7. Appendices

7.1. Performance report

The following is an example of a performance report showing a summary of the load shifted on a given mine for a period of a month.

REMS Monthly report for Kopanang
1 October 2006 to 31 October 2006

08 November 2006

REMS
UNDER PROMISE
OVER DELIVER

HVAC
INTERNATIONAL

Table of Contents

Introduction 2
Performanc measurement 2
REMS Savings 2
Total energy used and peak time load shift 4
Missed Opportunities 5
Condonables 5
Conclusion 5

Introduction

The purpose of this report is to inform the reader of the savings the REMS system realised at Kopanang for the period from 1 October 2006 to 31 October 2006.

Performanc measurement

The REMS system has been operational for ???? months. See the table below for the savings achieved.

Month	Proposed monthly saving possible	Monthly saving achieved	Unrealised potential / Over performance	Accumulated proposed savings possible	Accumulated actual savings
July 2006	R 0	R 294 060	R 294 060	R 0	R 294 060
August 2006	R 0	R 306 497	R 306 497	R 0	R 600 557
September 2006	R 0	R 32 209	R 32 209	R 0	R 632 766
October 2006	R 0	R 57 348	R 57 348	R 0	R 690 114

Table: Actual savings

REMS Savings

The total saving for the period is R 57 348. The average MW reduction for the evening peak, excluding condonable days, is 9.43 MW. The figure below shows daily cost saving that was achieved.

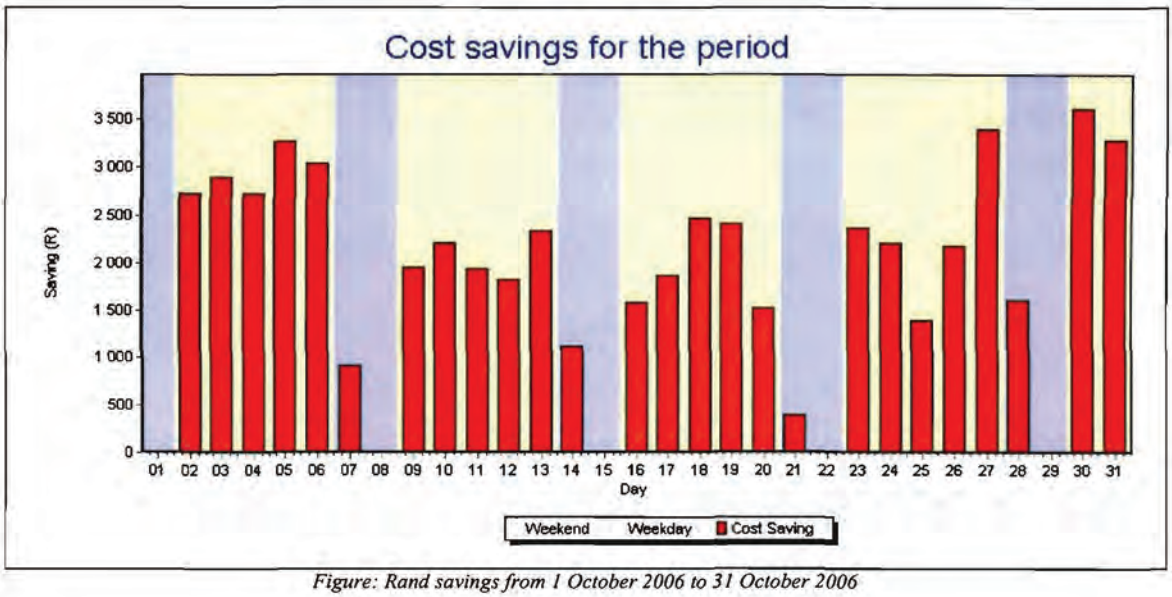


Figure: Rand savings from 1 October 2006 to 31 October 2006

Total energy used and peak time load shift

The first figure below shows the average load and baseline (MW) for 1 October 2006 to 31 October 2006. The second figure shows the morning and evening loadshift (MW) for the same period.

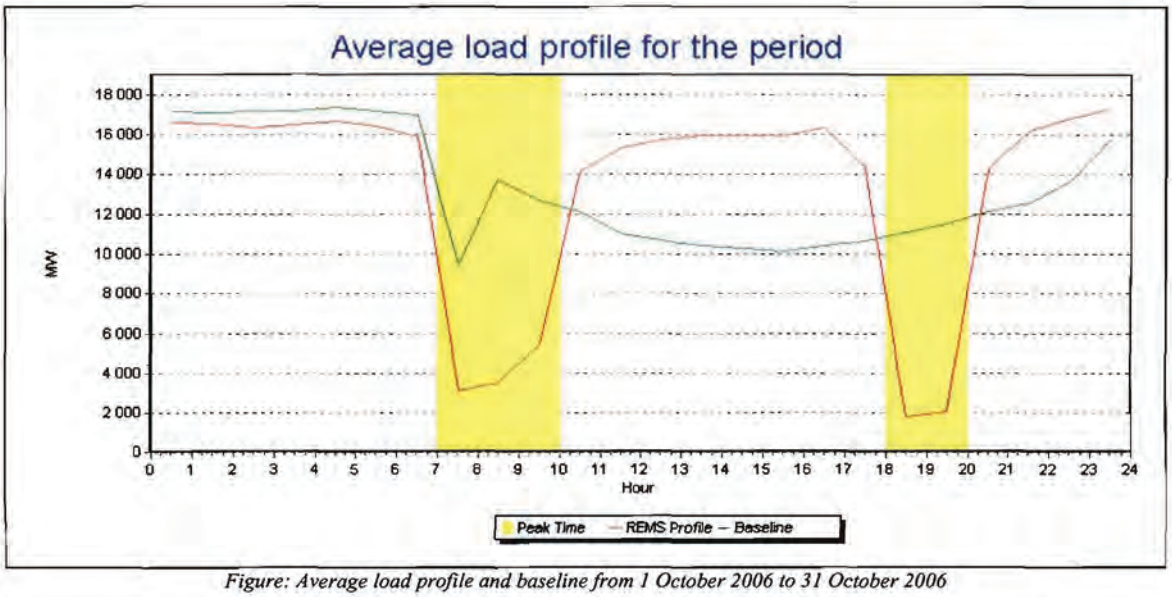


Figure: Average load profile and baseline from 1 October 2006 to 31 October 2006

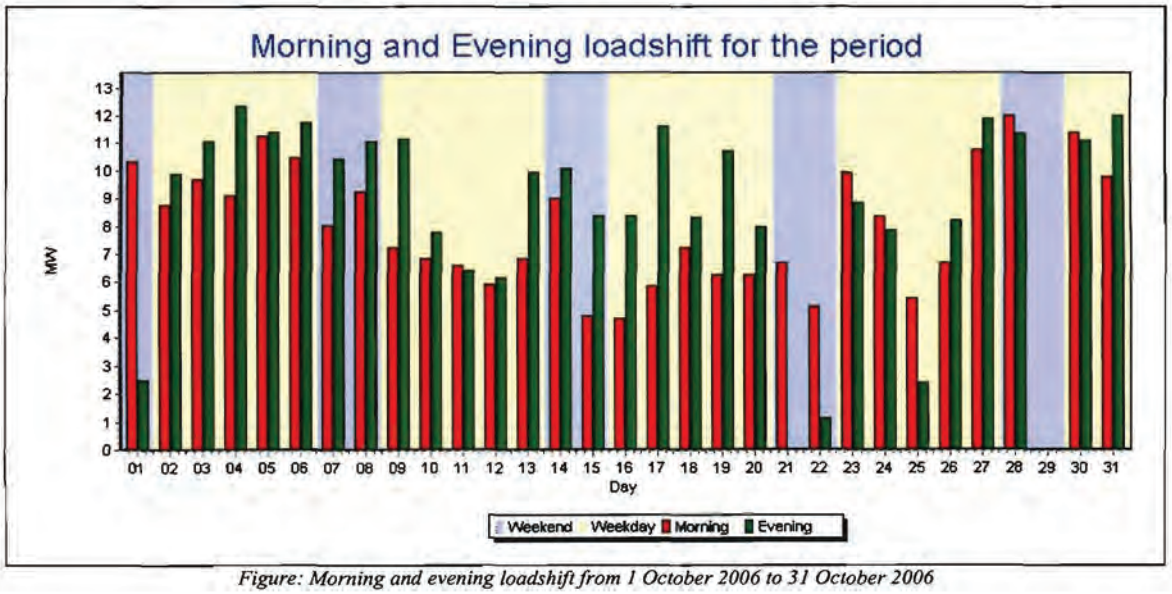


Figure: Morning and evening loadshift from 1 October 2006 to 31 October 2006

Missed Opportunities

Condonables

There is 1 condonable day for this period.

Day	Condonable
29 October	Data loss

Table: Condonable days

Conclusion

For the period form 1 October to 31 October the REMS system saved a total of **R 57 348** for Kopanang, with a evening peak load reduction of **9.43 MW**. If all the missed opportunities were realised the costs saving would have been **R 81 148**.

7.2. Daily performance report

The following shows a daily performance report, showing the daily performance achieved on a given project.

REMS Daily report for Kopanang Pumps
2006-09-19

08 November 2006

REMS
UNDER PROMISE
OVER DELIVER

HVAC
INTERNATIONAL

Load shift results for Kopanang Pumps for 19 September 2006

Parameter	Value
Morning Peak	0.37 MW
Evening Peak	4.47 MW
Average Evening Peak for month	4.47 MW
Contractual	3.00 MW
Cost Saving	R 454
Energy usage	141.80 MWh
System on Manual	0 % of the day

Table: Summary of day

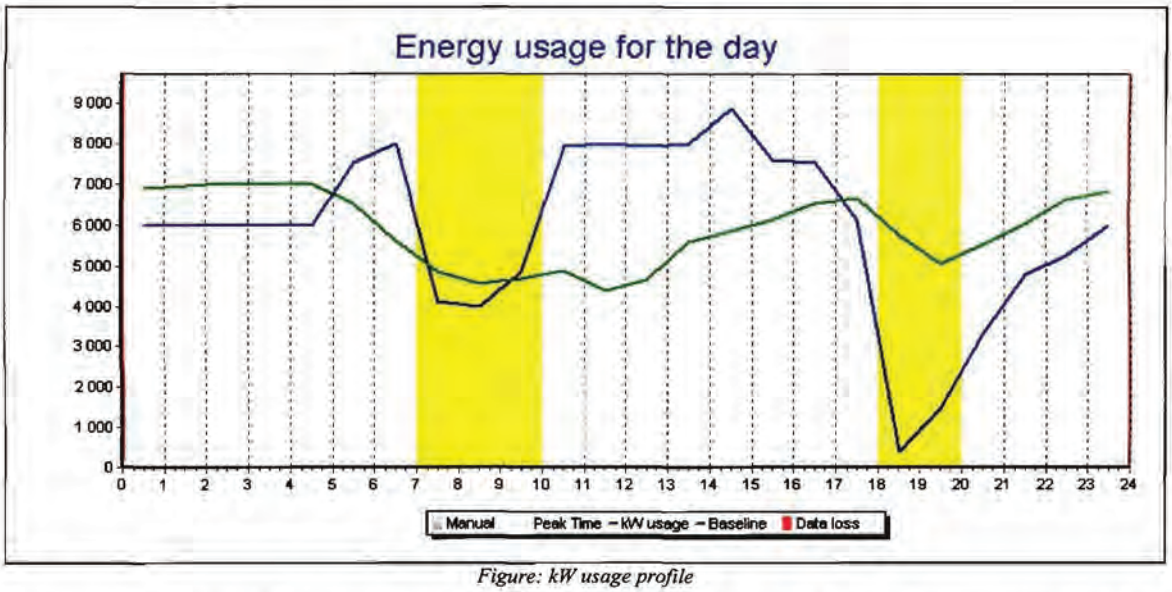


Figure: kW usage profile

The above graphs shows the energy usage profile for the day with manual overrides and data loss overlays.

