

Program Style and Methodology

Project: General Use

Date: February 5th 2008

Revision: 2.1.0

Company: Quantum Blue Technology LLC.

Copyright Notice

Copyright ©2005, 2006, 2007, 2008 Quantum Blue Technology LLC. – All rights reserved worldwide. This document is proprietary to **Quantum Blue Technology LLC.** And may contain information that is to be maintained as Trade Secret. It is intended for use only by **Quantum Blue Technology LLC.** Employees and its' contractors, customer employees, and authorized personnel. It may not be copied, translated, or transcribed in whole or in part without the express permission of the copyright holder **Quantum Blue Technology LLC.**

Quantum Blue Technology LLC
1424 Welsh Way, Ramona
California 92065
U.S.A.

Phone: USA (858) 837-2160
Email: info@QuantumBlueTechnology.com
Web: www.QuantumBlueTechnology.com

Please Note:

Due to ongoing design and development, **Quantum Blue Technology LLC.** May, at any time, and without notification, amend and update either this document.

Change History

Date	Version	Author	Reason for Change
1/5/06	1.0.0	Steve McClure	Initial Draft.
1/8/06	1.1.0	Steve McClure	Added more detail.
2/1/08	2.0.0	Steve McClure	First Release.
2/5/08	2.1.0	Steve McClure	Minor updates.

Table of Contents

1	SUMMARY.....	1
1.1	SCOPE.....	1
1.2	ABBREVIATIONS.....	1
2	SOFTWARE DEVELOPMENT METHODOLOGY	2
2.1	HEADER FILES.....	2
2.1.1	<i>Module Title Block</i>	3
2.1.2	<i>System Constants (Definitions)</i>	3
2.1.2.1	Integer Definitions.....	3
2.1.2.2	Compiler Defaults	3
2.1.2.3	Example.....	4
2.1.3	<i>System Types (Typedefs)</i>	6
2.1.3.1	Example.....	6
2.1.4	<i>System Global Variables (Publics)</i>	8
2.1.4.1	Example.....	8
2.1.5	<i>System Global Variables (Externals)</i>	10
2.1.5.1	Example.....	10
2.1.6	<i>Module Prototypes</i>	12
2.1.6.1	Example.....	12
2.2	SOURCE FILES	13
2.2.1	<i>Module Title Block</i>	13
2.2.2	<i>Function Title Block</i>	13
2.2.3	<i>Function Parameters</i>	14
2.2.3.1	Passed Parameters	14
2.2.3.2	Return Parameter.....	15
2.2.4	<i>Function Code Layout</i>	15
2.2.4.1	For Construct.....	15
2.2.4.2	While Construct.....	16
2.2.4.3	Do While Construct.....	16
2.2.4.4	Switch Construct	16
2.2.4.5	If Construct.....	17
2.2.5	<i>Error Detection</i>	17
2.2.6	<i>Pseudocode</i>	18
2.2.7	<i>Comments</i>	18
2.2.8	<i>Example</i>	19
3	APPENDIX	52
3.1	CODE EDITOR.....	52
3.1.1	<i>Use of Tabs versus Spaces</i>	52

Program Style and Methodology

1 Summary

This document describes the Quantum Blue Technology LLC. programming methodology used in developing company software.

1.1 Scope

This document identifies the layout of the program unit structure. This includes the module header, function headers, and general code constructs.

1.2 Abbreviations

IDE Integrated Development Environment
PC Pseudo Code (also known as Structured English)
PDL Program Design Language.

Program Style and Methodology

2 Software Development Methodology

The purpose of programming is to develop a set of instructions, which make logical sense to both a computer system used to execute the instructions and to a human being who has to develop and maintain these instructions.

With this regard, the use of logical clear concise unambiguous names for modules, structures, variables and functions can not be overemphasized.

A program is built from the following types of files:

1. Header Files
2. Source Files

2.1 Header Files

In this methodology, the following types of header files are used:

1. The system constants header file
2. The system types header file
3. The system global variable header file
4. The module prototype header file

The advantage in this methodology is that it is a clean approach with specific definitions only existing in specific locations - a programmer immediately knows which file to edit for a given definition.

The disadvantage in this methodology is that a change to one of these files may result in the entire system having to be rebuilt. On current computer system this may result in having to wait an additional few seconds longer for the entire system build to complete - this additional time delay is hardly an issue in regard to the conceptual gain in code layout clarity.

Program Style and Methodology

2.1.1 Module Title Block

Each header file contains a descriptive title block, which identifies:

1. The module name
2. The customer's copyright information
3. The project name
4. The original author
5. The module creation date
6. A brief description of the module
7. The module version history information

2.1.2 System Constants (Definitions)

A project system constants header file is identified by the name "Definitions.h". This file is included in every source module.

2.1.2.1 Integer Definitions

A common programming problem exists with the manner in which a specific compiler defines the integer variable type. For instance, one compiler may identify an `int` as a 32-bit quantity whereas another compiler identifies it as a 16-bit quantity.

This confusion can easily develop into programming errors.

To bypass this problem the `uint` and `sint` literal definitions have been created as shown below:

```
/* Variable Integer definitions */
#define uint8      unsigned char
#define uint16     unsigned int
#define uint32     unsigned long

#define sint8      signed char
#define sint16     signed int
#define sint32     signed long
```

These literal definitions are located in the system definition header file.

2.1.2.2 Compiler Defaults

Compilers usually have specific default settings - for example, when defining a `char` variable, one compiler may default this to an `unsigned char`. However, another compiler may default to a `signed char`. It is also possible to further complicate the situation by overriding the compiler defaults at compile time.

To remove this confusion the integer definitions are used as discussed above.

Program Style and Methodology

2.1.2.3 Example

The following extract is an example of a system constants definition header file:

```
//-----  
// Project: POS Business Application  
// Module: Definitions.h  
// Author: Stephen W. McClure  
// Date: Oct 2005  
//-----  
// Description:  
//  
// This header file contains the system definitions.  
//-----  
// Version History:  
//  
// Version: 1.00      Date: 10/10/2005      Author: Steve McClure  
// Reason: Initial release.  
//-----  
// Version: 1.01      Date: 02/12/2006      Author: Steve McClure  
// Reason: Added matrix changes.  
//-----  
// This program is the exclusive property of Quantum Blue Technology LLC.  
// Copyright(c) 2005, 2006, 2007, 2008.  
//  
// Reproduction, disclosure, or use, in whole or in part, are not to be  
// undertaken except with prior written authorization from the owner  
// Quantum Blue Technology LLC.  
//  
// Contact Information: Quantum Blue Technology LLC  
//                        1424 Welsh Way  
//                        Ramona  
//                        CA 92065  
//  
//                        Phone: (858) 837-2160  
//                        Email: info@quantumbluetechology.com  
//-----  
  
#ifndef SysdefsH  
#define SysdefsH  
  
/* Variable Integer definitions */  
#define uint8      unsigned char  
#define uint16     unsigned int  
#define uint32     unsigned long  
  
#define sint8      signed char  
#define sint16     signed int  
#define sint32     signed long  
  
/* General Definitions */  
#define TRUE      1  
#define FALSE     0  
  
#define ON        1  
#define OFF       0  
  
#define SUCCESS   1  
#define FAILURE   0
```

Program Style and Methodology

```
/* DSP Register Addresses */
#define TIMER_CTL0_REG_ADDRESS    0x01940000
#define TIMER_PRD0_REG_ADDRESS    0x01940004
#define TIMER_CNT0_REG_ADDRESS    0x01940008

/* Test LED State */
#define LED_OFF                    0
#define LED_HEARTBEAT_ON          1
#define LED_ERROR_ON              2
#define LED_ERROR_OFF            3

#endif
```

Discussion

1. The use of the definitions `#ifndef SysdefsH`, `#define SysdefsH` and at the end of the file `#endif` ensure that the include file will not generate a compiler error if it is included multiple times.
2. When defining various common device states, use the device as the first part of the state name - this keeps the various device states distinct from each other.
3. Column alignment aids in readability and reduces the likelihood of mistakes.

Program Style and Methodology

2.1.3 System Types (Typedefs)

A project system types header file is identified by the name “Typedefs.h”. This file is included in every source module.

