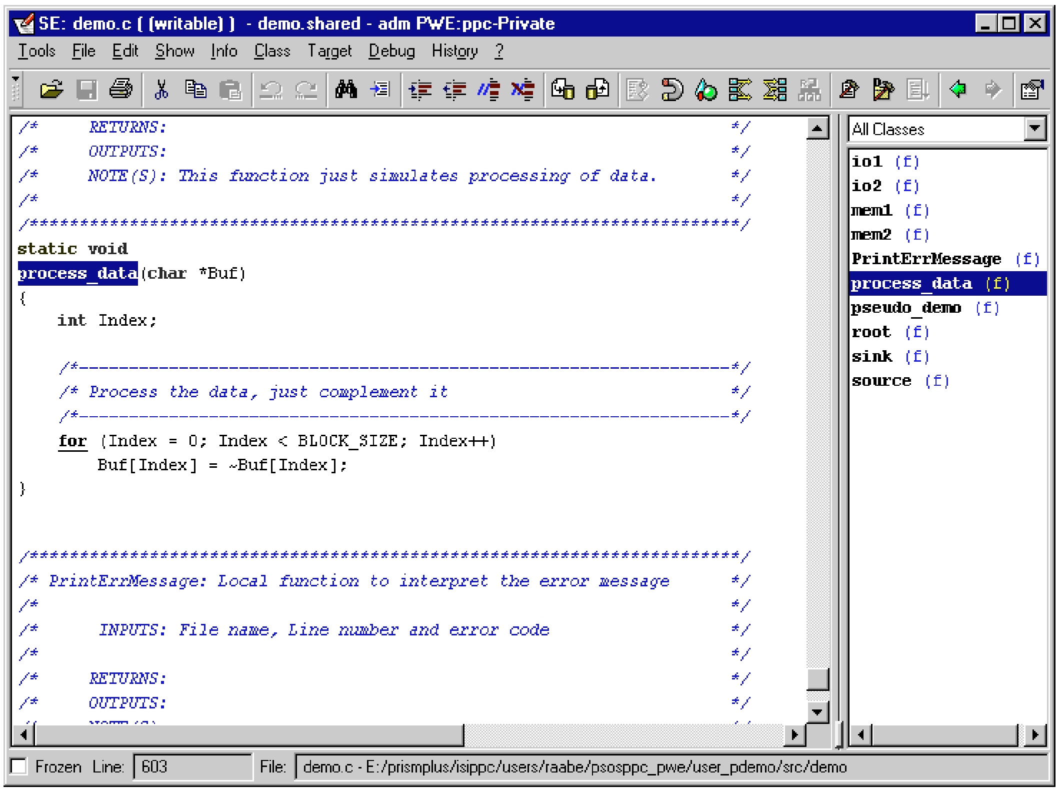

SE: demo.c ((writable)) - demo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes
Frozen Line: 83
File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/user_pdemo/src/demo

```
/*      RETURNS:                                                        */
/*      OUTPUTS:                                                        */
/*      NOTE(S): Executes as task 'ROOT'.                               */
/*                                                                      */
/************************************************************************/
void
root(void)
{
unsigned long qidss, qid;
unsigned long iopb[4], ioretval;
unsigned long date, time, ticks;
unsigned long tid[10], demo_tid;
void *data_ptr;
unsigned long rc;
unsigned long ptid, nbufs;
unsigned long smid;
void *seg_ptr;
unsigned long rnid, rsize;

/*----------------------------------------------------------------------*/
/* Set date to May 1, 1995, time to 8:30 AM, and start the system      */
/* clock running.                                                       */
/*----------------------------------------------------------------------*/
date = (1995 << 16) + (5 << 8) + 1;
time = (8 << 16) + (30 << 8);
```

```
io1  (f)
io2  (f)
mem1  (f)
mem2  (f)
PrintErrMessage  (f)
process_data  (f)
pseudo_demo  (f)
root  (f)
sink  (f)
source  (f)
```

In the quick navigation window you also see there is a function "**process_data ()**" shown. Click on it and in the main window the function name is highlighted.