Summary for 38 Controller

The figure below shows the detail description for 38 Controller for the day. The status is the actual amount of pumps running and schedule is the amount of pumps REMS requests.

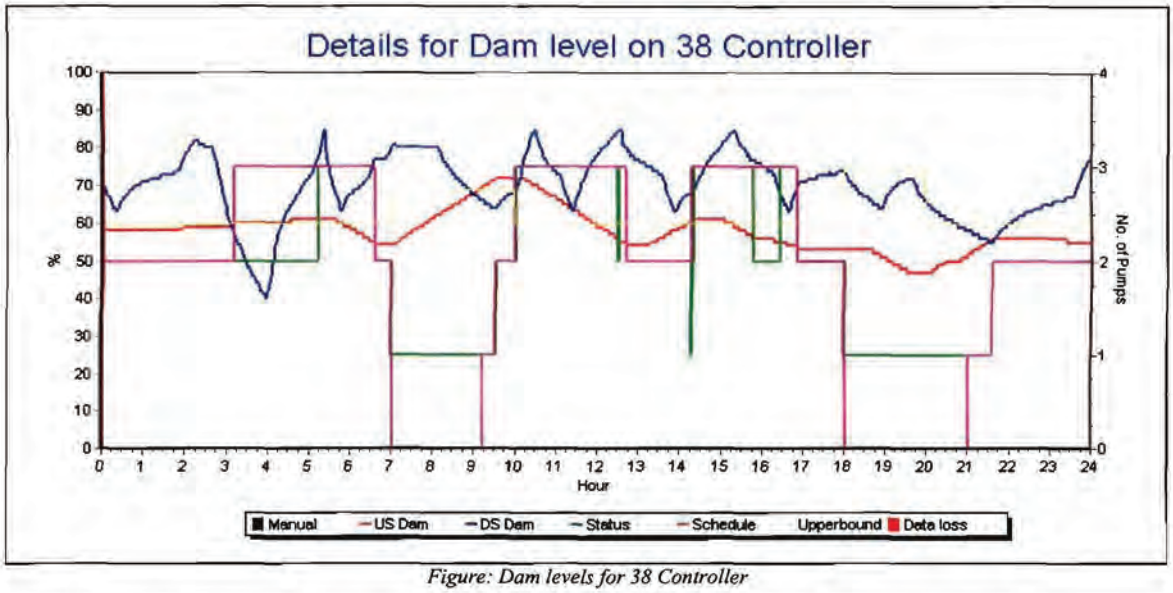


Figure: Dam levels for 38 Controller

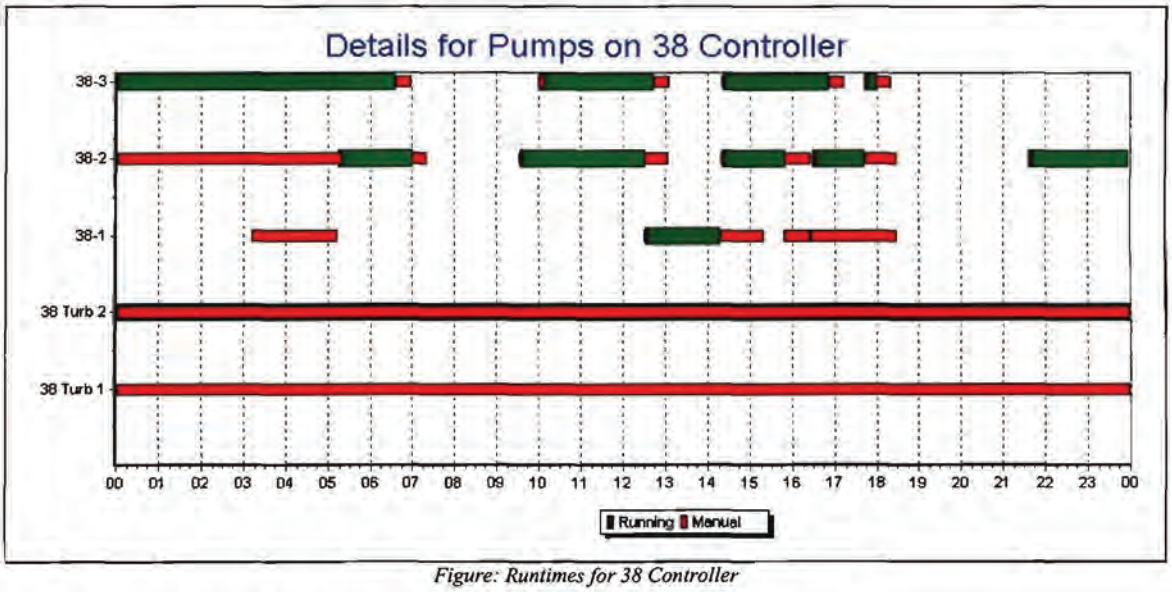


Figure: Runtimes for 38 Controller

The above figure shows the actual runtimes for the pumps in this level.

Summary for 75 Controller

The figure below shows the detail description for 75 Controller for the day. The status is the actual amount of pumps running and schedule is the amount of pumps REMS requests.

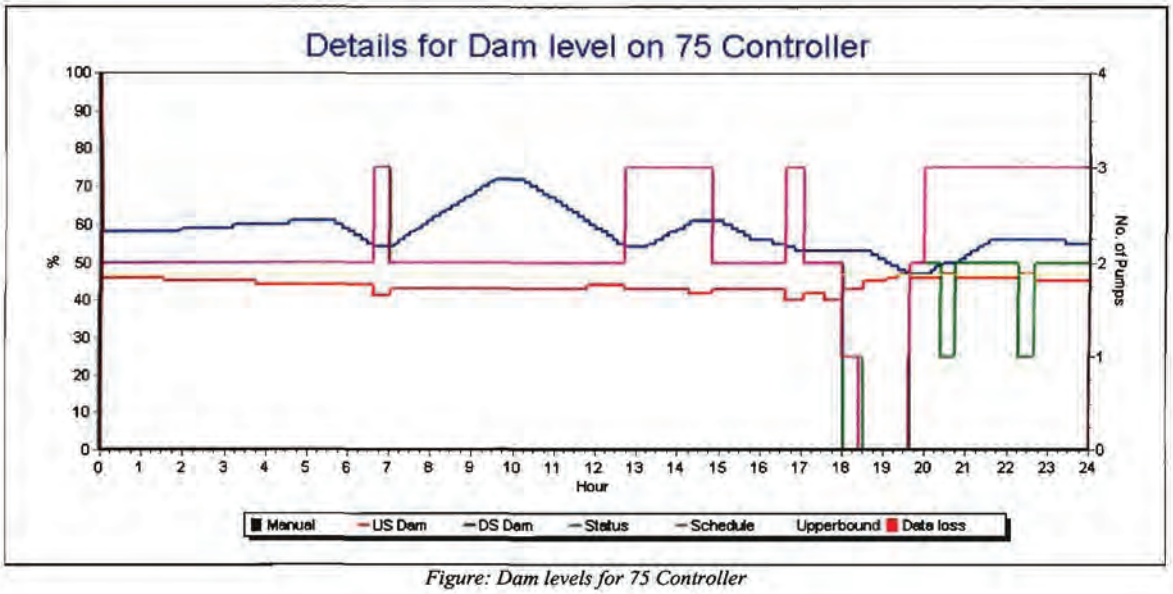


Figure: Dam levels for 75 Controller

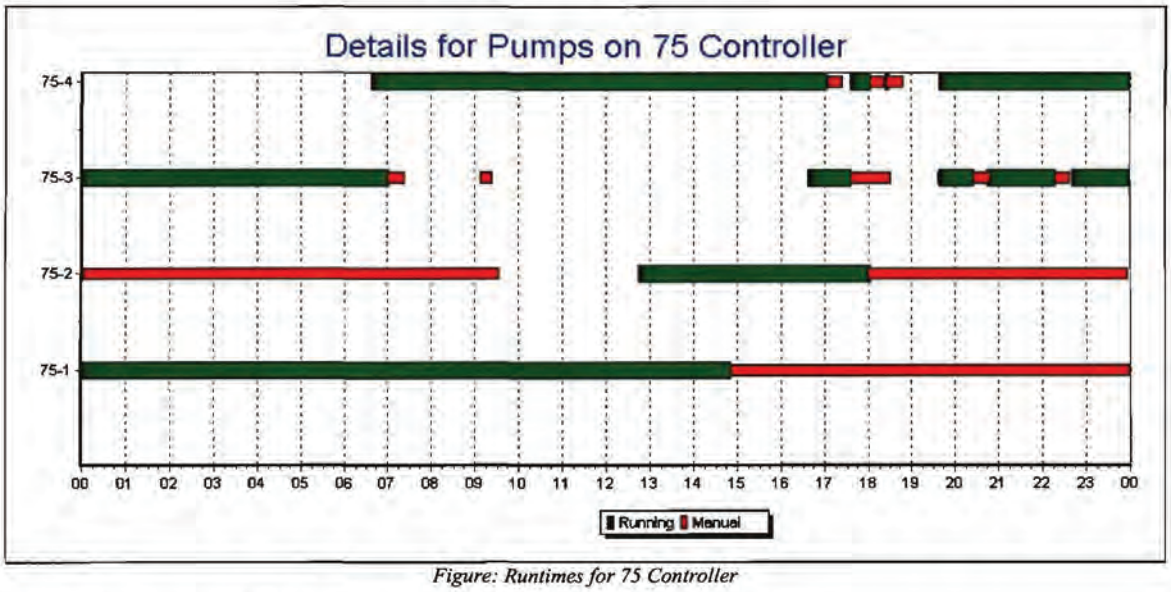


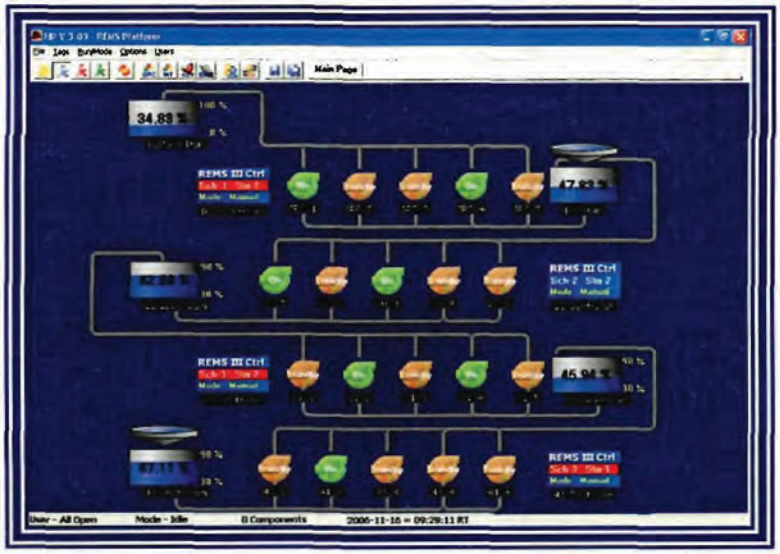
Figure: Runtimes for 75 Controller

The above figure shows the actual runtimes for the pumps in this level.