2.1.3.1 Example

The following extract is an example of a system types header file:

```
//-----  
// Project: POS Business Application  
// Module: Typedefs.h  
// Author: Stephen W. McClure  
// Date: Oct 2005  
//-----  
// Description:  
//  
// This header file contains the various system typedef definitions.  
//  
// Note: System Variable sizes are as follows:  
//  
//          char          1 byte  
//  
//          short int     2 bytes  
//          int           4 bytes  
//          long int      4 bytes  
//  
//          float         4 bytes  
//  
//          DWORD         4 bytes  
//  
//          double        8 bytes  
//          long double   10 bytes  
//  
//-----  
// Version History:  
//  
// Version: 1.00      Date: 10/10/2005      Author: Steve McClure  
// Reason: Initial release.  
//-----  
// Version: 1.01      Date: 02/12/2006      Author: Steve McClure  
// Reason: Added matrix changes.  
//-----  
// This program is the exclusive property of Quantum Blue Technology LLC.  
// Copyright(c) 2005, 2006, 2007, 2008.  
//  
// Reproduction, disclosure, or use, in whole or in part, are not to be  
// undertaken except with prior written authorization from the owner  
// Quantum Blue Technology LLC.  
//  
// Contact Information: Quantum Blue Technology LLC  
//                      1424 Welsh Way  
//                      Ramona  
//                      CA 92065  
//  
//                      Phone: (858) 837-2160  
//                      Email: info@quantumbluetechology.com  
//-----
```

Program Style and Methodology

```
#ifndef SystypesH
#define SystypesH

/* SETUP Data Message */
typedef struct SETUP_DATA_
{
    uint8      request_type;
    uint8      request;
    uint16     value;
    uint16     index;
    uint16     length;
    SETUP_DATA_ * next_record;
}
    SETUP_DATA;

#endif
```

Please note the following:

1. The use of the definitions `#ifndef SystypesH`, `#define SystypesH` and at the end of the file `#endif` ensure that the include file will not generate a compiler error if it is included multiple times.
2. The `typedef` construct is used to permit a variable to be defined simply by using the `typedef` name, for example:

```
    SETUP_DATA    setup_data;
```
3. In order to help reduce the number of different variable names, the `typedef` is written using capital letters and the variable of that type uses the same name but in lowercase letters.
4. When the `typedef` is defined, the `typedef` name with an underscore, for example `SETUP_DATA_` follows the `struct` reserved word. This permits the compiler to recognize pointer variables, which point to the same structure as that is being defined.
5. Column alignment aids in readability and reduces the likelihood of mistakes.

Program Style and Methodology

2.1.4 System Global Variables (Publics)

A project system global variables public header file is identified by the name "Publics.h".

2.1.4.1 Example

The following extract is an example of a system global variables public header file:

```
//-----  
// Project: POS Business Application  
// Module: Publics.h  
// Author: Stephen W. McClure  
// Date: Oct 2005  
//-----  
// Description:  
//  
// This header file contains the global public variable definitions.  
//-----  
// Version History:  
//  
// Version: 1.00      Date: 10/10/2005      Author: Steve McClure  
// Reason: Initial release.  
//-----  
// Version: 1.01      Date: 02/14/2006      Author: Steve McClure  
// Reason: Added matrix changes.  
//-----  
// This program is the exclusive property of Quantum Blue Technology LLC.  
// Copyright(c) 2005, 2006, 2007, 2008.  
//  
// Reproduction, disclosure, or use, in whole or in part, are not to be  
// undertaken except with prior written authorization from the owner  
// Quantum Blue Technology LLC.  
//  
// Contact Information: Quantum Blue Technology LLC  
//                        1424 Welsh Way  
//                        Ramona  
//                        CA 92065  
//  
//                        Phone: (858) 837-2160  
//                        Email: info@quantumbluetechology.com  
//-----  
  
#ifndef PublicsH  
#define PublicsH  
  
#include "Typedefs.h"  
  
#include "POS_Inventory.h"  
#include "POS_Timers.h"  
#include "POS_Network.h"  
#include "POS_Printer_Index.h"  
#include "POS_Communications.h"  
  
#include <dos.h>  
#include <SyncObjs.hpp>  
  
/* System Variables */  
HANDLE single_task_execution_mutex = NULL;
```

Program Style and Methodology

```
int  network_registration_error      = FALSE;
int  network_licenses_all_used_error = FALSE;

#endif
```

Discussion

1. The use of the definitions `#ifndef PublicsH`, `#define PublicsH` and at the end of the file `#endif` ensure that the include file will not generate a compiler error if it is included multiple times.
2. Note the use of descriptive typedef and variable names. Unambiguous names greatly add to the readability of program code and the myth of self-documented code could actually become a reality. The time saving in program maintenance is enormous.
3. Column alignment aids in readability and reduces the likelihood of mistakes.

Program Style and Methodology

2.1.5 System Global Variables (Externals)

A project system global variables external header file is identified by the name "Externals.h".

2.1.5.1 Example

The following extract is an example of a system global variables external header file:

```
//-----  
// Project: POS Business Application  
// Module: Externals.h  
// Author: Stephen W. McClure  
// Date: Oct 2005  
//-----  
// Description:  
//  
// This header file contains external variable definitions.  
//-----  
// Version History:  
//  
// Version: 1.00      Date: 10/10/2005      Author: Steve McClure  
// Reason: Initial release.  
//-----  
// Version: 1.01      Date: 02/14/2006      Author: Steve McClure  
// Reason: Added matrix changes.  
//-----  
// This program is the exclusive property of Quantum Blue Technology LLC.  
// Copyright(c) 2005, 2006, 2007, 2008.  
//  
// Reproduction, disclosure, or use, in whole or in part, are not to be  
// undertaken except with prior written authorization from the owner  
// Quantum Blue Technology LLC.  
//  
// Contact Information: Quantum Blue Technology LLC  
//                        1424 Welsh Way  
//                        Ramona  
//                        CA 92065  
//  
//                        Phone: (858) 837-2160  
//                        Email: info@quantumbluetechology.com  
//-----  
  
#ifndef ExternalsH  
#define ExternalsH  
  
#include "POS_Timers.h"  
#include "POS_Network.h"  
#include "POS_Printer_Index.h"  
#include "POS_Communications.h"  
  
#include <dos.h>  
#include <SyncObjs.hpp>  
  
/* System Variables */  
extern HANDLE single_task_execution_mutex;
```

Program Style and Methodology

```
extern int  network_registration_error;  
extern int  network_licenses_all_used_error;  
  
#endif
```

Discussion

1. The use of the definitions `#ifndef ExternalsH, #define ExternalsH` and at the end of the file `#endif` ensure that the include file will not generate a compiler error if it is included multiple times.
2. Note the use of descriptive typedef and variable names. Unambiguous names greatly add to the readability of program code and the myth of self-documented code could actually become a reality. The time saving in program maintenance is enormous.
3. Column alignment aids in readability and reduces the likelihood of mistakes.

Program Style and Methodology

2.1.6 Module Prototypes

Each source code module (ie. “.c”, or “.cpp” file) which contains function definitions will have an associated prototype header file. The name of the file will be identical to the source code module except it will utilize the “.h” suffix.

For example:

digital.c - The digital source code functions definitions.
digital.h - The digital source code function prototype definitions.

2.1.6.1 Example

The following extract is an example of a module prototype header file:

```
//-----  
// Project: POS Business Application  
// Module: POS_System.h  
// Author: Stephen W. McClure  
// Date: Oct 2005  
//-----  
// Description:  
//  
// This header file contains the POS System function prototypes.  
//-----  
// Version History:  
//  
// Version: 1.00      Date: 10/10/2005      Author: Steve McClure  
// Reason: Initial release.  
//-----  
// Version: 1.01      Date: 02/14/2006      Author: Steve McClure  
// Reason: Added matrix changes.  
//-----  
// This program is the exclusive property of Quantum Blue Technology LLC.  
// Copyright(c) 2005, 2006, 2007, 2008.  
//  
// Reproduction, disclosure, or use, in whole or in part, are not to be  
// undertaken except with prior written authorization from the owner  
// Quantum Blue Technology LLC.  
//  
// Contact Information: Quantum Blue Technology LLC  
//                      1424 Welsh Way  
//                      Ramona  
//                      CA 92065  
//  
//                      Phone: (858) 837-2160  
//                      Email: info@quantumbluetechology.com  
//-----  
  
#ifndef POS_SystemH  
#define POS_SystemH
```

Program Style and Methodology

```
int    POS_MessageBox (void          * Null_ptr,
                      const char * Text,
                      const char * Caption,
                      int          Flags);

int    Status_display_clear (void);
int    Status_display (char * status_string, int timeout);
int    Status_display_beep (char * status_string, int timeout);

#endif
```

Please note the following:

1. The use of the definitions `#ifndef POS_SystemH, #define POS_SystemH` and at the end of the file `#endif` ensure that the include file will not generate a compiler error if it is included multiple times.