> When you are in the **SE** and have a file open, there is a shortcut (Ctrl-E on Windows) or a menu entry (**Show->Header File** when in the source of an implementation file or **Show->Implementation File** when in the source of an header file) to switch between the implementation and header file back and forth.


SE: demo.c ((writable)) - demo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes
Frozen Line:

```
/*     RETURNS:                                                        */
/*     OUTPUTS:                                                        */
/*     NOTE(S): This function just simulates processing of data.      */
/*                                                                     */
/***********************************************************************/
static void
process_data(char *Buf)
{
    int Index;

    /*-----------------------------------------------------------------*/
    /* Process the data, just complement it                            */
    /*-----------------------------------------------------------------*/
    for (Index = 0; Index < BLOCK_SIZE; Index++)
        Buf[Index] = ~Buf[Index];
}


/***********************************************************************/
/* PrintErrMessage: Local function to interpret the error message      */
/*                                                                     */
/*      INPUTS: File name, Line number and error code                  */
/*                                                                     */
/*     RETURNS:                                                        */
/*     OUTPUTS:                                                        */
```

```
io1 (f)
io2 (f)
mem1 (f)
mem2 (f)
PrintErrMessage (f)
process_data (f)
pseudo_demo (f)
root (f)
sink (f)
source (f)
```

Line: 603 File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/user_pdemo/src/demo

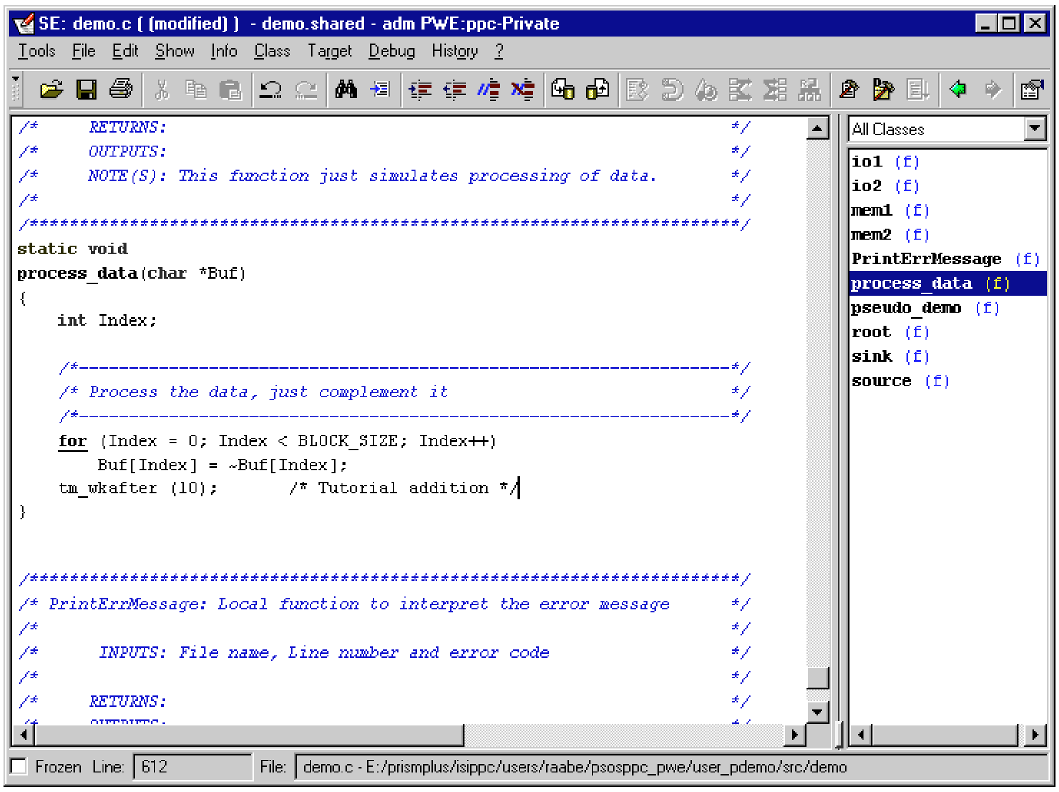
Other SNiFF+ Tools

To find out about the other Browsing tools offered by SNiFF+ please refer to the section "Starting with a standard pSOSystem application" in this document!