7.3. Code layout of REMS

This appendix goes into an in depth look at the code and code structure of REMS.

REMS CODE LAYOUT



This appendix presents a layout and discussion of the Delphi code of REMS – the system presented in this thesis.

Table of contents

Table of contents ii

List of Figures iii

1. INTRODUCTION 1

1.1. PLATFORM 3

1.1.1. MAINFORM 3

1.1.2. PROJECT 8

1.1.3. PLATFORM ICON 11

1.1.4. ALARM HANDLER 14

1.1.5. TAG HANDLER 16

1.1.6. SMS SENDER 20

1.1.7. USER MANAGER 21

1.2. PUMP DLL 23

1.2.1. PUMP 24

1.2.2. PUMP EDITOR 28

1.2.3. PUMP VIEWER 31

1.3. DAM DLL 34

1.3.1. DAM 34

1.3.2. DAM EDITOR 38

1.3.3. DAM VIEWER 41

1.4. PUMP GROUP CONTROLLER DLL 44

1.4.1. PUMPGROUP 44

1.4.2. PUMPGROUP CONTROLLER 48

1.4.3. PUMPGROUP EDITOR 50

1.4.4. PUMPGROUP VIEWER 53

List of Figures

Figure 1-1 Unit structure of the Platform 3
Figure 1-2 Platform with Icon 11
Figure 1-3 Pump DLL used to represent actual pumps 23
Figure 1-4 Unit structure of the Pump DLL 24
Figure 1-5 Pump DLL Editor 29
Figure 1-6 Pump Viewer 32
Figure 1-7 Dam DLL 34
Figure 1-8 Unit structure of the Dam DLL 35
Figure 1-9 Dam DLL Editor 39
Figure 1-10 Dam Viewer 42
Figure 1-11 Unit structure of the Pump group controller DLL 44
Figure 1-12 Pump group controller editor 50
Figure 1-13 Pump group controller DLL information panel 53

1. Introduction

This thesis present the development of a system that is used to shift electrical load and realise electrical running cost reductions on water pumping systems. This system is called REMS.

The REMS was written/coded in Delphi. Delphi is a programming language focussed on RAD (Rapid Application Development). Delphi makes it easy to develop reliable Microsoft Windows based application.

REMS was developed in four main parts. These parts are:

1. Platform – This is the main part of the application. It incorporates the GUI (Graphical User Interface) to which the user interacts.

2. Pump DLL (Dynamic Linked Library). The Pump DLL represents a pump with its corresponding inputs, outputs and other characteristics. The user interacts to the Pump DLL via the Platform GUI.

3. Dam DLL. The Dam DLL is used to connect to and represent an actual physical dam. The Dam DLL conveys real-time information about the actual dam to the user.

4. Pump Group Controller DLL. This DLL embodies the control philosophy that is used to shift electrical load and realise reduced running cost reductions.

REMS was developed and coded according to the OOP (Object Oriented Programming) standard. This implies that the code is divided into self sustained elements that can function individually. These elements are referred to as *units*.

REMS consists of units, each of which focuses on specific tasks. This appendix gives a broad overview of these units and corresponding tasks. Each unit consists of an interface and an implementation section. The interface lists the procedures, functions and properties within the unit. The implementations show the embodiment of the functions and procedures. To enhance readability only the interfaces of the units are discussed.

1.1. Platform

The Platform is built up of a number of units. The next figure illustrates these units and how they relate to each other.

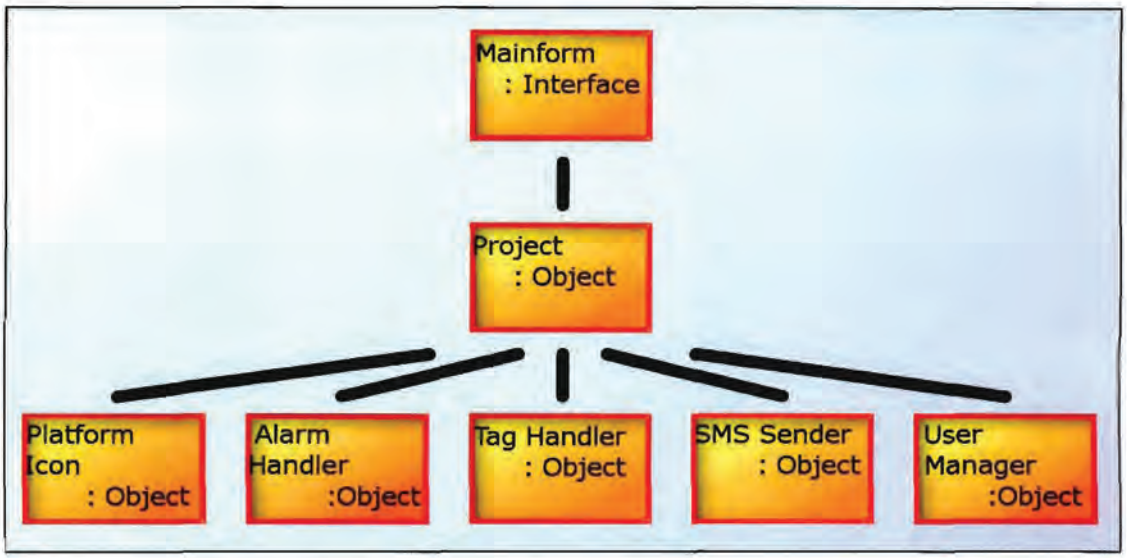


Figure 1-1 Unit structure of the Platform

The following sections are a short discussion of each unit that make up the Platform.