2.2 Source Files

The source module is comprised of the following sections:

1. Module Title Block
2. Function Title Block
3. Function Parameters
4. Function Code Layout
5. Comments

2.2.1 Module Title Block

Each source module contains a descriptive title block, which identifies:

1. The module name
2. The customer's copyright information
3. The project name
4. The original author
5. The module creation date
6. A brief description of the module
7. The module version history information

2.2.2 Function Title Block

Each function is preceded by a descriptive title block, which identifies:

1. The function name
2. A brief description
3. The passed parameters
4. The returned parameter

Program Style and Methodology

The following style should be used:

```
//-----  
// Status Display Clear  
//  
// This function removes any displayed message from the status display.  
//  
// Returns: SUCCESS  
//-----  
  
int Status_display_clear (void)  
{  
  
/* Erase the Status display message */  
Status_display ("", INDEFINITELY);  
  
/* Note that the display is inactive */  
status_display = INACTIVE;  
  
return (SUCCESS);  
}
```

2.2.3 Function Parameters

Each function may utilize the following types of parameter:

1. Passed Parameters
2. Returned Parameters

2.2.3.1 Passed Parameters

The following style is used to define the function name and associated parameters:

```
int status_display (char * status_string, int timeout)  
  
{  
    uint8    key;  
    uint8    digit;  
  
    ... code ...  
    ... code ...  
}
```

Program Style and Methodology

If there are too many parameters to list on one line, the following style is used:

```
int  status_display (char * status_string,
                    char * error_message,
                    int   timeout)

{
    uint8  key;
    uint8  digit;

    ...   code ...
    ...   code ...
}
```

2.2.3.2 Return Parameter

The return parameter is to be used for returning the function status.

The function status is either:

1. SUCCESS
2. FAILURE

These literal constants are defined in the “systypes.h” file as follows:

```
#define SUCCESS 1
#define FAILURE 0
```

2.2.4 Function Code Layout

The function code layout style for the following constructs will now be discussed:

1. The For Construct
2. The While Construct
3. The Do... While Construct
4. The IF Construct
5. The Switch Construct

2.2.4.1 For Construct

The code layout style of the FOR construct is as follows:

```
for (index = 0; index < MAX_ITERATIONS; index++)
{
    key = rs232_getch();
    ...
    ...
}
```

Program Style and Methodology

2.2.4.2 While Construct

The code layout style of the WHILE construct is as follows:

```
index = 0;
while (index < MAX_ITERATIONS)
{
    key = rs232_getch();
    ...
    ...
    index++;
}
```

2.2.4.3 Do While Construct

The code layout style of the Do .. WHILE construct is as follows:

```
index = 0;
do
{
    key = rs232_getch();
    ...
    ...
    index++;
}
while (index < MAX_ITERATIONS);
```

Note: This construct is rarely used.

2.2.4.4 Switch Construct

The code layout style of the switch construct is as follows:

```
switch (digit)
{
    case TEST_INVALID:
        return (FAILURE);
    case TEST_ONCE:
        perform_flash_memory_test_once();
        break;
    case TEST_REPEATEDLY:
        perform_flash_memory_test_repeatedly();
        break;
    default:
        return (FAILURE);
}
```

Program Style and Methodology

2.2.4.5 If Construct

The code layout style of the various IF constructs are as follows:

```
if (index < MAX_ITERATIONS)
    function_1();

if (index < MAX_ITERATIONS)
    function_1();
else
    function_2();

if (index < MAX_ITERATIONS)
    function_1();
else if (index == MAX_ITERATIONS)
    function_2();
else
    function_3();

if (index < MAX_ITERATIONS)
{
    function_1a();
    function_1b();
}
else if (index == MAX_ITERATIONS)
{
    if (test == ACTIVE)
    {
        test_function_1();
        test_function_2();
    }
}
else
    function_3();
```

2.2.5 Error Detection

There are many ways to handle error situations, which are detected within a function.

The manner, which will be utilized, is to exit from the function as soon as the error has been detected. This permits a clear concise logic flow exiting the function.

```
...
/* Test if error occurred */
if (error == DETECTED)
    return (FAILURE);
...
```

The method whereby the error condition is delegated to the distant else part of an IF condition is discouraged since such code becomes difficult to read when the function is of a significant size.

Program Style and Methodology

2.2.6 Pseudocode

Pseudocode (also known as structured English) should be used to provide a description of the function of the subsequent lines of code, as shown below:

```
/* Do Forever */
while (1)
{
    /* Wait for a key press */
    key = rs232_getch();

    /* If [ESC] key pressed - exit */
    if (key == ESC_KEY)
        return (FAILURE);

    /* If [ENTER] key pressed - return number */
    if (key == ENTER_KEY)
    {
        *number = result;
        return (SUCCESS);
    }
}
```

2.2.7 Comments

Comments should only be used where it is necessary to explain in detail the actions of a code fragment. The comments should be well to the right of the code such that the code and the comments are clearly delineated. Both `/* comment */` and `// Comment` styles are permitted:

```
TCR.BIT.IEDG = SET;          /* Input Capture interrupts triggered on positive edge */
disable_fan_fault_interrupt(); // Only enabled when fan switched on, otherwise switch
                               // generates interrupt in inactive state !!!
```

Program Style and Methodology

2.2.8 Example

The following example illustrates the overall code layout style:

```
//-----  
// Project: POS Business Application  
// Module: POS_System.cpp  
// Author: Stephen W. McClure  
// Date: Oct 2007  
//-----  
// Description:  
//  
// This module contains the POS system functions.  
//-----  
// This program is the exclusive property of Quantum Blue Technology LLC.  
// Copyright(c) 2005, 2006, 2007, 2008.  
//  
// Reproduction, disclosure, or use, in whole or in part, are not to be  
// undertaken except with prior written authorization from the owner  
// Quantum Blue Technology LLC.  
//  
// Contact Information: Quantum Blue Technology LLC  
//                        1424 Welsh Way  
//                        Ramona  
//                        CA 92065  
//  
//                        Phone: (858) 837-2160  
//                        Email: info@quantumbluetechology.com  
//-----  
  
#include <windows.h>                // This is required  
#undef GetEnvironmentVariable      // for proper operation of the GetEnvironmentVariable  
#include <vcl.h>  
#pragma hdrstop  
  
#include "Definitions.h"  
#include "Typedefs.h"  
#include "Externals.h"  
  
#include "POS_System.h"  
#include "POS_Main.h"  
#include "POS_Log.h"  
#include "POS_Files.h"  
#include "POS_Accounts_Payable.h"  
#include "POS_Accounts_Receivable.h"  
#include "POS_Bank_Reconciliation.h"  
#include "POS_General_Ledger.h"  
#include "POS_Inventory.h"  
#include "POS_Payroll.h"  
#include "POS_Purchase_Order.h"  
#include "POS_Register.h"  
#include "POS_Registry.h"  
#include "POS_Import.h"  
#include "POS_Invoice_Number.h"  
#include "POS_Register_Reports.h"  
  
#include <process.h>  
#include <sysutils.hpp>  
#include <dir.h>  
#include <fcntl.h>  
#include <io.h>  
#include <sys\stat.h>  
#include <stdio.h>  
#include <mem.h>  
#include <errno.h>  
#include <math.hpp>  
#include <Inifiles.hpp>
```

Program Style and Methodology

```
//-----  
// POS MessageBox  
//  
// This function displays an error message box.  
//  
// Returns: status of MessageBox key press.  
//-----  
  
int POS_MessageBox (void * Null_ptr, const char * Text, const char * Caption, int Flags)  
{  
    int status;  
  
    Application->NormalizeTopMosts();  
    status = MessageBox(Null_ptr, Text, Caption, Flags | MB_TASKMODAL);  
    Application->RestoreTopMosts();  
  
    return (status);  
}  
  
//-----  
// Status Display Clear  
//  
// This function removes any displayed message from the status display.  
//  
// Returns: SUCCESS  
//-----  
  
int Status_display_clear (void)  
{  
  
    /* Erase the Status display message */  
    Status_display ("", INDEFINITELY);  
  
    /* Note that the display is inactive */  
    status_display = INACTIVE;  
  
    return (SUCCESS);  
}  
  
//-----  
// Status Display Beep  
//  
// This function beeps and places the passed data into the status bar for the  
// specified timeout period.  
//  
// Returns: SUCCESS  
//-----  
  
int Status_display_beep (char * status_string, int timeout)  
{  
  
    /* The status display is now active */  
    status_display = ACTIVE;  
  
    /* Sound the warning */  
    MessageBeep(MB_ICONEXCLAMATION);  
  
    /* Display the status message */  
    Status_display (status_string, timeout);  
  
    /* SUCCESS */  
    return (SUCCESS);  
}
```