Prepare the compile and link run

To prepare for the compile and link run of the application we need to modify the **demo.c** file a bit. The **ESp** (**Embedded System profiler**) for example shows much more interesting information when adding the line "**tm_wkafter (10);**" at the end of the function "**process_data ()**".

Now add the line “ **tm_wkafter (10); /* Tutorial addition */**”.

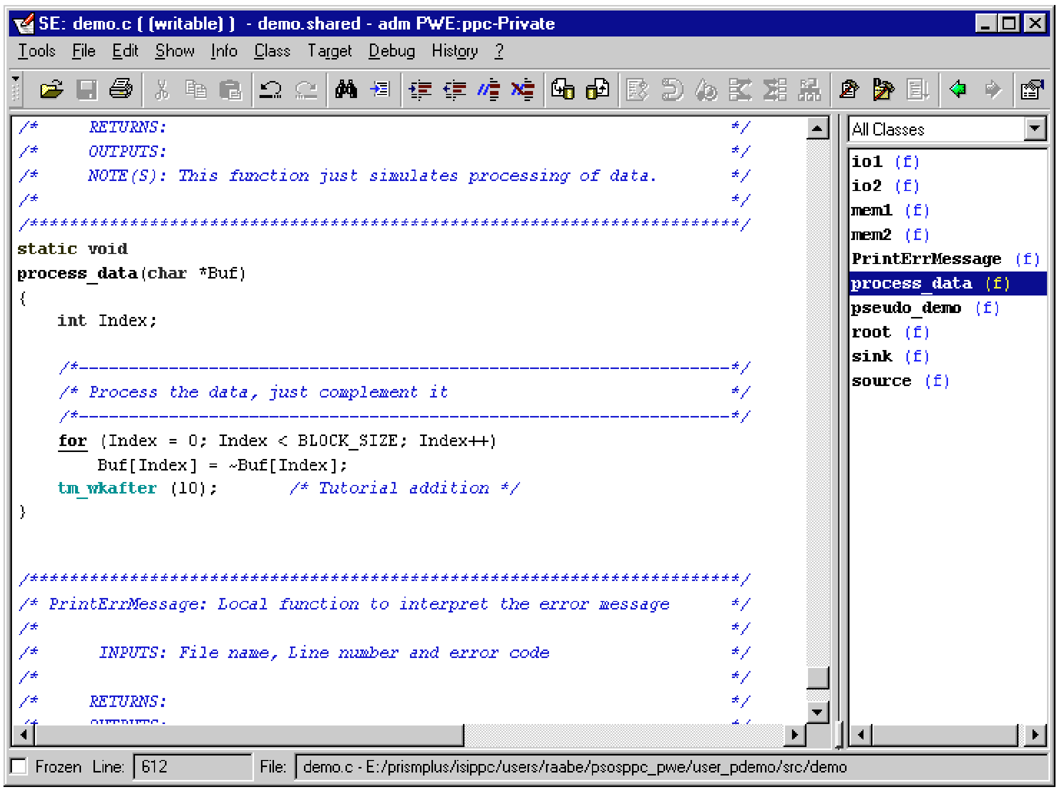

SE: demo.c ((modified)) - demo.shared - adm PWE:ppc-Private
Tools File Edit Show Info Class Target Debug History ?
All Classes
Frozen Line: 612
File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/user_pdemo/src/demo

```
/*      RETURNS:                                                        */
/*      OUTPUTS:                                                        */
/*      NOTE(S): This function just simulates processing of data.      */
/*                                                                      */
/************************************************************************/
static void
process_data(char *Buf)
{
    int Index;

    /*------------------------------------------------------------------*/
    /* Process the data, just complement it                             */
    /*------------------------------------------------------------------*/
    for (Index = 0; Index < BLOCK_SIZE; Index++)
        Buf[Index] = ~Buf[Index];
    tm_wkafter (10);        /* Tutorial addition */
}


/************************************************************************/
/* PrintErrMessage: Local function to interpret the error message       */
/*                                                                      */
/*       INPUTS: File name, Line number and error code                  */
/*                                                                      */
/*      RETURNS:                                                        */
```

```
io1 (f)
io2 (f)
mem1 (f)
mem2 (f)
PrintErrMessage (f)
process_data (f)
pseudo_demo (f)
root (f)
sink (f)
source (f)
```