1.1.1. Mainform

The Mainform of the application is the main interface of the application and also the parent of all the other units. The following shows the *interface* of the Mainform.

```
unit MainFrm;

interface

uses
    // Herargie
    Project,
    InformationPanel,

    AutoControlTerminationFrm,
    HVACIPlatformTypes,
    WhiteUnit,

    GR32,
    GR32_Image,
    GR32_Layers,
    Windows, Messages, SysUtils, Variants, Classes,
    Graphics, Controls, Forms,
    Dialogs, Menus, ExtCtrls, ComCtrls, ToolWin, ImgList, AdvListV,
  StdCtrls;
```

```
type
  TMainForm = class(TForm)
    MainMenu: TMainMenu;
    MainMenu_File: TMenuItem;
    MainMenu_Options: TMenuItem;
    MainMenu_Options_PlatformOptions1: TMenuItem;
    MainMenu_Options_RefreshScreen: TMenuItem;
    MainMenu_Tags: TMenuItem;
    MainMenu_Tags_InternalTags: TMenuItem;
    MainMenu_Tags_OPCTagsOptions: TMenuItem;
    PopupMenuWorkSpace: TPopupMenu;
    PopupMenuWorkSpaceCreateNewIcon: TMenuItem;
    PopupMenuWorkSpaceAutoCreateNewIcon: TMenuItem;
    Image32: TImage32;
    SaveDialog: TSaveDialog;
    MainMenu_File_SaveProjectAs: TMenuItem;
    OpenDialog: TOpenDialog;
    MainMenu_File_SaveProject: TMenuItem;
    MainMenu_File_LoadProject: TMenuItem;
    StatusBar: TStatusBar;
    MainMenu_Run: TMenuItem;
    MainMenu_Run_RunOptions: TMenuItem;
    PopupMenuWorkSpaceDeleteIcon: TMenuItem;
    PopupMenuWorkSpaceShowIconOptionForm: TMenuItem;
    PopupMenuWorkSpacePlatformSetupForm: TMenuItem;
    Toolbar: TToolBar;
    MainMenu_Users: TMenuItem;
    MainMenu_Users_SwitchUsers: TMenuItem;
    MainMenu_Users_UserManager: TMenuItem;
    ToolButtonSwitchUsers: TToolButton;
    ToolbarImager: TImageList;
    ToolButtonUserManager: TToolButton;
    ToolButton1: TToolButton;
    ToolButtonRunOptions: TToolButton;
    MainMenu_Run_AutoControl: TMenuItem;
    MainMenu_Run_ManualControl: TMenuItem;
    MainMenu_Run_IdleMode: TMenuItem;
    MainMenu_Run_EditMode: TMenuItem;
    ToolButton2: TToolButton;
    ToolButtonInternalTags: TToolButton;
    ToolButtonOPCOptions: TToolButton;
    ToolButtonAutoControl: TToolButton;
    ToolButtonManualControl: TToolButton;
    ToolButtonIdleMode: TToolButton;
    ToolButton6: TToolButton;
    ToolButtonEditorMode: TToolButton;
    PopupMenuStatusBar: TPopupMenu;
    PopupMenuStatusBarSwichUsers: TMenuItem;
    PopupMenuStatusBarResetSimulatedTime: TMenuItem;
    PopupMenuStatusBarTimeSettings: TMenuItem;
    PopupMenuWorkSpaceIconVisibility: TMenuItem;
    TabControlPageNavigator: TTabControl;
    ToolButton3: TToolButton;
    PopupMenuPageNavigator: TPopupMenu;
    PopupMenuPageNavigator_AddPage: TMenuItem;
    PopupMenuPageNavigator_DeletePage: TMenuItem;
    PopupMenuPageNavigator_RenamePage: TMenuItem;
    MainMenu_File_NewProject: TMenuItem;
    PopupMenuWorkspaceAddLine: TMenuItem;
    MainMenu_File_Exit: TMenuItem;
    MainMenu_File_BackUp: TMenuItem;
    ToolButtonCreateBackup: TToolButton;
    ToolButton5: TToolButton;
    MainMenu_Options_FindIconsOutsideBoundaries: TMenuItem;
    ToolButtonSave: TToolButton;
    PopupMenuWorkSpaceSwitchLineOrientation: TMenuItem;
    PopupMenuWorkSpaceSwitchLineSource: TMenuItem;
    ListViewActions: TAdvListView;
    PopupMenuActionListView: TPopupMenu;
    PopupMenuListViewActions_HideAll: TMenuItem;
    PopupMenuListViewActions_ActionOptions: TMenuItem;
    PopupMenuListViewActions_HideSingle: TMenuItem;
    ListViewAlarms: TAdvListView;
```

```
    PopupMenuListViewActions_DisplayAll: TMenuItem;
    PopupMenuWorkSpaceViewActionsMenu: TMenuItem;
    MainMenu_Options_PlatformThemeManager: TMenuItem;
    PopupMenuMoveLinetoFront: TMenuItem;
    PopupMenuSendLinetoBack: TMenuItem;
    toolBtnAlarms: TToolButton;
    toolBtnSMS: TToolButton;

    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
    procedure ToolbarResize(Sender: TObject);

    procedure Image32DblClick(Sender: TObject);
    procedure   Image32MouseDown(Sender:   TObject;   Button:   TMouseButton;   Shift:
TShiftState; X, Y: Integer; Layer: TCustomLayer);
    procedure  Image32MouseMove(Sender:  TObject;  Shift:  TShiftState;  X,Y:  Integer;
Layer: TCustomLayer);
    procedure    Image32MouseUp(Sender:    TObject;    Button:    TMouseButton;    Shift:
TShiftState; X, Y: Integer; Layer: TCustomLayer);
    procedure Image32Resize(Sender: TObject);
    procedure  Image32MouseWheel(Sender:  TObject;  Shift:  TShiftState;  WheelDelta:
Integer; MousePos: TPoint; var Handled: Boolean);
    procedure Image32KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

    procedure MainMenu_File_SaveProjectClick(Sender: TObject);
    procedure MainMenu_File_SaveProjectAsClick(Sender: TObject);
    procedure MainMenu_File_BackUpClick(Sender: TObject);
    procedure MainMenu_File_LoadProjectClick(Sender: TObject);
    procedure MainMenu_File_NewProjectClick(Sender: TObject);
    procedure MainMenu_File_ExitClick(Sender: TObject);

    procedure MainMenu_Tags_InternalTagsClick(Sender: TObject);
    procedure MainMenu_Tags_OPCTagsOptionsClick(Sender: TObject);

    procedure MainMenu_Options_PlatformOptions1Click(Sender: TObject);
    procedure MainMenu_Options_RefreshScreenClick(Sender: TObject);
    procedure MainMenu_Options_FindIconsOutsideBoundariesClick(Sender: TObject);

    procedure MainMenu_Run_AutoControlClick(Sender: TObject);
    procedure MainMenu_Run_ManualControlClick(Sender: TObject);
    procedure MainMenu_Run_IdleModeClick(Sender: TObject);
    procedure MainMenu_Run_EditModeClick(Sender: TObject);
    procedure MainMenu_Run_RunOptionsClick(Sender: TObject);

    procedure MainMenu_Users_SwitchUsersClick(Sender: TObject);
    procedure MainMenu_Users_UserManagerClick(Sender: TObject);

    // PopupMenuWorkspace
        procedure PopupMenuWorkSpacePopup(Sender: TObject);
            Procedure LoadPopupMenuDLLNames;
    procedure PopupMenuWorkSpaceCreateNewIconClick(Sender: TObject);
    Procedure PopupMenuWorkSpaceDeleteIconClick(Sender: TObject);
    procedure PopupMenuWorkSpacePlatformSetupFormClick(Sender: TObject);
    procedure PopupMenuWorkSpaceShowIconOptionFormClick(Sender: TObject);

    procedure PopupMenuWorkspaceAddLineClick(Sender: TObject);
    procedure PopupMenuWorkSpaceSwitchLineOrientationClick(Sender: TObject);
    procedure PopupMenuWorkSpaceSwitchLineSourceClick(Sender: TObject);

    procedure PopupMenuWorkSpaceIconVisibilityClick(Sender: TObject);
    procedure PopupMenuWorkSpaceViewActionsMenuClick(Sender: TObject);

    // List VIew Actions
    procedure ListViewActionsDblClick(Sender: TObject);
    // Popup Menu Actions List View
    procedure PopupMenuActionListViewPopup(Sender: TObject);

    procedure PopupMenuListViewActions_DisplayAllClick(Sender: TObject);
    procedure PopupMenuListViewActions_HideSingleClick(Sender: TObject);
    procedure PopupMenuListViewActions_HideAllClick(Sender: TObject);
    procedure PopupMenuListViewActions_ActionOptionsClick(Sender: TObject);

    // Popup Menu StatusBar
```

```
        procedure PopupMenuStatusBarPopup(Sender: TObject);
    procedure PopupMenuStatusBarSwichUsersClick(Sender: TObject);
    procedure PopupMenuStatusBarTimeSettingsClick(Sender: TObject);
    procedure PopupMenuStatusBarResetSimulatedTimeClick(Sender: TObject);

    // Popup Menu Tab Control Page Navigator
    procedure PopupMenuPageNavigator_AddPageClick(Sender: TObject);
    procedure PopupMenuPageNavigator_RenamePageClick(Sender: TObject);

    // Tool Buttons
    procedure ToolButtonOPCOptionsClick(Sender: TObject);
    procedure ToolButtonInternalTagsClick(Sender: TObject);

    procedure ToolButtonEditorModeClick(Sender: TObject);
    procedure ToolButtonIdleModeClick(Sender: TObject);
    procedure ToolButtonManualControlClick(Sender: TObject);
    procedure ToolButtonAutoControlClick(Sender: TObject);
    procedure ToolButtonRunOptionsClick(Sender: TObject);
    procedure ToolButtonSwitchUsersClick(Sender: TObject);
    procedure ToolButtonUserManagerClick(Sender: TObject);
    // Status Bar
    procedure StatusBarDblClick(Sender: TObject);
    procedure   StatusBarMouseDown(Sender:   TObject;   Button:   TMouseButton;   Shift:
TShiftState; X, Y: Integer);
    // Tab Control Page Navigator
    procedure TabControlPageNavigatorMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
    procedure TabControlPageNavigatorChange(Sender: TObject);

    // Nuwe Stuff  MOET NOG GESKUIF WORD
    procedure PopupMenuPageNavigator_DeletePageClick(Sender: TObject);
    procedure ToolButtonCreateBackupClick(Sender: TObject);
    procedure ToolButtonSaveClick(Sender: TObject);
    procedure MainMenu_Options_PlatformThemeManagerClick(Sender: TObject);
    procedure PopupMenuMoveLinetoFrontClick(Sender: TObject);
    procedure PopupMenuSendLinetoBackClick(Sender: TObject);
    procedure toolBtnAlarmsClick(Sender: TObject);
    procedure toolBtnSMSClick(Sender: TObject);
    procedure ListViewAlarmsDblClick(Sender: TObject);
    procedure ListViewAlarmsResize(Sender: TObject);

  private
    PPreviousMousePosistion : TPoint;
    PPreviousSelectedIconIndex : Integer;

    PSelectedLineIndex : Integer;
    PPlatformMode : TPlatformMode;

    PImageBeginEndUpdateNestingCounter : Integer;
    PImageMustRefreshOnMouseUp : Boolean;

    PLayer1 : TBitmapLayer;

    Procedure InitialiseerAlleVeranderlikes;
    Procedure InitialiseerLayers;
        Procedure DrawGrid;

    Procedure ActivatePlatformOptionForm;
    Procedure ActivatePaltformThemeManagerForm;

    Procedure ScaleXY(Var AX , AY : Integer);
    Function  SnapToGrid(Var AX , AY : Integer) : Boolean;
    procedure SetPlatformMode(const Value: TPlatformMode);
        Function  PermisionToDEActivateEditMode : Boolean;
        Function  PermisionToActivateEditMode : Boolean;
        Function  PermisionToDEActivateIdleMode : Boolean;
        Function  PermisionToActivateIdleMode : Boolean;
        Function  PermisionToDEActivateManualMode : Boolean;
        Function  PermisionToActivateManualMode : Boolean;
        Function  PermisionToDEActivateAutoMode : Boolean;
        Function  PermisionToActivateAutoMode : Boolean;
    function ReadWorkspaceScale: Real;
    procedure WriteWorkspaceScale(const Value: Real);
```

```
  public


    FProject : TProject;
    FInfoPanel : TInformationPanelForm;

    Property FWorkspaceScale : Real Read ReadWorkspaceScale write WriteWorkspaceScale;
Public

    FWorkSpaceColor : TColor;
    FWorkSpaceFontSize : Integer;

    FWorkSpaceFontColor : TColor;
    FWorkSpaceFontBold : Boolean;
    FWorkSpaceFontBackgroundColor : TColor;
    FWorkSpaceFontBackgroundAlpha : Byte;

    FWorkSpaceShowGrid : Boolean;
    FWorkSpaceGridColor : TColor;
    FWorkSpaceGridSize : Byte;              // SAVED IN PROJECT FILE
    FWorkSpaceShowScrollBars : Boolean;
    FWorkSpaceSnapToGrid : Boolean;         // SAVED IN PROJECT FILE

    FWorkSpaceDrawIconsInHighContras : Boolean;

    // Hierdie hanteer die multiple pages van die projek
        // alle indexse werk vanaf 0 tot X
        FWorkSpaceCurrentPage : Byte;
        FWorkSpacePageNames : TStringList;

    FApplicationDescription : String100;    // SAVED IN PROJECT FILE
    FProjectFile : String250;


    // HIerdie is die Tipe van die Laaste DLL Icon wat gemaak is.
    PLastAutoCreatedDLLType : TDLLType;
    // Hierdie is die Items wat bykom onder die WorkspacePopup Menu on die
AutoCreateDLL Types Dinge
    PopupMenuItems : Array of TMenuItem;

    // HIerdie is die Procedure Event Notification wat die bogenoemde Array roep
    procedure AutoCreateADLLTypeEvent(Sender: TObject);
    // Hierdie is die REcreate Funksie wat geroep wird as jy insers druk
    Procedure ReCreateLastCreatedDLLTypeIcon;

    Property FPlatformMode : TPlatformMode Read PPlatformMode Write SetPlatformMode;

    Function  ApplicationEXEName : String;

    // Screen
    Procedure RefreshALL;
    Procedure RefreshWorkspace;
        Procedure DrawAllIcons;
        // Hooks wat die Icons gebruik om hul self te teken
        Function  ScreenBitmapBeginUpdate : Boolean;
        Function  ScreenBitmapEndUpdate : Boolean;
        Function  ScreenBitmap : TBitmap32;


    // Tabbed Control - Work Space Page Navigator
    Procedure UpdateTabControlPageNavigator;

    Procedure PageNavigatorSwithPage(AIndex : Byte);
    Procedure PageNavigatorAddPage();
    Procedure PageNavigatorDeletePage(AIndex : Byte);

    // List View ACTIONS and Alarms
    Function  AddActionListItem : TListItem;
    Procedure UpdateListViewActionsAlarmsHeight;

    Function  AddAlarmListItem : TListItem;
    Procedure UpdateListViewAlarmsAlarmsHeight;
```

```
    // Status Bar
    Procedure UpdateStatusBar;
    Procedure SetTimeOnStatusBar(ATime : Double; ATimeMode: TTimeMode);


    // Backup Spawn Goeters
    Function  MakePlatformBackupSpawn: Boolean;
    // Save Goeters
    Function  SavePlatform : Boolean;
    Function  LoadPlatform : Boolean;
        Procedure LoadPlatformVersion1(AFileStream : TFIleStream);
        Procedure LoadPlatformVersion2(AFileStream : TFIleStream);
        Procedure LoadPlatformVersion3(AFileStream : TFIleStream);

    Function  SaveProject(AFilepathName : String) : Boolean;
        Function  GetSaveprojectAs : String;
    Function  LoadProject(AFilepathName : String) : Boolean;
        Function  GetLoadProject : String;
        Procedure LoadProjectVersion1(AFileStream : TFIleStream);
        Procedure LoadProjectVersion2(AFileStream : TFIleStream);
        Procedure LoadProjectVersion3(AFileStream : TFIleStream);
end;
```

Code block 1 - Mainform interface

1.1.2. Project

The Project unit act as parent of the main functional units that give the Platform its distinct functionalities.