Program Style and Methodology

```
//-----  
// Status Display Blank  
//  
// This function displays the new status message only if the currently  
// displayed status message is blank.  
//  
// Returns: SUCCESS  
//-----  
  
int Status_display_blank (char * status_string, int timeout)  
{  
  
/* Is the Status Message currently blank? */  
if (status_display == INACTIVE)  
{  
    /* Status display is now active */  
    status_display = ACTIVE;  
  
    /* Yes - Display the status message */  
    Status_display (status_string, timeout);  
  
    /* SUCCESS */  
    return (SUCCESS);  
}  
  
/* FAILURE */  
return (FAILURE);  
}  
  
  
//-----  
// Status Display Beep Blank  
//  
// This function beeps and places the passed data into the status bar for the  
// specified timeout period only if the currently displayed status message is  
// blank.  
//  
// Returns: SUCCESS  
//-----  
  
int Status_display_beep_blank (char * status_string, int timeout)  
{  
  
/* Is the Status Message currently blank? */  
if (status_display == INACTIVE)  
{  
    /* Status display is now active */  
    status_display = ACTIVE;  
  
    /* Yes - Display the status message */  
    Status_display_beep (status_string, timeout);  
  
    /* SUCCESS */  
    return (SUCCESS);  
}  
  
/* FAILURE */  
return (FAILURE);  
}
```

Program Style and Methodology

```
//-----  
// Clean Up String  
//  
// This function cleans up the parameter string by removing leading and  
// trailing spaces, multiple embedded spaces are replaced by a single space,  
// and all control characters are removed.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int clean_up_string (AnsiString * parameter)  
{  
    int    parameter_length;  
    char   buffer [MAX_STRING_SIZE];  
    char * read_ptr;  
    char * write_ptr;  
  
    AnsiString local_parameter;  
    AnsiString trimmed_parameter;  
  
    /* Set Text to Uppercase */  
    local_parameter = *parameter;  
  
    /* Remove leading and trailing spaces and any control characters */  
    trimmed_parameter = local_parameter.Trim();  
  
    /* Make a copy of the parameter string */  
    strcpy (buffer, trimmed_parameter.c_str());  
  
    /* Initialize pointers */  
    read_ptr = buffer;  
    write_ptr = buffer;  
  
    /* Scan string and remove excess spaces */  
    while (*read_ptr != 0x00)  
    {  
        /* Copy non-blank characters */  
        if (*read_ptr != ' ')  
            *write_ptr++ = *read_ptr++;  
  
        /* Copy single space character */  
        if (*read_ptr == ' ')  
        {  
            /* Copy single space character */  
            *write_ptr++ = *read_ptr++;  
  
            /* Remove any excess spaces */  
            while (*read_ptr == ' ')  
            {  
                /* Step to next character */  
                read_ptr++;  
            }  
        }  
    }  
  
    /* Null terminate the updated string */  
    *write_ptr = 0x00;  
  
    /* Return updates string */  
    *parameter = buffer;  
  
    /* All is well... */  
    return (SUCCESS);  
}
```

Program Style and Methodology

```
//-----  
// Read System Registry File  
//  
// This function reads the system configuration registry for specific  
// system variables.  
//  
// Note: The POS variable is set to UPPER CASE.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int read_system_registry_parameters (void)  
{  
  
/* Note: Only the Computer Name is currently obtained from the System Registry */  
POS_computer_name = AnsiUpperCase(get_registry_string  
  
("SYSTEM\\CurrentControlSet\\Control\\ComputerName\\ComputerName",  
 "ComputerName"));  
  
/* Is the Computer Name defined? */  
if (POS_computer_name == "")  
{  
    /* No - Set the computer name to "Undefined" */  
    POS_computer_name = "UNDEFINED";  
}  
  
/* Success */  
return (SUCCESS);  
}
```

Program Style and Methodology

```
//-----  
// Read Configuration  
//  
// This function reads the specified configuration parameter from the  
// configuration file. If the configuration parameter does not exist then  
// the default is returned.  
//  
// Returns: Configuration Parameter  
//-----  
  
AnsiString Read_configuration (AnsiString section_name,  
                              AnsiString configuration_name,  
                              AnsiString default_setting)  
{  
    int index;  
    int start_of_section;  
    int end_of_section;  
    int number_of_configuration_lines;  
    AnsiString full_section_name;  
    AnsiString configuration_string;  
    AnsiString first_token;  
    AnsiString second_token;  
    AnsiString third_token;  
  
    /* Load local configuration file into Memo Box */  
    POS_Main_Form->System_Configuration_Memo->Lines->Clear();  
    POS_Main_Form->System_Configuration_Memo->Lines->  
    >LoadFromFile(system_configuration_file_name);  
  
    /* Initialization */  
    index = 0;  
    start_of_section = 0;  
    end_of_section = 0;  
    number_of_configuration_lines = POS_Main_Form->System_Configuration_Memo->Lines->Count;  
    full_section_name = "[" + section_name + "];"  
  
    /* Scan for section name */  
    while (index < number_of_configuration_lines)  
    {  
        /* Section Name found? */  
        if (AnsiUpperCase(POS_Main_Form->System_Configuration_Memo->Lines->Strings[index]) ==  
            AnsiUpperCase(full_section_name))  
        {  
            /* Start of section found */  
            start_of_section = index + 1;  
            break;  
        }  
  
        /* Step to next configuration entry */  
        index++;  
    }  
  
    /* Section name not found? */  
    if (index >= number_of_configuration_lines)  
    {  
        /* Return default value */  
        return (AnsiUpperCase(default_setting));  
    }  
  
    /* Step to next line */  
    index++;  
}
```

Program Style and Methodology

```
/* Scan for subsequent section name or end of file */
while (index < number_of_configuration_lines)
{
    /* Next section name detected? */
    if (POS_Main_Form->System_Configuration_Memo->Lines->Strings[index].c_str()[0] ==
    '[')
    {
        /* End of section found */
        break;
    }

    /* Step to next configuration entry */
    index++;
}

/* End of section found */
end_of_section = index;

/* Search through section for required configuration variable */
for (index = start_of_section; index < end_of_section; index++)
{
    /* Get the next configuration string */
    configuration_string = POS_Main_Form->System_Configuration_Memo->Lines-
>Strings[index];
    configuration_string = configuration_string.Trim();

    /* Check that configuration string is not null and is not a comment */
    if ((configuration_string != "") &&
        (configuration_string.c_str()[0] != ';'))
    {
        /* Extract tokens from string */
        if (extract_token (configuration_string, &first_token, 0) == FAILURE)
            return (default_setting);
        if (extract_token (configuration_string, &second_token, 1) == FAILURE)
            return (default_setting);
        if (extract_token (configuration_string, &third_token, 2) == FAILURE)
            return (default_setting);

        /* Check for configuration variable name */
        if (AnsiUpperCase(first_token) == AnsiUpperCase(configuration_name))
        {
            /* Check that second token is the "=" sign */
            if (second_token == "=")
            {
                /* Return configuration setting */
                return (AnsiUpperCase(third_token));
            }

            /* Error detected */
            return (AnsiUpperCase(default_setting));
        }
    }
}

/* Configuration variable name not found - return default value */
return (AnsiUpperCase(default_setting));
}
```

Program Style and Methodology

```
//-----  
// Write Configuration  
//  
// This function writes the specified configuration parameter to the  
// configuration file. If the configuration parameter exists then its value  
// is changed. If the configuration parameter does not exist then a new  
// parameter is written.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int Write_configuration (AnsiString section_name,  
                        AnsiString configuration_name,  
                        AnsiString configuration_value)  
{  
    int index;  
    int start_of_section;  
    int end_of_section;  
    int number_of_configuration_lines;  
    AnsiString full_section_name;  
    AnsiString configuration_string;  
    AnsiString first_token;  
    AnsiString new_configuration_section;  
    AnsiString new_configuration_variable;  
  
    /* Initialization */  
    new_configuration_section = "[" + section_name + "];  
  
    /* Does the configuration value contain a space? */  
    if (StrPos (configuration_value.c_str(), " ") == NULL)  
    {  
        /* No - Just use the configuration value */  
        new_configuration_variable = configuration_name + " = " + configuration_value;  
    }  
    else  
    {  
        /* Yes - Place the configuration value in double quotes */  
        new_configuration_variable = configuration_name + " = \"" + configuration_value +  
        "\"";  
    }  
  
    /*-----  
    /* First determine if the configuration name exists in the section */  
    /*-----  
  
    /* Load local configuration file into Memo Box */  
    POS_Main_Form->System_Configuration_Memo->Lines->Clear();  
    POS_Main_Form->System_Configuration_Memo->Lines->  
    >LoadFromFile(system_configuration_file_name);  
  
    /* Initialization */  
    index = 0;  
    start_of_section = 0;  
    end_of_section = 0;  
    number_of_configuration_lines = POS_Main_Form->System_Configuration_Memo->Lines->Count;  
    full_section_name = "[" + section_name + "];
```