You might see, that the file is stated in the title as “**(modified)**” so save the file to disk by pressing the “**floppy disk**” icon (or on Windows the shortcut **Ctrl-s**).

Now the title states “**(writable)**” and the “**floppy disk**” icon is greyed out. Also the syntax colouring process completed. The comment is coloured in blue and the tm_wkafter () pSOSystem call is coloured in green.



```
/*     RETURNS:                                                        */
/*     OUTPUTS:                                                        */
/*     NOTE(S): This function just simulates processing of data.      */
/*                                                                     */
/***********************************************************************/
static void
process_data(char *Buf)
{
    int Index;

    /*-----------------------------------------------------------------*/
    /* Process the data, just complement it                            */
    /*-----------------------------------------------------------------*/
    for (Index = 0; Index < BLOCK_SIZE; Index++)
        Buf[Index] = ~Buf[Index];
    tm_wkafter (10);       /* Tutorial addition */
}


/***********************************************************************/
/* PrintErrMessage: Local function to interpret the error message     */
/*                                                                     */
/*      INPUTS: File name, Line number and error code                 */
/*                                                                     */
/*     RETURNS:                                                        */
```

Frozen Line: 612 File: demo.c - E:/prismplus/isippc/users/raabe/psosppc_pwe/user_pdemo/src/demo

All **pSOSystem** API calls are coloured in different flavours of green, depending on the **pSOSystem** component they belong to (like **pSOS+**, **pROBE+**, **pREPC+**, **pNA+**, …)

Compile and Link run

To generate a downloadable **pSOSystem** binary, start the compile and link run in the **PE** by selecting the “**pss_main.shared**” project and using the menu “**Target->Make->ram.elf**”. A new tool opens up, the “**Local SH**” or **SH**. Here are all make-utility based activities executed. The make run is invoked by “**psosmake ram.elf**” which compiles all necessary files and links them to the binary “**ram.elf**”. A link run results in a file called “**ram.map**” which containing the linker’s symbol table.

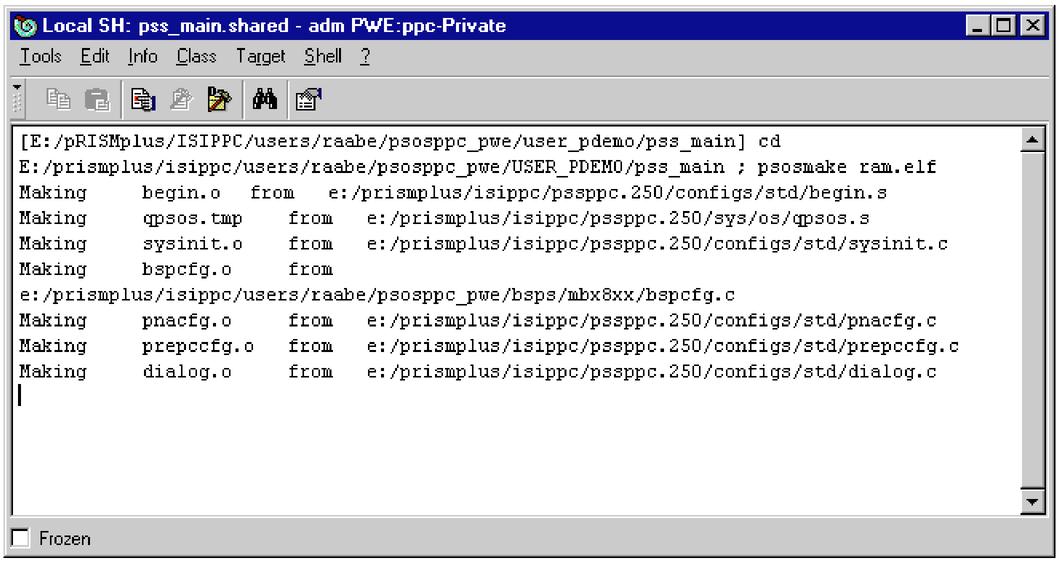
The executables for other targets than PowerPC are:

68K and MIPS: ram.elf

ARM: ram.axf

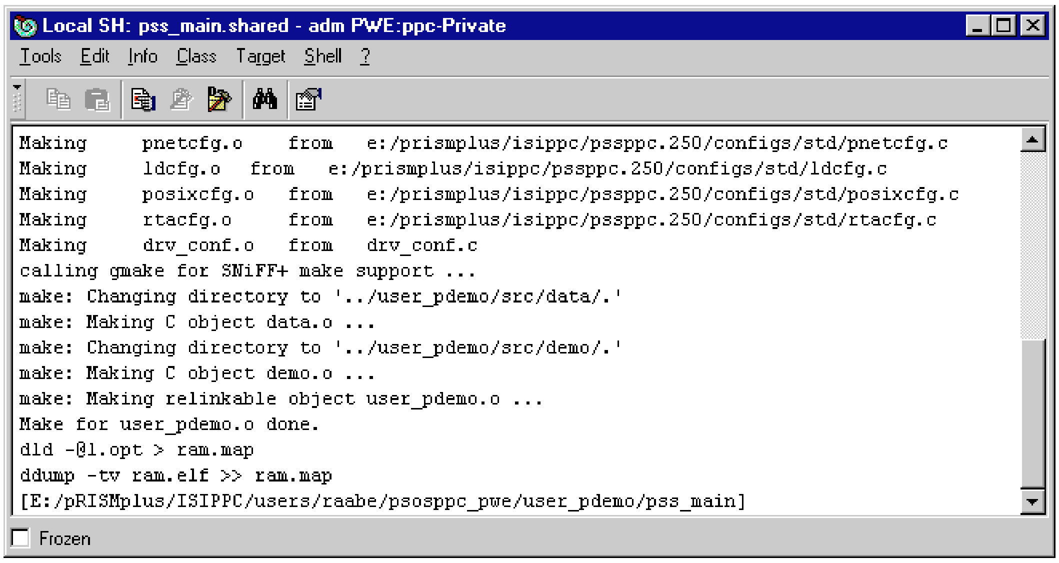
X86: ram.abs

The Shell opens with compiling the **pSOSystem** related files:


Local SH: pss_main.shared - adm PWE:ppc-Private
Tools Edit Info Class Target Shell ?
Frozen

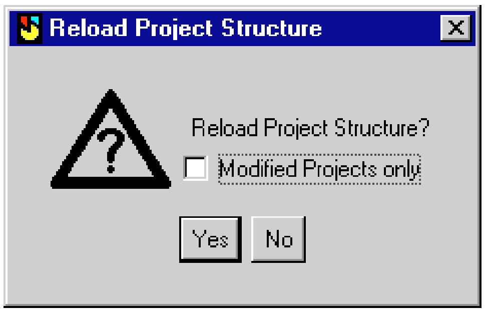
```
[E:/pRISMplus/ISIPPC/users/raabe/psosppc_pwe/user_pdemo/pss_main] cd
E:/prismplus/isippc/users/raabe/psosppc_pwe/USER_PDEMO/pss_main ; psosmake ram.elf
Making     begin.o   from   e:/prismplus/isippc/pssppc.250/configs/std/begin.s
Making     qpsos.tmp    from   e:/prismplus/isippc/pssppc.250/sys/os/qpsos.s
Making     sysinit.o    from   e:/prismplus/isippc/pssppc.250/configs/std/sysinit.c
Making     bspcfg.o     from
e:/prismplus/isippc/users/raabe/psosppc_pwe/bsps/mbx8xx/bspcfg.c
Making     pnacfg.o     from   e:/prismplus/isippc/pssppc.250/configs/std/pnacfg.c
Making     prepccfg.o   from   e:/prismplus/isippc/pssppc.250/configs/std/prepccfg.c
Making     dialog.o     from   e:/prismplus/isippc/pssppc.250/configs/std/dialog.c
```