The units are the:

1. Platform Icon
2. Alarm Handler
3. Tag Handler
4. SMS Handler
5. User Manager

The Project unit handles the communication and data flow between these five unit instances and also between these five unit instances and the Mainform unit interface.

The Project unit interface is shown as follows:

```
unit Project;

interface

Uses
```

```
    // Herargie
    PlatformIcon,  {Array of}
    PlatformLines,
    PlatformActionsAlarms,

    AlarmHandler,
    TagHandler,
    SMSSender,
    UserManager,

    WhiteTimerUnit,
    // Eie ander tools
//    HVACMessenger,
    WhiteUnit,
    HVACIPlatformTypes,

    SimulationToolsFrm,

    GR32,
    GR32_Image,
    GR32_Layers,

    Dialogs,
    // Windows Tools Ek
    Math,
    StrUtils,
    Classes,
    SysUtils,
    Windows,
    Forms,
    // Windown Tools Automaties
    Types;

Type
    TProject = Class(TObject)
        Constructor Create(AOwner : TComponent; AParent : TObject); Reintroduce;
        Destructor Free; Reintroduce;
    Private
        POwner : TComponent;
        PParent : TObject;

//        PHVACMessengerServer: THVACMessengerServer;

        PIcons : Array of TPlatformIcon;
        PPlatformLines : TPlatformLines;
        PPlatformActionsAlarms : TPlatformActionAlarms;
        PAlarmHandler : TAlarmHandler;
        PTagHandler : TTagHandler;
        PUserManager : TUserManager;

        PTimerUnit : TWhiteTimerUnit;

        PRunDrawCounter : Integer;
        PRunRunCounter : Integer;

        PRunSimulatedTimeCounter : Double;
        PSimulationToolsForm : TSimulationToolsForm;

        Procedure TimeEverySecond();
                Function  GetNowTime : Double;
                Function  GetTimeTagTime : Double;
            Procedure PulseAllRunning(ANowTime: TDateTime);
            Procedure PulseAllDrawing;
            Procedure PulseMainFormUpdates;
            Procedure PulsePublishPlatformMode;

        Procedure InitialiseerVeranderlikes;
    Public
        FSMSSender : TSMSSender;

        FRunActivated : Boolean;
        FRunDrawInterval : Integer; {Every so many Seconds}
        FRunRunInterval : Integer; {every so many Seconds}
```

```
        FRunInitialMPlatformMode : TPlatformMode;
        FRunAutoControlEnabled : Boolean;

        FPublishPlatformModeTag : String; // Die Mode van die Platform word hierin
geskryf -1 vir Idle 0 vir Manual en 1 vir Auto COntrol

        FPlatformTimeMode : TTimeMode;

        FRunSimulatedStartDateTime : Double;
        FRunSimulatedSpeed : Integer; {Sec/Sec}
        FRunSimulationRunning : Boolean;      // hierdie word gebruik om die Simulation
te pause.
                                              // word NIE gesave nie. is altyd true as
projek laai.

        FRunTimeTagModeTag : String100;

        Property  FPlatformLines : TPlatformLines read PPlatformLines;
        Property          FPlatformActionsAlarms     :     TPlatformActionAlarms     read
PPlatformActionsAlarms;

        Property  FAlarmHandler : TAlarmHandler read PAlarmHandler;
        Property  FTagHandler : TTagHandler read PTagHandler;
        Property  FUserManager : TUserManager Read PUserManager;

        // Pulsing Simulation ETC.
        Procedure ResetSimulatedTime;
        Function  PublicGetNowTime : Double;

        Function  NumberOfIcons : Integer;
        function  CreateNewIcon(AX, AY: Integer; ACurrentPageIndex : Byte; ADLLType :
TDLLType = dtNone): Boolean;
            Function GetUniqueIconName : Integer;
            Function GetUniqueIconDescription(ADLLType : TDLLType = dtNone) : String;
        Function  DeleteIcon(AIndex : Integer) : Boolean;
        Procedure DeleteAllIcons();
        Function  FIconByIndex(Index : Integer) : TPlatformIcon;
        Function  FIconByName(AName : Integer) : TPlatformIcon;

        // Al die Teken Goed
        Procedure DrawAllIcons();

        Function   SendScreenPositionToAllIcons(AX,AY : Integer; ACurrentPageIndex :
Byte) : Integer;
        Function  SelectedIconIndex : Integer;
        Procedure MoveIcon(AIndex , AX , AY : Integer;ACurrentPageIndex : Byte);
        Function  FindIconsOutsideBoudaries(AX,AY, APageIndex : Integer) : Integer;

        Function  FindAvailableDLLs() : TDLLTypeSET;
        Function  FindUsedDLLs() : TDLLTypeSET;


//Was           Function   ActivateDLLBrowser(ADLLTypes: TDLLTypeSET; var AIconName :
Integer) : Pointer;
//MOetLyk Na       TCallBackFunctionDLLBrowser =                Function(ADLLTypeSET :
TDLLTypeSET;
//                                                            var ADLLRefs : Array of
TDLLPointerReference;
//                                                          AMultiSelectable : Boolean =
False;
//                                                            AAutoReturnAll : Boolean =
False) : Boolean of object;
        Function  ActivateDLLBrowser(ADLLTypes: TDLLTypeSET;
                                     var ADLLRefs : TDLLPointerReferenceArray;
                                     AMultiSelectable : Boolean = False;
                                     AAutoReturnAll : Boolean = False;
                                     ABrowserCaption : String = '') : Boolean;
            Function           FindAllPointersForDLLReferences(var      ADLLRefs      :
TDLLPointerReferenceArray) : Boolean;
            Function    FindAllPointersForDLLsInDLLTypeSet(ADLLTypeSet  :  TDLLTypeSET;
var ADLLRefs : TDLLPointerReferenceArray) : Boolean;


        Function PlatformMode : TPlatformMode;
```

```
        Procedure ActivateRunPulseOptionsForm;
        Procedure ActivateIconPlacementForm(ACurrentPageIndex : Byte);

        Procedure ActivateSimulationToolsForm;
        Procedure FreeSimulationToolsForm;

        Procedure SaveFile(AFileStream : TFileStream);
        Procedure LoadFile(AFileStream : TFileStream);
            Procedure LoadFileStructure1(AFileStream : TFileStream);
            Procedure LoadFileStructure2(AFileStream : TFileStream);
            Procedure LoadFileStructure3(AFileStream : TFileStream);
            Procedure LoadFileStructure4(AFileStream : TFileStream);
            Procedure LoadFileStructure5(AFileStream : TFileStream);
            Procedure LoadFileStructure6(AFileStream : TFileStream);
            Procedure LoadFileStructure7(AFileStream : TFileStream);
    end;
```

Code block 2 – Project interface

1.1.3. Platform Icon

The Platform Icon acts as a user interface icon on the Platform that the user interacts with. The next image shows the Platform with Platform Icons created on the interface. These Platform Icons are the pumps and the dams. An instance of the Platform Icon is created for each of these items.