Program Style and Methodology

```
/* Scan for section name */
while (index < number_of_configuration_lines)
{
    /* Section Name found? */
    if (AnsiUpperCase(POS_Main_Form->System_Configuration_Memo->Lines->Strings[index]) ==
        AnsiUpperCase(full_section_name))
    {
        /* Start of section found */
        start_of_section = index + 1;
        break;
    }

    /* Step to next configuration entry */
    index++;
}

/* Section name not found? */
if (index >= number_of_configuration_lines)
{
    /* Create New Section Name */
    POS_Main_Form->System_Configuration_Memo->Lines->Add(" ");
    POS_Main_Form->System_Configuration_Memo->Lines->Add(new_configuration_section);

    /* Write New Configuration Variable */
    POS_Main_Form->System_Configuration_Memo->Lines->Add(new_configuration_variable);

    /* Save Configuration File */
    POS_Main_Form->System_Configuration_Memo->Lines->SaveToFile
(system_configuration_file_name);

    /* Return */
    return (SUCCESS);
}

/* Step to next line */
index++;

/* Scan for subsequent section name or end of file */
while (index < number_of_configuration_lines)
{
    /* Next section name detected? */
    if (POS_Main_Form->System_Configuration_Memo->Lines->Strings[index].c_str()[0] == '[')
    {
        /* End of section found */
        break;
    }

    /* Step to next configuration entry */
    index++;
}

/* End of section found */
end_of_section = index;

/* Search through section for required configuration variable */
for (index = start_of_section; index < end_of_section; index++)
{
    /* Get the next configuration string */
    configuration_string = POS_Main_Form->System_Configuration_Memo->Lines-
>Strings[index];
    configuration_string = configuration_string.Trim();

    /* Check that configuration string is not null and is not a comment */
    if ((configuration_string != "") &&
        (configuration_string.c_str()[0] != ';'))
    {
        /* Extract tokens from string */
        if (extract_token (configuration_string, &first_token, 0) == FAILURE) return (FAILURE);
    }
}
```

Program Style and Methodology

```
/* Check for configuration variable name */
if (AnsiUpperCase(first_token) == AnsiUpperCase(configuration_name))
{
    /* Configuration variable found - Delete it */
    POS_Main_Form->System_Configuration_Memo->Lines->Delete(index);
    break;
}
}

/* The configuration variable was deleted or did not exist. */
/* Append a New configuration variable to the end of the section */
POS_Main_Form->System_Configuration_Memo->Lines->Insert(index,
new_configuration_variable);

/* Save Configuration File */
POS_Main_Form->System_Configuration_Memo->Lines->SaveToFile
(system_configuration_file_name);

/* Success */
return (SUCCESS);
}
```

Program Style and Methodology

```
//-----  
// Read System Configuration File  
//  
// This function reads the system configuration file and sets all the system  
// variables.  
//  
// Note: The POS variable is set to UPPER CASE.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int read_system_configuration_file (void)  
{  
  
/* Access [System] Section Variables */  
// Note: Computer name is found from the Registry.  
  
POS_application = Read_configuration ("System", "Application", "Client");  
POS_daylight_savings = Read_configuration ("System", "Daylight_Savings", "On");  
POS_sound = Read_configuration ("System", "Sound", "OFF");  
  
/* Access [Drives] Section Variables */  
POS_server_drive = Read_configuration ("Drives", "Server", "C:");  
POS_server_name = Read_configuration ("Drives", "Server_Name", "");  
POS_server_shared_directory = Read_configuration ("Drives", "Server_Shared_Directory", "");  
  
POS_local_drive_low_limit = Read_configuration ("Drives", "local_drive_low_limit", "2.0");  
POS_server_drive_low_limit = Read_configuration ("Drives", "server_drive_low_limit", "2.0");  
  
/* Access [Directory] Section Variables */  
POS_project_directory_name = Read_configuration ("Directories", "Project", "PROJ");  
  
/* Fixed directory names */  
POS_system_directory_name = "System";  
POS_log_directory_name = "Log";  
POS_license_directory_name = "License";  
POS_inventory_directory_name = "Inventory";  
POS_accounts_payable_directory_name = "Accounts_Payable";  
POS_accounts_receivable_directory_name = "Accounts_Receivable";  
POS_point_of_sale_directory_name = "POS";  
POS_general_ledger_directory_name = "General_Ledger";  
POS_bank_reconciliation_directory_name = "Bank_Reconciliation";  
POS_purchase_order_directory_name = "Purchase_Order";  
  
POS_pdf_directory_name = "PDF";  
  
/* Access [Options] Section Variables */  
POS_register = Read_configuration ("Options", "Register", "0");  
  
/* Access [General] Section Variables */  
POS_currency_symbol = Read_configuration ("General", "Currency_Symbol", "$");  
  
POS_minimize_screens = Read_configuration ("General", "Minimize_Screens", "OFF");  
POS_fixed_forms = Read_configuration ("General", "Fixed_Forms", "Fixed");  
POS_display_network_messages = Read_configuration ("General", "Display_Network_Messages", "OFF");  
  
POS_auto_pos = Read_configuration ("General", "Auto_POS", "OFF");  
POS_auto_shutdown = Read_configuration ("General", "Auto_Shutdown", "OFF");  
POS_exit_block = Read_configuration ("General", "Exit_Block", "OFF");  
  
/* Access [Programs] Section Variables */  
POS_PDF_Reader = Read_configuration ("Programs", "PDF_Reader", "AcroRd32.exe");  
  
/* Success */  
return (SUCCESS);  
}
```

Program Style and Methodology

```
//-----  
// Log  
//  
// This function appends the log_entry to the end of the current system log.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int Log (char * log_entry)  
  
{  
int file_handle;  
int bytes_written;  
char buffer[MAX_STRING_SIZE + 100];  
  
/* Test if log file operational */  
if (log_file_status == INACTIVE)  
{  
/* Log file is not available */  
return (FAILURE);  
}  
  
/* Does the Log file exist? */  
if (file_exists (system_log_file_name) == FAILURE)  
{  
/* No - Do nothing otherwise infinite loop can occur!!! */  
return (FAILURE);  
}  
  
else  
{  
/* Attempt to open the existing training log file */  
if (file_open (system_log_file_name, &file_handle, OPEN_READ_WRITE_EXCLUSIVE) == FAILURE)  
{  
/* Display error message */  
sprintf (buffer, "Could not open Log File '%s'!!! [errno = %d, %s]",  
system_log_file_name,  
errno,  
_sys_errlist[errno]);  
  
POS_MessageBox (NULL, buffer, "System Log File Error", MB_OK|MB_ICONEXCLAMATION);  
return (FAILURE);  
}  
}  
  
/* Prepend the data and time stamp to the log entry */  
TimeSeparator = ':';  
LongTimeFormat = "hh:mm:ssam/pm";  
ShortTimeFormat = "hh:mm:ssam/pm";  
  
DateSeparator = '/';  
LongDateFormat = "mm/dd/yyyy";  
ShortDateFormat = "mm/dd/yyyy";  
  
/* Create the log file entry */  
sprintf (buffer, "%s %s", DateTimeToStr(Now()).c_str(), log_entry);  
  
/* Restore time format to normal setting */  
LongTimeFormat = "hh:mm:ss am/pm";  
ShortTimeFormat = "hh:mm:ss am/pm";  
  
/* Place file pointer at end of file */  
if (file_seek_end (file_handle) == FAILURE)  
{  
sprintf (buffer, "Could not seek to end of Log File '%s'.", system_log_file_name);  
POS_MessageBox (NULL, buffer, "System Log File Error", MB_OK|MB_ICONEXCLAMATION);  
return (FAILURE);  
}  
}
```

Program Style and Methodology

```
/* File opened successfully - Append log entry */
if (file_write (file_handle, buffer, strlen(buffer)) == FAILURE)
{
    /* Display error message */
    sprintf (buffer, "Log: Error detected when writing log entry to file '%s'!!! [errno =
%d, %s]",
            system_log_file_name,
            errno,
            _sys_errlist[errno]);

    POS_MessageBox (NULL, buffer, "System Log File Error", MB_OK|MB_ICONEXCLAMATION);
    file_close (file_handle);
    return (FAILURE);
}

/* System log entry appended - now append a <CR> */
if (file_write (file_handle, "\n", 1) == FAILURE)
{
    /* No - Display error message */
    sprintf (buffer, "Log: Error detected when writing <CR> to file '%s'!!! [errno = %d, %s]",
            system_log_file_name,
            errno,
            _sys_errlist[errno]);

    POS_MessageBox (NULL, buffer, "System Log File Error", MB_OK|MB_ICONEXCLAMATION);
    file_close (file_handle);
    return (FAILURE);
}

/* All is well, close the file */
file_close (file_handle);

return (SUCCESS);
}
```