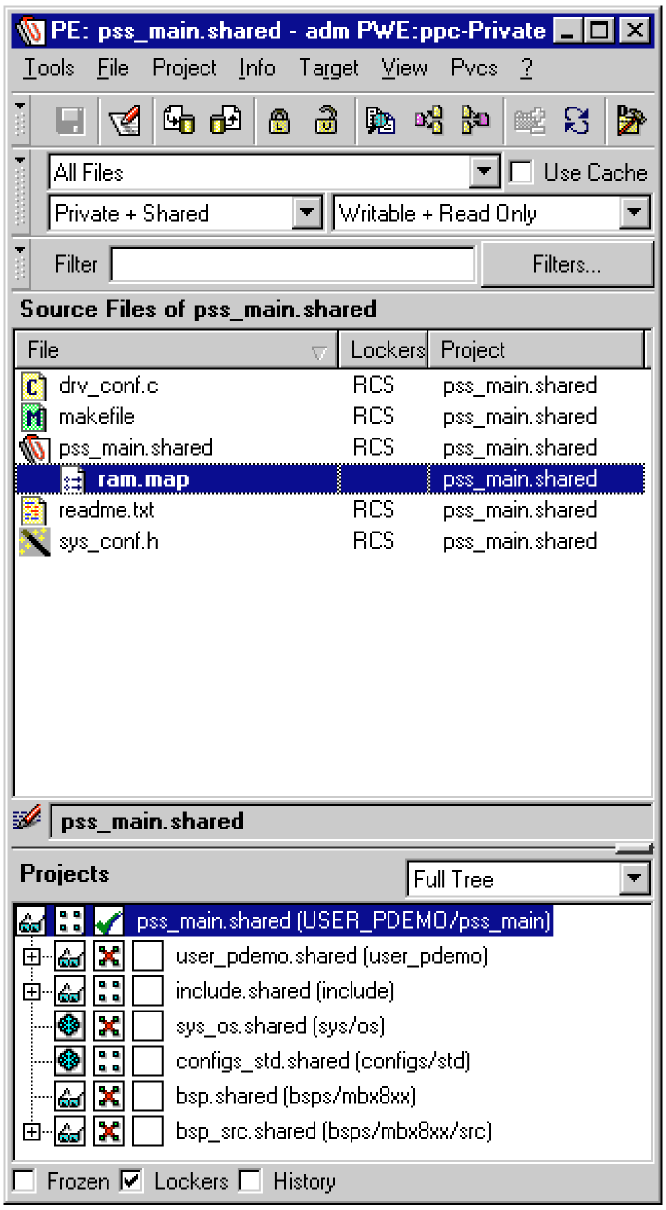
and ends with compiling the existing codebase:


Local SH: pss_main.shared - adm PWE:ppc-Private
Tools Edit Info Class Target Shell ?
Frozen

```
Making     pnetcfg.o    from   e:/prismplus/isippc/pssppc.250/configs/std/pnetcfg.c
Making     ldcfg.o   from   e:/prismplus/isippc/pssppc.250/configs/std/ldcfg.c
Making     posixcfg.o   from   e:/prismplus/isippc/pssppc.250/configs/std/posixcfg.c
Making     rtacfg.o     from   e:/prismplus/isippc/pssppc.250/configs/std/rtacfg.c
Making     drv_conf.o   from   drv_conf.c
calling gmake for SNiFF+ make support ...
make: Changing directory to '../user_pdemo/src/data/.'
make: Making C object data.o ...
make: Changing directory to '../user_pdemo/src/demo/.'
make: Making C object demo.o ...
make: Making relinkable object user_pdemo.o ...
Make for user_pdemo.o done.
dld -@l.opt > ram.map
ddump -tv ram.elf >> ram.map
[E:/pRISMplus/ISIPPC/users/raabe/psosppc_pwe/user_pdemo/pss_main]
```

To review this file, you should reload the project pdemo in the **PE** by check marking only "**pss_main.shared**" in the project sub window and using the menu "**Project->Reload Project-> In Current Working Environment**". A dialog asks you whether you want to reload all projects, you answer (without check marking) a "**YES**".