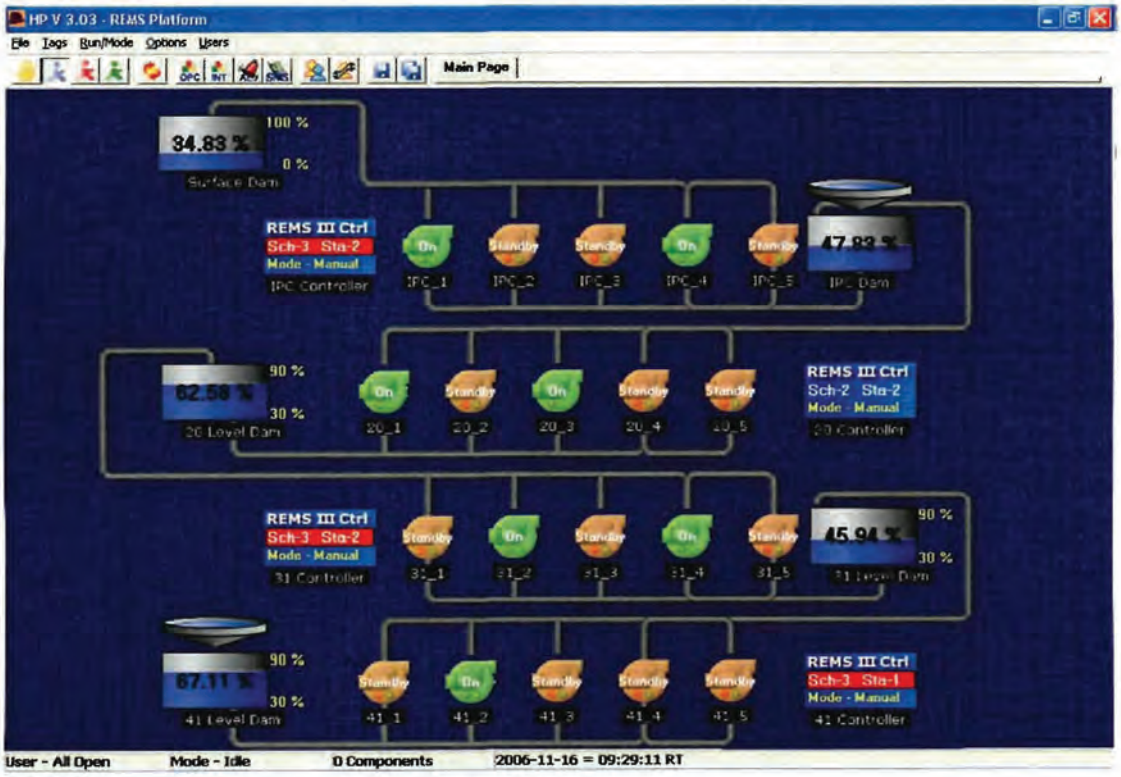


Figure 1-2 Platform with Icon

The Platform Icon unit handles the functionality regarding each icon. The unit also dictates the interaction between the different icons. The implementation of the Platform Icon unit is shows as follows:

```
unit PlatformIcon;

interface

uses
    // Herargie
    IconDLLInterface,
    // Eie ander tools
    WhiteUnit,
    WhiteIntegerList,
    HVACIPlatformTypes,

    GR32,
    GR32_Image,
    GR32_Layers,
    // Windows Tools Ek
    Math,
    Classes,
    Graphics,
    SysUtils,
    Dialogs,
    Forms,
    // Windown Tools Automaties
    Windows;


Type
    TIconPlacement = record
            PageIndex : Byte;
            Position : TPoint;
            StretchFactor : Real;
        end;

Type
    TPlatformIcon = Class(TObject)
        Constructor Create(AOwner : TComponent; AParent : TObject; AName : Integer);
Reintroduce;
        Destructor Free; Reintroduce;
    Private
        POwner : TComponent;
        PParent : TObject;

        PIconDLL : TIconDLLInterface;

        // Set Properties
        PPlacements : Array of TIconPlacement;

        PSize : TPoint;
        PDescription : String50;
        PDescriptionShow : Boolean;
        PName : Integer;

        PImagePathName : String150;

        PPicture : Graphics.TBitmap;
        // GUI Changing Properties
        PSelected : Boolean;

        PTimeDLLRequestedRefresh : Double;
        PDLLRefreshPending : Boolean;

        PDEBUGDRAWCOUNTER : Integer;

        Procedure InitialiseerVeranderlikes;
```

```
        Function  GetPlacementIndex(APageIndex : Byte) : Integer;


        Procedure DrawClearPreviousPosition();
        Procedure DrawMarkTriangle();
        Procedure DrawBitmapImage();
        Procedure DrawDescription();
        Procedure DrawDebugDrawCounter();

        // All Set and Get Procedures
        procedure SetPSelected(const Value: Boolean);

        function  GetFPositionX(APageIndex: Byte): Integer;
        procedure SetFPositionX(APageIndex: Byte; const Value: Integer);
        function  GetFPositionY(APageIndex: Byte): Integer;
        procedure SetFPositionY(APageIndex: Byte; const Value: Integer);

        function  GetFPosition(APageIndex: Byte): TPoint;

        function  GetFStretchFactor(APageIndex: Byte): Real;
        procedure SetFStretchFactor(APageIndex: Byte; const Value: Real);

    Public
        Property  FPositionX[APageIndex : Byte] : Integer read GetFPositionX write
SetFPositionX;
        Property  FPositionY[APageIndex : Byte] : Integer read GetFPositionY write
SetFPositionY;
        Property FPosition[APageIndex : Byte] : TPoint read GetFPosition;
        Property FStretchFactor[APageIndex : Byte] : Real read GetFStretchFactor write
SetFStretchFactor;
        Property FSizeX : Integer read PSize.X write PSize.X;
        Property FSizeY : Integer read PSize.Y write PSize.Y;
        Property FDescription : String50 read PDescription write PDescription;
        Property   FDescriptionShow   :   Boolean   read   PDescriptionShow   write
PDescriptionShow;
        Property FName : Integer Read PName;
        Property FImagePathName : String150 read PImagePathName write PImagePathName;

        Property FSelected : Boolean read PSelected write SetPSelected;

        Function IsSelected(AX,AY : Integer; ACurrentPageIndex : Byte) : Boolean;

        Function  NumberOfPlacements : Byte;
        Procedure  AddPlacment(APageIndex  :  Byte  ;  APositionX,APositionY  :  Integer;
AStretchFactor : Real = 1);
        Procedure DeletePlacement(APageIndex : Byte);
        Function  IsVisiableOnThisPage(APageIndex : Byte) : Boolean;

        Procedure LoadDLL(ADLLType : TDLLType);
        Function  GetDLLType : TDLLType;
        Function  GETDLLPASPointer : Pointer;

        Procedure Run(ANowTime: TDateTime);

        Procedure Draw(AClearPosition : Boolean = False; ADraw : Boolean = True);
            //Link to the main Page That is used to draw the Icon
            Function  WorkSpaceCurrentPage : Integer;
            Function  WorkSpaceBitmapBeginUpdate : Boolean;
            Function  WorkspaceBitmap : TBitmap32;
            Function  WorkSpaceBitmapEndUpdate : Boolean;

            // This tool is written to bridge from GR32 to TBitmap
            Procedure  CarryBitmapToBitmap32(ABitmap  :  Graphics.TBitmap  ;ABitmap32  :
TBitmap32; APasteCentre : TPoint; AStretchFactor : Real);

        Procedure ActivatePlatformIconEditForm;
        Procedure ActivateDLLEditForm;
        Procedure ActivateDLLViewForm;

        Function  LoadTPictureFromImageFile : Boolean;

        Procedure SaveFile(AFileStream : TFileStream);
        Procedure LoadFile(AFileStream : TFileStream);
```

```
            Procedure LoadFileStructure1(AFileStream : TFileStream);
            Procedure LoadFileStructure2(AFileStream : TFileStream);
            Procedure LoadFileStructure3(AFileStream : TFileStream);

        // UPWARDS COMMUNICATION
        // All Tools (Linked to Main App Functions) used by DLL for Upwards
Communicatoin
        Function  GetTagValue(ATag : String) : Double;
        Procedure SetTagValue(AString : String; AValue : Double);
        Function  IconDescription(AString : String) : String;
        Function  TagBrowser : String;
        Function  DLLBrowser(ADLLTypeSET : TDLLTypeSET;
                             var ADLLRefs : TDLLPointerReferenceArray;
                             AMultiSelectable : Boolean = False;
                             AAutoReturnAll : Boolean = False;
                             ABrowserCaption : String = '') : Boolean;

        Function  RefreshIcon() : Boolean;
        Function  PlatformAction(AActionCode, AI1, AI2, AI3 : Integer; AS1, AS2, AS3 :
String) : Boolean;

        Function  OPCConnected : Boolean;
        Function  PlatformMode : TPlatformMode;
        Function  UserPrivilege : TUserPrivilege;
        Function  UserName : String;
        Function  EditInternalTag(ATagname : String; AActivateEditor : Boolean = True)
: Boolean;

        Procedure RaiseAction(AActionDescription, AActionExplenation : String);
        Procedure     RaiseAlarm(AAlarmPriority     :     Byte;     AAlarmDescription,
AAlarmExplenation : String);
    end;
```

Code block 3 - Platform Icon

1.1.4. Alarm Handler

The Alarm Handler unit is responsible for all the alarm functionality in REMS. This unit enables the user to create or set alarms to be raised in any defined condition. The Alarm Handler will then raise these alarms in the preset conditions.

An example of this is when the user creates a alarm to be sounded when a certain dam level is above a given limit. After the alarm condition is defined the Alarm Handler unit will keep track of the given dam and raise alarm when the level is above the given limit.

The Alarm Handler unit provide the functionality to raise the alarms in any combination of the following formats:

1. Visual. A visual message is shown on the Mainform interface.

2. Audio. An audio sound is played.

3. E-Mail. An E-Mail is send to specified E-Mail addresses with an explanation of the alarm that is raised.

4. SMS. A SMS is send to a specified Cellphone number.

All the alarms that is raised is also logged for further reference. The following shows the implementation of the Alarm Handler:

```
unit AlarmHandler;

interface


Uses
    // Herargie
    Alarm,
    TagHandler,
    SMSSender,
    HVACIPlatformTypes,
    // Other Aplication Tools
    MSScriptControl_TLB,
    //WIndows Tools Self
    // Windows Tools Automatically
    SysUtils,
    Dialogs,
    Classes;

Type
    TAlarmHandler = Class(TObject)
      Constructor Create(AOwner:TComponent; AParent:TObject); Reintroduce;
      Destructor  Free; Reintroduce;
    private
      POwner : TComponent;
      PParent : TObject;

      PHandlerFormIsOpen : boolean;

      Procedure InitialiseerVeranderlikes;
    public
      FEnableAllAlarms : boolean;
      FAcceptTag : String250;
      FAlarms : Array of TAlarm;

      Procedure CopyAlarm(AIndex : Integer; ALaunchEditFrm : Boolean);

      Function  GetAllAlarms : TStringList;

      Function  NumberOfAlarms : Integer;
      Procedure AddAlarm(ALaunchEditFrm : Boolean);
      Function  AddAlarmTagStrings(AAlarmName,AAlarmTagString:String):Boolean;
      Procedure DeleteAlarm(AIndex : Integer);
      Procedure DeleteAllAlarms();
        Procedure ValidateAlarmDescription(var ADescription : String);

      Function  IsThisAnAlarm(AName : String): Boolean;
      Function  Alarm(AName : String): TAlarm;

      Procedure Run;

      Function SaveFile(AFileStream:TFileStream):Boolean;
      Function LoadFile(AFileStream:TFileStream):Boolean;
        Function LoadFile1(AFileStream:TFileStream):Boolean;
```

```
        Function LoadFile2(AFileStream:TFileStream):Boolean;

      // Communication Down to Main Programm
      Procedure ActivateAlarmManager;
      Function     ToggleDirectAlarmEditor(AIntTagName   :   String;   AActivateEditor   :
Boolean = True) : Boolean;

      Function SMSSender : TSMSSender;
      Function ActivateGlobalTagBrowser : String;
      Function GetGlobalTagValue(ATag : String) : Double;
      Procedure SetDoubleTagValue(ATag : String; AValue : Double);
      Procedure SetStringTagValue(ATag : String; AValue : String);
      Function  Time : Double;
      Function OPCConnected : boolean;
      Function PlatformMode : TPlatformMode;

      Procedure RaiseAction(AActionDescription, AActionExplenation : String);
      Procedure RaiseAlarm(AAlarmDescription, AAlarmExplenation : String);
    end;
```

Code block 4 - Alarm Handler interface

1.1.5. Tag Handler

The Tag Handler is the unit responsible for communication to SCADA packages via OPC. This unit incorporates OPC API (Application Protocol Interface) that enables the REMS to establish an OPC Connection and retrieve OPC Tags and OPC Tag Values from a OPC enabled SCADA.