Program Style and Methodology

```
//-----  
// Start Calculator  
//  
// This function spawns the calculator.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
void start_calculator (void)  
{  
    int result;  
    char buffer [MAX_STRING_SIZE];  
    char executable [MAX_STRING_SIZE];  
  
    AnsiString system_root;  
  
    /* Get System Root environment variable */  
    system_root = GetEnvironmentVariable("SystemRoot");  
  
    /* Build executable file name */  
    strcpy (executable, system_root.c_str());  
    strcat (executable, "\\System32\\calc.exe");  
  
    /* Start the Calculator */  
    result = spawnlp (P_NOWAIT, executable, executable, NULL);  
  
    /* Success? */  
    if (result == INVALID)  
    {  
        /* No - Tell User to start Windows Calculator manually */  
        POS_MessageBox (NULL,  
            "Windows Calculator not found... Please start manually.",  
            "Calculator",  
            MB_OK|MB_ICONEXCLAMATION);  
  
        /* Could not start Windows Calculator */  
        sprintf (buffer,  
            "Could not start 'calc.exe' process [errno = %d, %s].",  
            errno,  
            _sys_errlist[errno]);  
  
        Log_error (1, buffer);  
    }  
}
```

Program Style and Methodology

```
//-----  
// Start Windows Explorer  
//  
// This function spawns the Microsoft Windows Explorer.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
void start_windows_explorer (void)  
{  
    int result;  
    char buffer [MAX_STRING_SIZE];  
    char executable [MAX_STRING_SIZE];  
  
    AnsiString system_root;  
  
    /* Get System Root environment variable */  
    system_root = GetEnvironmentVariable("SystemRoot");  
  
    /* Build executable file name */  
    strcpy (executable, system_root.c_str());  
    strcat (executable, "\\explorer.exe");  
  
    /* Start Windows Explorer */  
    result = spawnlp (P_NOWAIT, executable, executable, NULL);  
  
    /* Success? */  
    if (result == INVALID)  
    {  
        /* No - Tell User to start Windows Explorer manually */  
        POS_MessageBox (NULL,  
            "Windows Explorer not found... Please start manually.",  
            "Windows Explorer",  
            MB_OK|MB_ICONEXCLAMATION);  
  
        /* Could not start Windows Explorer */  
        sprintf (buffer,  
            "Could not start 'explorer.exe' process [errno = %d, %s].",  
            errno,  
            _sys_errlist[errno]);  
  
        Log_error (1, buffer);  
    }  
}
```

Program Style and Methodology

```
//-----  
// Start Auto Shutdown  
//  
// This function starts the auto shutdown process.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
void start_auto_shutdown (void)  
{  
    int result;  
    char buffer [MAX_STRING_SIZE];  
  
    // !!! Only an Administrator or a user who is part of the Administrators Group can use tsshutdn !!!  
  
    /* Start the shutdown process */  
    result = spawnlp (P_NOWAIT, "tsshutdn.exe", "20", "/powerdown", NULL);  
  
    /* Was an error detected? */  
    if (result == INVALID)  
    {  
        /* Yes - Log error */  
        sprintf (buffer,  
                "Could not start the Auto Shutdown process [errno = %d, %s].",  
                errno,  
                _sys_errlist[errno]);  
  
        Log_error (1, buffer);  
        return;  
    }  
  
    /* Log Auto Shutdown process started */  
    Log_info ("Auto Shutdown process started...");  
}  
  
//-----  
// Play Cash Register Sound  
//  
// This function plays the cash register sound clip.  
//-----  
  
void play_cash_register_sound (void)  
{  
    char sound_file_name [MAX_STRING_SIZE];  
  
    /* Build path using system startup drive and directory */  
    fnmerge (sound_file_name,  
            system_startup_drive.c_str(),  
            system_startup_directory.c_str(),  
            "Cash_Register",  
            ".wav");  
  
    try {  
        POS_Register_Form->MediaPlayer1->FileName = sound_file_name;  
        POS_Register_Form->MediaPlayer1->Open();  
        POS_Register_Form->MediaPlayer1->Play();  
    }  
  
    catch (...)  
    {  
        Status_display ("Sound not found...", TIMEOUT_5_SECONDS);  
    }  
}
```

Program Style and Methodology

```
//-----  
// Play Computer Says No Sound  
//  
// This function plays "The Computer Says No" sound clip.  
//-----  
  
void play_computer_says_no_sound (void)  
{  
char sound_file_name [MAX_STRING_SIZE];  
  
/* Are we signed on as a SUPERVISOR? */  
if (SUPERVISOR_override)  
{  
    /* Yes - Build path using system startup drive and directory */  
    fnmerge (sound_file_name,  
            system_startup_drive.c_str(),  
            system_startup_directory.c_str(),  
            "The Computer Says No",  
            ".wav");  
  
    try {  
        POS_Register_Form->MediaPlayer1->FileName = sound_file_name;  
        POS_Register_Form->MediaPlayer1->Open();  
        POS_Register_Form->MediaPlayer1->Play();  
    }  
  
    catch (...)  
    {  
        Status_display ("Sound not found...", TIMEOUT_5_SECONDS);  
    }  
}  
}  
  
//-----  
// Limit String Size  
//  
// This function shortens the string to the specified limit value. This is  
// used when printing long strings in short fields. If the string is too  
// long it is truncated with the last two string characters being displayed  
// as "...".  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int limit_string_size (char * text, int limit)  
{  
int text_length;  
  
/* Get string length */  
text_length = strlen (text);  
  
/* Determine if text exceeds limit */  
if (text_length > limit)  
{  
    /* Abbreviate text string */  
    text[limit - 2] = '.';  
    text[limit - 1] = '.';  
    text[limit] = 0x00;  
}  
  
/* Success */  
return (SUCCESS);  
}
```

Program Style and Methodology

```
//-----  
// Limit String Size No Dots  
//  
// This function shortens the string to the specified limit value. This is  
// used when printing long strings in short fields. If the string is too  
// long it is simply truncated.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int limit_string_size_no_dots (char * text, int limit)  
{  
    int text_length;  
  
    /* Get string length */  
    text_length = strlen (text);  
  
    /* Determine if text exceeds limit */  
    if (text_length > limit)  
    {  
        /* Abbreviate text string */  
        text[limit] = 0x00;  
    }  
  
    /* Success */  
    return (SUCCESS);  
}  
  
  
  
  
  
  
  
  
  
  
//-----  
// Move Cursor to Home Position  
//  
// This function moves the cursor to the top of the MemoBox.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int move_cursor_to_home_position (TMemo * MemoBox)  
{  
    AnsiString AnsiBuffer;  
  
    /* Abort if memobox does not exist */  
    if (MemoBox == NULL)  
        return (FAILURE);  
  
    /* Display top of report */  
    MemoBox->SelStart = 0;  
  
    /* Success */  
    return (SUCCESS);  
}
```

Program Style and Methodology

```
//-----  
// Set Timer  
//  
// This function sets the timer to the specified timeout value. Since the  
// timers are decremented by a separate thread, a locking mechanism is used  
// to ensure correct operation.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
void set_timer (UINT32 * timer, UINT32 timeout)  
{  
/* Obtain sole access to the timers */  
timer_access->Acquire();  
  
/* Initialize the timer */  
*timer = timeout;  
  
/* Release sole access to the timers */  
timer_access->Release();  
}  
  
//-----  
// Get Timer  
//  
// This function gets the timeout count left in the specified timer.  
// Since the timers are decremented by a separate thread, a locking mechanism  
// is used to ensure correct operation.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
UINT32 get_timer (UINT32 * timer)  
{  
UINT32 timer_value;  
  
/* Obtain sole access to the timers */  
timer_access->Acquire();  
  
/* Get the current timer value */  
timer_value = *timer;  
  
/* Release sole access to the timers */  
timer_access->Release();  
  
/* Return the timer value */  
return (timer_value);  
}
```

Program Style and Methodology

```
//-----  
// Insert Money Symbol  
//  
// This function takes the specified string and inserts a money symbol at the  
// beginning of the number (if space permits).  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int insert_money_symbol (char * buffer)  
{  
    char    money_symbol;  
    char *  pointer;  
  
    pointer = buffer;  
  
    while (*pointer != 0x00)  
    {  
        /* Test for the first numeric digit */  
        if (*pointer != ASCII_SPACE)  
        {  
            /* Step to the previous character */  
            pointer--;  
  
            /* Are we still within the buffer? */  
            if (pointer >= buffer)  
            {  
                /* Yes - Replace character with a money symbol */  
                money_symbol = POS_currency_symbol.c_str()[0];  
                *pointer = money_symbol;  
                return (SUCCESS);  
            }  
  
            /* No - Buffer string not large enough for money symbol */  
            return (FAILURE);  
        }  
  
        /* Step to next character */  
        pointer++;  
    }  
  
    /* No non-space characters in buffer */  
    return (FAILURE);  
}
```