PE: pss_main.shared - adm PWE:ppc-Private

Source Files of pss_main.shared

File	Lockers	Project
drv_conf.c	RCS	pss_main.shared
makefile	RCS	pss_main.shared
pss_main.shared	RCS	pss_main.shared
ram.map		pss_main.shared
readme.txt	RCS	pss_main.shared
sys_conf.h	RCS	pss_main.shared

pss_main.shared

Projects — Full Tree

- pss_main.shared (USER_PDEMO/pss_main)
  - user_pdemo.shared (user_pdemo)
  - include.shared (include)
  - sys_os.shared (sys/os)
  - configs_std.shared (configs/std)
  - bsp.shared (bsps/mbx8xx)
  - bsp_src.shared (bsps/mbx8xx/src)

☐ Frozen ☑ Lockers ☐ History

Now the **PE** contains new files (which are placed indented in the file window). There is “**ram.map**”, containing the symbol table.

If you need more than one of the same tool open (e.g. two **SE**s), there is a “**Frozen**” checkmark at the lower border. When this checkmark is selected the tool opens a new window when a new tool invocation takes place.

5 SNiFF+ in pRISM+ for the Team

Starting with a standard pSOSystem application (pdemo):

For Teams there is no support for a standard **pSOSystem** application from scratch. To achieve the Team Support you need to copy the standard **pSOSystem** application into the SSWE (**$PSS_USER_SSWE**) and follow the description of "**Starting with an existing codebase**". For further Information refer to the pRISM+ User's Guide chapter **6.4.4** and **6.7.2**.

Starting with an existing codebase (user_pdemo):

Preparations

You should already have the project "user_pdemo" created as described in Chapter 4 – Starting with an existing codebase.

How to add a new user to the team

Login as a new User. Open the Korn Shell using the Start-Button **Start->pRISM+ PPC->Utilities->Korn Shell PPC** and enter the commands:

1. Create the directory **user_pdemo** in the new user's PWE
   (mkdir $PSS_USER_PWE/user_pdemo).
2. Copy the pRISMSpace file **user_pdemo.psp**
   (cp <old user's PWE>/user_pdemo/user_pdemo.psp $PSS_USER_PWE/user_pdemo).
3. Create the directory holding the **pss_main.shared** project description
   (mkdir –p $PSS_USER_PWE/user_pdemo/pss_main/sniffprj).
4. Copy the SNiFF+ project description **pss_main.shared**
   (cp <old user's PWE>/user_pdemo/pss_main/sniffprj /pss_main.shared $PSS_USER_PWE/user_pdemo/pss_main/sniffprj).
5. Set the SNiFF+ project description **pss_main.shared** to read-only
   (chmod a-w $PSS_USER_PWE/user_pdemo/pss_main/sniffprj/ pss_main.shared).
6. Start pRISM+ as usual.
7. Open the just copied user_pdemo.psp file by using the menu
   File->Open. Browse to $PSS_USER_PWE/user_pdemo and open **user_pdemo.psp**.

To synchronize the work of the team members you have the choice of

- synchronization every time when a user checks in a new version
- synchronization on a periodical basis organized by the team administrator

For further information please refer to the SNiFF+ User's Guide.

To synchronize on checking in, set the Project attributed to
Update Shared Project Immediately.

To synchronize on a periodical basis, use the script
$SNIFF_DIR/ws_support/Start_updateWS.bat.

6 Document History

Version	Date	Comment	Author
1.0	March, 14th 2000	Derived from “White Paper.doc” Released for beta3	Martin Raabe
1.1	April, 3rd 2000	Added Note for standard applications and Team Development	Martin Raabe
1.2	April 14th, 2000	Modified formatting to US-Letter Format and ran the spell checker; Adobe Acrobat PDF generated.	Martin Oberhuber