This OPC connection is used by the REMS to retrieve information regarding the project setup such as dam levels and pump statuses. OPC tags are also used to control components such as pumps and valves. The following shows the Tag Handler interface:]

```
unit TagHandler;

interface

Uses
    // Herargie
    InternalTagHandler,
    OPCConnection,
    // Own Tools
    HVACIPlatformTypes,
    // Windows Tools EK
    Dialogs,
    SysUtils,           //Now

    Classes;
    // Windows Tools Automatically Inserted
Type
    TGetNowTimeFunctionAddress = Function() : Double of Object;

Type
    TTaghandler = Class(TObject)
        Constructor    Create(AOwner    :    TComponent;    AParent    :    TObject;
AGetNowTimeFunctionAddress : TGetNowTimeFunctionAddress);
        Destructor  Free; Reintroduce;
    Private
```

```
        PParent : TObject;
        POwner : TComponent;

        POPCConnection : TOPCConnection;
        PInternalTagHandler : TInternalTagHandler;

        Procedure InitialiseerVeranderlikes;
    Public
        FGetNowTimeFunctionAddress : TGetNowTimeFunctionAddress;

        Property FInternalTagHandler : TInternalTagHandler read PInternalTagHandler;
        Property FOPCConnection : TOPCConnection Read POPCConnection;

        Function  GetTagValueBinary(ATagName : String) : Byte;
        Function  GetTagValueByte(ATagName : String) : Byte;
        Function  GetTagValueDouble(ATagName : String) : Double;
            Function IsThisAnInternalTag(ATag : String) : Boolean;

        Procedure SetTagValueDouble(AString : String50; AValue : Double);
        Procedure SetTagValueString(AString : String50; AValue : String);

        Function ActivateTagBrowser : String;

        Function GetNowTime : Double;

        Procedure SaveFile(AFileStream : TFileStream);
        Procedure LoadFile(AFileStream : TFileStream);
            Procedure LoadFileStructure1(AFileStream : TFileStream);
    end;

interface
Uses
        dOPC,
    dOPCIntF,
    dOPCCom,
    dOPCComn,

    TagBrowsingThread,

    SyncObjs,

        Dialogs,        //For ShowMessage
        SysUtils, //For Inttostr
    Forms,

    WhiteUnit,

        Classes;

Type
    TTagDataType = (ttBinary,ttByte,ttDouble,ttUnknown);
    String200 = String[200];
    TOnServerAutoRestartEvent = Procedure() of Object;
    TOnWriteError = Procedure (AErrorMessage : String) of Object;
    TActivateTagBrowerFunctionAddress = Function() : String of Object;

Type
    TOPCConnection = Class(TObject)
        Constructor              Create(AOwner:TComponent;              AParent:TObject;
AActivateTagBrowserFunctionAddress : TActivateTagBrowerFunctionAddress); Reintroduce;
        Destructor Destroy; Reintroduce;
    Private
        POwner : TComponent;
        PParent        : TObject;

        PActivateTagBrowserFunctionAddress : TActivateTagBrowerFunctionAddress;

        PAllTagsStringList : TStringList;

        PHostName      : String200;
        PServerName    : String200;

        PAutomaticStartUp : Boolean;
        PAutoConnectWait : integer;
```

```
        PFilterInOnlySelectedTags : Boolean;              // This will make sure only
tags that confides to our PFIlterInOnlySelectedTagKeys will be inserted into the Tag
Browser
            PFilterInOnlySelectedTagKeys : TStringList;
        PLoadAvailableTagsWithConnection : Boolean;        // If this is true the
available tags will be browser just after connection to the OPC has been made
                                                           // If False the Browsing of
tags will be done in a sub-thread just after connection has been made
        PFilterPath : Boolean;
        PUseHierarchicalBrowsing : Boolean;


        PAutoamtedTabBrowserThread : TOPCTagBrowsingThread;

        OPCServer      : TdOPCServer;
        OPCItems : TdOPCItems;

        PConnectedToServer : Boolean;
        PTimeToMakeOPCConnection : Double;
        PNumberOfTagsFilteredOut : Integer;

            PServerCheckingTag : String200; {Kyk na die tag, as tag nie verander nie
neem die komponent aan die OPC server hang}
            PServerCheckingActive : Boolean; { Die is natuurlik om die server chekking
te dis en enable.}
            PServerCheckingInterval : Integer; {Sekondes REMS sal elke soveel sekondes
kyk of die Checking tag verander het}
            PServerCheckingLastCheckTime : Double;
            PServerCheckingLastCheckValue : Double;
            PServerCheckingLastCheckValueTime : Double;

//White
        Function OPCItem(ATag : String) : TdOPCItem;

            Procedure CheckServer;

        Function  RefreshOPCServer    : Boolean;
            Function      BrowseAndListAvailableOPCTags(AShowMessages    :    Boolean)    :
Integer; // Returns Number of tags Found
            Procedure BrowseAndListAvailableOPCTagsINSIDETHREAD;

                Function  DoesThisTagPassTagFilterCriteria(ATag : String) : Boolean;
                Procedure RecursiveAdd(Browser: TdOPCBrowser; ItemList: TStrings);
                Function  FilterOPCItemID(AItemID : String) : String;

        Procedure PrivateOnServerShutDown(Sender: TObject; Reason: String);
        Procedure PrivateOnServerWriteError(Sender: TObject; ItemList: TdOPCItemList);
        Procedure  PrivateOnServerError(Sender  :  TObject;  Errorstring:  string;  var
RaiseException : boolean);

        // All the Gets and Sets
        procedure SetServerCheckingTag(const Value: String200);
        procedure SetServerCheckingActive(const Value: Boolean);
        procedure SetServerCheckingInterval(const Value: Integer);

        Procedure InitialiseerVeranderdelikes;
            Procedure InitialiseerOPCServerAndOPCItems;
            PRocedure FreeOPCServerAndOPCItems;
    Public
        OnServerAutoRestart : TOnServerAutoRestartEvent;
        OnWriteError : TOnWriteError;

        // Connection Settings
        Property FHostName : String200 Read PHostName write PHostName;
        Property FServerName : String200 Read PServerName write PServerName;
        Property   FAutomaticStartUp   :   Boolean   Read   PAutomaticStartUp   Write
PAutomaticStartUp;
        Property   FAutoConnectWait   :   integer   Read   PAutoConnectWait   Write
PAutoConnectWait;
        Property      FLoadAvailableTagsWithConnection      :      Boolean      Read
PLoadAvailableTagsWithConnection Write PLoadAvailableTagsWithConnection;
        Property FFilterPath : Boolean Read PFilterPath Write PFilterPath;
        Property  FUseHierarchicalBrowsing  :  Boolean  Read  PUseHierarchicalBrowsing
Write PUseHierarchicalBrowsing;
```

```
        //
        Property FFilterInOnlySelectedTags : Boolean Read PFilterInOnlySelectedTags
Write PFilterInOnlySelectedTags;
        PRoperty      FFIlterInOnlySelectedTagKeys      :      TStringList      Read
PFIlterInOnlySelectedTagKeys Write PFIlterInOnlySelectedTagKeys;

        // Server Checking Settings
        Property    FServerCheckingTag  :  String200  read  PServerCheckingTag  write
SetServerCheckingTag;
        Property   FServerCheckingActive : Boolean Read PServerCheckingActive write
SetServerCheckingActive;
        Property        FServerCheckingInterval    :    Integer    {Seconds}    Read
PServerCheckingInterval Write SetServerCheckingInterval;

        // Only for Debuggin

            // Function Used By The External Tag Browsing Function;
            Function   PublicBrowseAndListAvailableOPCTags(AShowMesages : Boolean) :
Integer; // Returns Number of tags Found
            Procedure PublicRecursiveAdd(Browser: TdOPCBrowser; ItemList: TStrings);
            Function  PublicFilterOPCItemID(AItemID : String) : String;
            Function  PublicDoesThisTagPassTagFilterCriteria(ATag : String) : Boolean;
            Procedure GiveNewTags(ATags : TStringList);

        Procedure ActivateOPCConnectionOptionsForm;
        Function  ActivateTagBrowser : String;

        Function  FindAllOPCServers : TStringList;

        Function  ResetServer(AHostName, AServerName : String): Boolean;Overload;
        Function  ResetServer():Boolean;Overload;
        Function  IsClientOnline : Boolean;
        Property  TimeToMakeOPCConnection : Double Read PTimeToMakeOPCConnection;

        Procedure DisconnectOPCServer;

        Function  GetNumberOfTags : Integer;  // Dit is al die tags wat beskikbaar is
        Function  NumberOfUsedTags : Integer; // Dit is net die tag wat gebruik word
        Function  NumberOfFilteredOutTags : Integer;
        Function  GetAllTags : TStringList;

        Function  DoesThisTagExist(ATag : String) : Boolean;
        Function  GetTagTypeString(ATag : String) : String;
        Function  GetTagValueString(ATag : String) : String;

        Function  GetBinaryTagValue(ATag : String) : Byte;
        Function  GetByteTagValue(ATag : String) : Byte;
        Function  GetDoubleTagValue(ATag : String) : Double;

        Function  SetBinaryTagValue(ATag : String; AValue : Byte) : Boolean;
        Function  SetByteTagValue  (ATag : String; AValue : Byte) : Boolean;
        Function  SetDoubleTagValue(ATag : String; AValue : Double) : Boolean;
        Function  SetStringTagValue(ATag : String; AValue : String) : Boolean;

        Procedure SaveAllTagNamesToCSV(AFileName : String);
        Procedure LoadTagsFromCSV(AFileName : String);

        Procedure SaveFile(AFileStream : TFileStream);
        Procedure LoadFile(AFileStream : TFileStream);
            Procedure LoadFileStructure1(AFileStream : TFileStream);
            Procedure LoadFileStructure2(AFileStream : TFileStream);
            Procedure LoadFileStructure3(AFileStream : TFileStream);
            Procedure LoadFileStructure4(AFileStream : TFileStream);
            Procedure LoadFileStructure5(AFileStream : TFileStream);
            Procedure LoadFileStructure6(AFileStream : TFileStream);
    end;
```

Code block 5 - Tag Handler interface

1.1.6. SMS Sender

The SMS Sender unit enables the REMS to send SMS's via the South African Cellular Telephone Network. This is used when alarms are raised to inform chosen people in given situations via an SMS.

The following shows the SMS Sender interface:

```
unit SMSSender;

interface

Uses
    Classes,
    WhiteUnit,
    HVACIPlatformTypes,
    SMSSendingThread,
    Dialogs,
    SysUtils,
    ExtCtrls,
    SMS_COMAPILib_TLB;

Type
    TContact = record
        Name : String50;
        Number : String50;
    end;

Type
    TSMSSender = Class(TObject)
        Constructor Create(AOwner:TComponent; AParent:TObject); reintroduce;
        Destructor Free; reintroduce;
    Private
        POwner : TComponent;
        PParent : TObject;

        PTimer: TTimer;

        PLastSMSSendingTime : Double;

        PSMSQ : Array of TSMSQEntry;

        Procedure InitieerVeranderlikes;

        Procedure TimerFunction(Sender: TObject);
            Procedure CheckSMSQ;

        Function GetTime : double;

        Procedure  LaunchSMSSendingThread(ASMSBody:WideString;  ARecepeintsNumbers  :
Array of WideString;ASMSQIndex : Integer);
    Public
        FSMSNotificationEnabled : Boolean;

        FDirectSMS : boolean;
          FSCNumber : string50;
          FValidityPeriod : string50;
          FCOMPort : string50;
          FBaudRate : string50;

        FClickatellSMS : boolean;
          FMineName : String50;
          FSendingMedium : TSendMedium;
          FMWebConnectionNumber : String50;
          FInternetProxy : String50;
```

```
        FINternetPort : String50;

      FContacts : Array of TContact;

      // SMS Q Tools
      Procedure SMSQAddEntry(ARecepientNames : TStringList; ABody : WideString);
          Function GetNumberForContacName(AName : String) : WideString;
      Function NumberOfSMSQ : Integer;
      Procedure EmptySMSQ;

      // Contact Tools
      Function  NumberOfContacts : Byte;

      Procedure AddContact(AName : String; ANumber : WideString);
      Procedure DeleteContact(AIndex : Integer);

      {User Public}
      Procedure DELETESMSFIRSINQ;

      Procedure ActivateSMSOptionsForm;
      {Save Stuff}

      Procedure SaveSMSSender(AFileStream: TFileStream);
      Procedure LoadSMSSender(AFileStream: TFileStream);
          Procedure LoadSMSSender1(AFileStream: TFileStream);
          Procedure LoadSMSSender2(AFileStream: TFileStream);
  end;
```

Code block 6 - SMS Sender

1.1.7. User Manager

The User manager Unit dictates which REMS user is permitted to perform which actions within the Platform. The system administrator uses this unit to grant abilities and functionalities to each user that is permitted to perform actions on REMS.