Program Style and Methodology

```
//-----  
// Remove Money Symbol  
//  
// This function takes the specified string and removes the money symbol  
// (if it exists).  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
AnsiString remove_money_symbol (AnsiString money_string)  
{  
    char    input_buffer  [MAX_STRING_SIZE];  
    char    output_buffer [MAX_STRING_SIZE];  
    char    money_symbol;  
  
    AnsiString result_string;  
  
    /* Get the money symbol */  
    money_symbol = POS_currency_symbol.c_str()[0];  
  
    /* Get the money string without additional spaces */  
    strcpy (input_buffer, money_string.Trim().c_str());  
  
    /* Is a money symbol used? */  
    if (input_buffer[0] == money_symbol)  
    {  
        /* Yes - Set the output buffer to the money string without the money symbol */  
        strcpy (output_buffer, &money_string.c_str()[1]);  
    }  
    else  
    {  
        /* No - Set the output buffer to the full money string */  
        strcpy (output_buffer, &money_string.c_str()[0]);  
    }  
  
    /* Convert money string back to an AnsiString */  
    result_string = output_buffer;  
  
    /* No non-space characters in buffer */  
    return (result_string);  
}
```

Program Style and Methodology

```
//-----  
// Random Initialize  
//  
// This function random initializes the specified data structure.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int random_initialize (void * data_structure, long size)  
{  
    char * random_structure;  
    long index;  
    time_t random_seed;  
  
    /* Re-Seed the random number generator */  
    srand((unsigned) time(&random_seed));  
  
    /* Initialization */  
    random_structure = (char *) data_structure;  
  
    /* Randomize the entire data structure */  
    for (index = 0; index < size; index++)  
    {  
        /* Randomize each string in the structure */  
        *random_structure++ = (char) RandomRange (0x00, 0xFF);  
    }  
  
    /* Success */  
    return (SUCCESS);  
}
```

Program Style and Methodology

```
//-----  
// Determine Days In Month  
//  
// This function determines the number of days in a given month [1..12].  
//  
// Returns:  Number of days in the specified month.  
//          FAILURE - Date is not valid.  
//-----  
  
int  determine_days_in_month (int  month, int  year)  
{  
int  days_in_month;  
int  days_in_february;  
  
/*****  
/* Determine days in february */  
*****/  
  
/* Is this a leap year? */  
if ((year % 4) == 0)  
{  
    /* Yes - Is this a new century? */  
    if ((year % 100) == 0)  
    {  
        /* Yes - Is this century divisible by 400? */  
        if ((year % 400) == 0)  
        {  
            /* Yes - It is a true leap year */  
            days_in_february = 29;  
        }  
        else  
        {  
            /* No - It is not a true leap year */  
            days_in_february = 28;  
        }  
    }  
    else  
    {  
        /* Not a new century but still a true leap year */  
        days_in_february = 29;  
    }  
}  
else  
{  
    /* Not a leap year */  
    days_in_february = 28;  
}  
  
/* Determine days in month */  
switch (month)  
{  
    case 1: days_in_month = 31; break;           // Jan  
    case 2: days_in_month = days_in_february; break; // Feb  
    case 3: days_in_month = 31; break;           // Mar  
    case 4: days_in_month = 30; break;           // Apr  
    case 5: days_in_month = 31; break;           // May  
    case 6: days_in_month = 30; break;           // Jun  
    case 7: days_in_month = 31; break;           // Jul  
    case 8: days_in_month = 31; break;           // Aug  
    case 9: days_in_month = 30; break;           // Sep  
    case 10: days_in_month = 31; break;          // Oct  
    case 11: days_in_month = 30; break;          // Nov  
    case 12: days_in_month = 31; break;          // Dec  
    default: days_in_month = 0;  
}  
  
/* Return number of days in the month */  
return (days_in_month);  
}
```

Program Style and Methodology

```
//-----  
// Convert To Double  
//  
// This function converts the text in an EditBox into a double number. If  
// no text is present then the number is set to the default value.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int Convert_To_Double (TEdit * EditBox, double * number, double default_value)  
{  
    char    buffer [MAX_STRING_SIZE];  
    double  result;  
  
    AnsiString  text_number;  
  
    /* Convert the EditBox text to a double number */  
    try {  
        /* Remove leading and trailing spaces */  
        text_number = EditBox->Text.Trim();  
  
        /* Is this a null string? */  
        if (EditBox->Text.Length() == 0)  
        {  
            /* Yes - No text entered - set to default */  
            *number = default_value;  
            return (SUCCESS);  
        }  
  
        /* Is the first text character a currency symbol? */  
        if (text_number.c_str()[0] == POS_currency_symbol.c_str()[0])  
        {  
            /* Yes - Remove any leading currency symbol */  
            strcpy (buffer, &text_number.c_str()[1]);  
            text_number = buffer;  
        }  
  
        /* Try to convert text into a number */  
        result = text_number.ToDouble();  
        *number = result;  
        return (SUCCESS);  
    }  
  
    catch (...)  
    {  
        /* Invalid number - Inform the user */  
        return (FAILURE);  
    }  
}
```

Program Style and Methodology

```
//-----  
// Convert To Int  
//  
// This function converts the text in an EditBox into a integer number.  If  
// no text is present then the number is set to the default value.  
//  
// Returns:  SUCCESS or FAILURE  
//-----  
  
int Convert_To_Int (TEdit * EditBox, double * number, int default_value)  
{  
    char buffer [MAX_STRING_SIZE];  
    int result;  
  
    AnsiString text_number;  
  
    /* Convert the EditBox text to an integer number */  
    try {  
        /* Remove leading and trailing spaces */  
        text_number = EditBox->Text.Trim();  
  
        /* Is this a null string? */  
        if (EditBox->Text.Length() == 0)  
        {  
            /* Yes - No text entered - set to default */  
            *number = default_value;  
            return (SUCCESS);  
        }  
  
        /* Is the first text character a currency symbol? */  
        if (text_number.c_str()[0] == POS_currency_symbol.c_str()[0])  
        {  
            /* Yes - Remove any leading currency symbol */  
            strcpy (buffer, &text_number.c_str()[1]);  
            text_number = buffer;  
        }  
  
        /* Try to convert text into a number */  
        result = text_number.ToInt();  
        *number = result;  
        return (SUCCESS);  
    }  
  
    catch (...)  
    {  
        /* Invalid number - Inform the user */  
        return (FAILURE);  
    }  
}
```

Program Style and Methodology

```
//-----  
// Get Version Numbers  
//  
// This function reads the version information table.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int get_version_numbers (char * file_name,  
                        char * version_numbers)  
{  
    char * version_table;  
    char * version_number_string;  
    DWORD version_table_size;  
    unsigned int parameter_length;  
    AnsiString version_search_string;  
    AnsiString file_version;  
  
    /* Initialization */  
    strcpy (version_numbers, "0.0.0.0");  
  
    /* Get the version table size */  
    version_table_size = GetFileVersionInfoSize(file_name, &version_table_size);  
  
    /* Does the version table exist? */  
    if (version_table_size == 0)  
    {  
        /* No - Exit */  
        return (FAILURE);  
    }  
  
    /* Allocate dynamic memory for the version table */  
    version_table = (char *) malloc(version_table_size);  
  
    /* Was the version table memory allocated okay? */  
    if (version_table == NULL)  
    {  
        /* No - Exit */  
        return (FAILURE);  
    }  
  
    /* Build version table search string */  
    version_search_string = "StringFileInfo\\040904E4\\";  
    version_search_string += "FileVersion";  
  
    /*Read the Version Table */  
    if (GetFileVersionInfo(file_name, 0, version_table_size, version_table) == 0)  
    {  
        /* Failed - Exit */  
        free (version_table);  
        return (FAILURE);  
    }  
  
    /* Obtain the version numbers */  
    if (VerQueryValue (version_table,  
                      version_search_string.c_str(),  
                      (void **)&version_number_string,  
                      &parameter_length))  
    {  
        /* Get the version number string */  
        strcpy (version_numbers, version_number_string);  
    }  
  
    /* Release version table memory */  
    free (version_table);  
  
    /* Success */  
    return (SUCCESS);  
}
```

Program Style and Methodology

```
//-----  
// Extract Quad Numbers  
//  
// This function extracts four integer numbers from a quad string such as  
// "1234.2.34341.2".  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int extract_quad_numbers (char * quad_number_string,  
                          int * number1,  
                          int * number2,  
                          int * number3,  
                          int * number4)  
{  
char * string_ptr;  
char * decimal_point_ptr;  
  
/*****/  
/* First Number */  
/*****/  
  
/* Initialize */  
string_ptr = quad_number_string;  
  
/* Find first decimal point */  
decimal_point_ptr = StrPos (string_ptr, ".");  
  
if (decimal_point_ptr == NULL)  
    return (FAILURE);  
  
/* Set decimal point to string terminator */  
*decimal_point_ptr = 0x00;  
  
/* Extract first number */  
*number1 = atoi (string_ptr);  
  
/*****/  
/* Second Number */  
/*****/  
  
/* Set string ptr to start of second number */  
string_ptr = ++decimal_point_ptr;  
  
/* Find next decimal point */  
decimal_point_ptr = StrPos (string_ptr, ".");  
  
if (decimal_point_ptr == NULL)  
    return (FAILURE);  
  
/* Set decimal point to string terminator */  
*decimal_point_ptr = 0x00;  
  
/* Extract second number */  
*number2 = atoi (string_ptr);
```

Program Style and Methodology

```
/*
*****
/* Third Number */
*****

/* Set string ptr to start of third number */
string_ptr = ++decimal_point_ptr;

/* Find next decimal point */
decimal_point_ptr = StrPos (string_ptr, ".");

if (decimal_point_ptr == NULL)
    return (FAILURE);

/* Set decimal point to string terminator */
*decimal_point_ptr = 0x00;

/* Extract third number */
*number3 = atoi (string_ptr);

/*
*****
/* Fourth Number */
*****

/* Set string ptr to start of fourth number */
string_ptr = ++decimal_point_ptr;

/* Extract fourth number */
*number4 = atoi (string_ptr);

/* Success */
return (SUCCESS);
}

```

Program Style and Methodology

```
//-----  
// Get Major Version Number  
//  
// This function reads the version information table and extracts the major  
// version number.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int get_major_version_number (char * file_name,  
                             int * major_version_number)  
{  
    int number1;  
    int number2;  
    int number3;  
    int number4;  
    char version_numbers [MAX_STRING_SIZE];  
  
    /* Initialization */  
    *major_version_number = 0;  
  
    /* Obtain version numbers */  
    if (get_version_numbers (file_name, version_numbers) == FAILURE)  
        return (FAILURE);  
  
    /* Extract the four independent version numbers */  
    if (extract_quad_numbers (version_numbers, &number1, &number2, &number3, &number4) == FAILURE)  
        return (FAILURE);  
  
    /* Extract major version number */  
    *major_version_number = number1;  
  
    /* Success */  
    return (SUCCESS);  
}
```

Program Style and Methodology

```
//-----  
// Get Minor Version Number  
//  
// This function reads the version information table and extracts the minor  
// version number.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int get_minor_version_number (char * file_name,  
                             int * minor_version_number)  
{  
    int number1;  
    int number2;  
    int number3;  
    int number4;  
    char version_numbers [MAX_STRING_SIZE];  
  
    /* Initialization */  
    *minor_version_number = 0;  
  
    /* Obtain version numbers */  
    if (get_version_numbers (file_name, version_numbers) == FAILURE)  
        return (FAILURE);  
  
    /* Extract the four independent version numbers */  
    if (extract_quad_numbers (version_numbers, &number1, &number2, &number3, &number4) == FAILURE)  
        return (FAILURE);  
  
    /* Extract minor version number */  
    *minor_version_number = number2;  
  
    /* Success */  
    return (SUCCESS);  
}
```

Program Style and Methodology

```
//-----  
// Print MemoBox  
//  
// This function prints the specified MemoBox on to the printer.  
//  
// Returns: SUCCESS or FAILURE  
//-----  
  
int print_memobox (TPrinter * OUT_Printer, int orientation, TStrings * Document, char  
* Title)  
{  
    int index;  
    int line_count;  
    int max_pages;  
    int page_count;  
    int page_width;  
    int page_height;  
    int text_height;  
    int max_document_lines;  
    int max_document_pages;  
    int max_printed_lines_per_page;  
    int max_document_lines_per_page;  
    int number_of_document_lines;  
    int number_of_document_pages;  
    char buffer [MAX_STRING_SIZE];  
    struct date d;  
    struct time t;  
  
    /* Get number of lines in document */  
    number_of_document_lines = Document->Count;  
  
    /* Is the document empty? */  
    if (number_of_document_lines == 0)  
        return (FAILURE);  
  
    /* Get current Date and Time Information */  
    getdate(&d);  
    gettime(&t);  
  
    /* Determine maximum number of lines to the page */  
    page_width = OUT_Printer->PageWidth - PRINTER_LEFT_MARGIN; // - PRINTER_RIGHT_MARGIN;  
    page_height = OUT_Printer->PageHeight - PRINTER_TOP_MARGIN; // - PRINTER_BOTTOM_MARGIN;  
  
    OUT_Printer->Canvas->Font->Name = "Courier";  
    OUT_Printer->Canvas->Font->Pitch = fpFixed;  
    OUT_Printer->Canvas->Font->Size = 10;  
  
    text_height = OUT_Printer->Canvas->TextHeight(Document->Strings[0]);  
  
    if (text_height == 0)  
    {  
        for (index = 0; index < number_of_document_lines; index++)  
        {  
            /* Get the text height of another line */  
            text_height = OUT_Printer->Canvas->TextHeight(Document->Strings[index]);  
  
            /* Is the text height > 0 ? */  
            if (text_height > 0)  
                break;  
        }  
    }  
  
    /* Did we obtain a valid text height? */  
    if (text_height == 0)  
    {  
        /* No - Cancel print */  
        return (FAILURE);  
    }  
}
```

Program Style and Methodology

```
/* Is Portrait orientation required? */
if (orientation == PORTRAIT)
{
    /* Yes - Print page in Portrait Format */
    OUT_Printer->Orientation = poPortrait;

    /* Determine printing parameters */
    max_printed_lines_per_page = page_height / text_height;
    max_document_lines_per_page = max_printed_lines_per_page - 4;    //2 lines for header and
                                                                    //2 lines for footer

    number_of_document_pages = (number_of_document_lines / max_document_lines_per_page) + 1;
}
else
{
    /* No - Print page in Landscape Format */
    OUT_Printer->Orientation = poLandscape;

    /* Determine printing parameters */
    max_printed_lines_per_page = page_width / text_height;
    max_document_lines_per_page = max_printed_lines_per_page - 4;    //2 lines for header and
                                                                    //2 lines for footer

    number_of_document_pages = (number_of_document_lines / max_document_lines_per_page) + 1;
}

/* Initialization */
line_count = 0;
page_count = 1;

/* Start Printing */
OUT_Printer->Title = Title;
OUT_Printer->BeginDoc();

/* Output each line */
for (index = 0; index < number_of_document_lines; index++)
{
    /* Print next line of document */
    OUT_Printer->Canvas->TextOut (PRINTER_LEFT_MARGIN,
                                PRINTER_TOP_MARGIN + (line_count++ * text_height),
                                Document->Strings[index]);

    /* Print document and footer */
    if (line_count >= max_document_lines_per_page)
    {
        /* Print Blank Line */
        OUT_Printer->Canvas->TextOut (PRINTER_LEFT_MARGIN,
                                    PRINTER_TOP_MARGIN + (line_count++ * text_height),
                                    "");

        /* Print Footer */
        sprintf (buffer, "Date: %4d-%02d-%02d   Time: %02d-%02d-%02d   Page: %d / %d",
                d.da_year, d.da_mon, d.da_day, t.ti_hour, t.ti_min, t.ti_sec, page_count,
                number_of_document_pages);

        OUT_Printer->Canvas->TextOut (PRINTER_LEFT_MARGIN,
                                    PRINTER_TOP_MARGIN + (line_count * text_height),
                                    buffer);

        /* Reset line counter */
        line_count = 0;

        /* Increment page counter */
        page_count++;
    }
}
```

Program Style and Methodology

```
/* Still more to print? */
    if (index + 1 < number_of_document_lines)
    {
        /* Yes - Start a new page */
        OUT_Printer->NewPage();
    }
}

/* Write blank lines until end of page */
for (index = line_count; index <= max_document_lines_per_page; index++)
{
    /* Print Blank Line */
    OUT_Printer->Canvas->TextOut (PRINTER_LEFT_MARGIN,
                                  PRINTER_TOP_MARGIN + (line_count++ * text_height),
                                  "");
}

/* Print Footer */
sprintf (buffer,
        "Date: %4d-%02d-%02d   Time: %02d-%02d-%02d   Page: %d / %d",
        d.da_year,
        d.da_mon,
        d.da_day,
        t.ti_hour,
        t.ti_min,
        t.ti_sec,
        page_count,
        number_of_document_pages);

OUT_Printer->Canvas->TextOut (PRINTER_LEFT_MARGIN,
                              PRINTER_TOP_MARGIN + (line_count * text_height),
                              buffer);

/* Stop Printing */
OUT_Printer->EndDoc();

/* Success */
return (SUCCESS);
}
```

Program Style and Methodology

3 Appendix

3.1 Code Editor

3.1.1 Use of Tabs versus Spaces

The code editor each programmer uses must be configured such that it replaces the TAB character with the equivalent number of space characters. This will permit a file, which has been modified by different programmers/editors to be displayed and/or printed with the correct indentation.