Before any user can perform any task within REMS, he must first log on with his given Username and Password. The user will the be restricted to functionalities as dictated by the REMS administrator.

The following shows the User Manager interface:

```
unit UserManager;

interface

Uses
    // Herargie
    // Other Tools
    HVACIPlatformTypes,
    // Windows Tools
    SysUtils,
    Dialogs,
    Classes;

Type
    TUserManager = Class(TObject)
            Constructor Create(AOwner : TComponent ; AParent : TObject) ; Reintroduce;
```

```
            Destructor  Free; Reintroduce;
        Private
            POwner : TComponent;
            PParent : TObject;

            PLastActivityTime : Double; // Hou die yd wanner daar laas gewerk is /
word nie in die file save structure gesave nie.

            PCurrentUserIndex : Integer;
            PUsers : Array of TUserProfile;

            Procedure InitialiseerVeranderlikes;
        Public
            FAutoLogoffEnabled : Boolean; // Maak dat as daar nie aktiviteit op die
Main Form is nie die User automaties aflog
            FAutoLogoffTime {Minutes} : Integer; // Minutes   --   Sien ^

            Function  NumberOfUsers : Integer;

            Function       TestAccessPrivilege(APrivilegeNeeded    :    TUserPrivilege;
AShowMessage : Boolean = True): Boolean;

            Function  AttempLogIn(AName,APassword : String) : Boolean;

            Function  GetCurrentUserPrivilege : TUserPrivilege;
            Function  GetCurrentUserName : String50;

            Function  GetUserProfile(AIndex : Integer) : TUserProfile;

            Procedure Run(ANowTime : Double);
            Procedure MouseActivity(ANowTime : Double);

            // Word duer die User Manager FORM Gebruik
                Procedure    AddUser(AName,APassword    :    String    ;    APrvilege    :
TUserPrivilege);
                Procedure DeleteUser(AIndex : Integer);

            Procedure ActivateUserManager;
            Procedure ActivateUserLogInForm;

            Procedure SaveFile(AFileStream : TFileStream);
            Procedure LoadFile(AFileStream : TFileStream);
                Procedure LoadFileStructure1(AFileStream : TFileStream);
                Procedure LoadFileStructure2(AFileStream : TFileStream);
        End;
```

Code block 7 - User Manager

1.2. Pump DLL

The Pump DLL (Dynamically Linked Library) is the code section that represents the actual pump on the mine. This part of the code is used to 'talk' or connect to the physical pumps. In doing this the Pump DLL code represent the actual physical pump in REMS.

An instance of the Pump DLL is created for each pump on the interface. The following figure shows a platform setup with several instances of the Pump DLL.

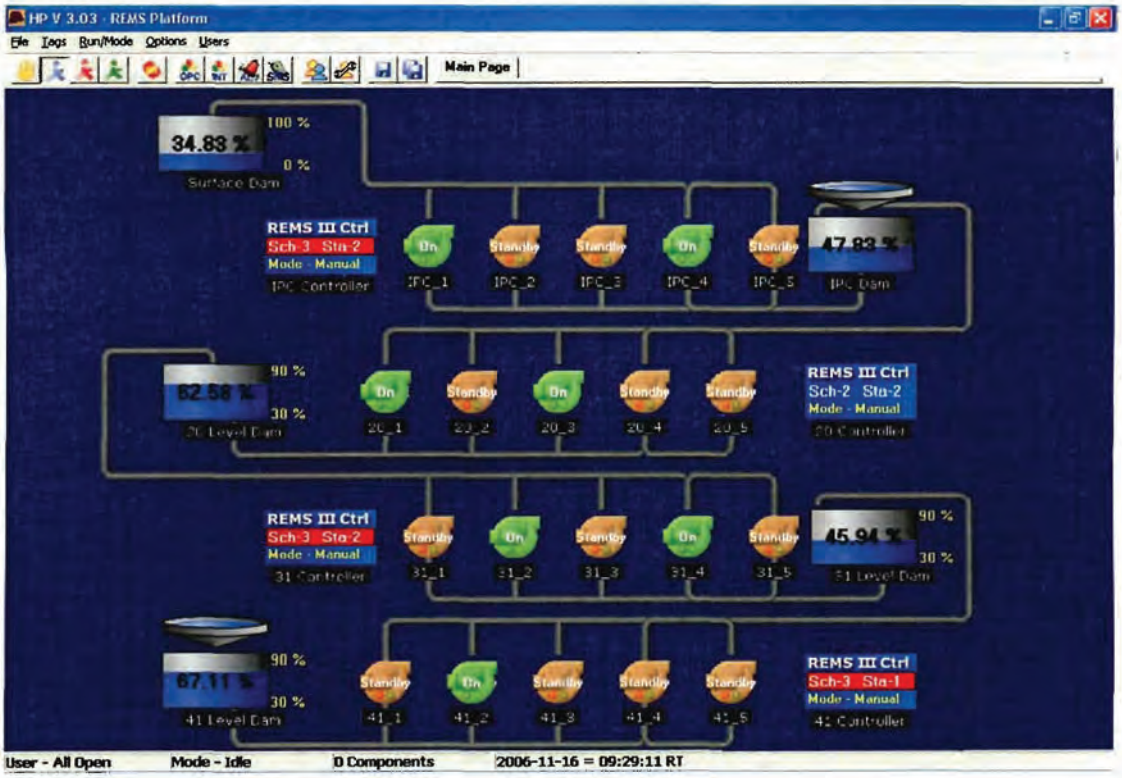


Figure 1-3 Pump DLL used to represent actual pumps

The Pump DLL consists of the following units as show in the following figure and thereafter discussed individually.

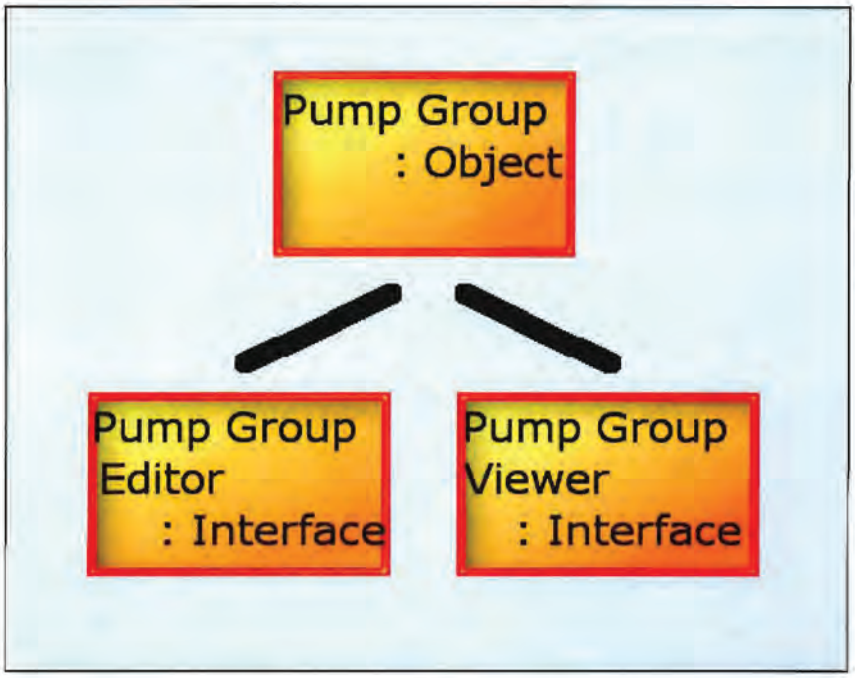


Figure 1-4 Unit structure of the Pump DLL

1.2.1. Pump

The Pump Unit contains the code that encapsulates all the functionality and user settings associated with each pump. The Pump Unit also acts as the connection interface between the Pump DLL and the Platform.

The following show the interface of the Pump Unit:

```
unit Pump;

interface

Uses
    // Afgelei Van
    PumpInterface,

    PumpPanel,
    // Hierargie
    // Ander Tools
    //PlatformIcon,  // This is used for Upward Communication.   Function in Platform
Icon enables the DLL to usitlise functions in the Main app
    HVACIPlatformTypes,
    // Windows Tools Self
    Forms,
    Dialogs,
    Controls,       // mrOk
//    Windows,        // GetRValue, GetGValue, GetBValue
    Graphics,       // TCanvas
    Types,          // TPoint
    SysUtils,       // ExtractFilePath
    ExtCtrls,       // TTimer
    Classes;
```

```
    // Windows Tools Automaties;


Type
  TPumpControlOld = (pcBitOld, pcByteOld);
  TPumpControl = (pcBit, pcByte, pcBitSpec, pc4, pc5, pc6, pc7, pc8, pc9, pc10);

Type
    TPump = Class(TPumpInterface)
        Constructor Create(AOwner: TComponent; AParent: TObject); Reintroduce;
        Destructor  Free; Reintroduce;
    Private
      POwner : TComponent;
      PParent : TObject;

      PPlatformMode : TPlatformMode;

      PSimStatus : boolean; // Running status during simulation
      PSimTemprature : Real;

      PTimeMode : TTimeMode; // Are we simulating?
      PNowTime : TDateTime;

      PPumpGraphic : TBitmap;

      PDisplayedPumpStatus : String;
      PDisplayedStartPermission : Boolean;
      PDisplayedStopPermission : Boolean;
{HB}  PDisplayedNoStopMinRuntime : Boolean;

      // Hierdie Is altyd By Default True, Maar 'n Panel DLL kan die waarder hiervan
verander as die pomp nie mag start nie.
      PPanelPermittedToStart : Boolean;

      PTempUnavailable : boolean;
      PPumpStartedOnce : boolean;
      PPumpStoppedOnce : boolean;
      PTempUnavailableTime : double;

      PTimeStarted : TDateTime;
      PTimeStopped : TDateTime;
      PHoldDelaySec : Integer;

      PViewForm : TForm;
      PFreeViewForm : Boolean;

      PGetTagValueFunctionAddress : TCallbackFunctionGetTagValue;
      PSetTagValueFunctionAddress : TCallbackFunctionSetTagValue;
      PIconDescriptionFunctionAddress : TCallbackFunctionIconDescription;
      PTagBrowserFunctionAddress : TCallBackFunctionTagBrowser;
      PDLLBrowserFunctionAddress : TCallBackFunctionDLLBrowser;

      PRefreshIconFunctionAddress : TCallBackFunctionRefreshIcon;
      PPlatformActionFunctionAddress : TCallBackFunctionPlatformAction;

      POPCConnectedFunctionAddress : TCallBackFunctionOPCConnected;
      PPlatformModeFunctionAddress : TCallBackFunctionPlatformMode;
      PUserPrivilegeFunctionAddress : TCallBackFunctionUserPrivilege;
      PUserNameFunctionAddress : TCallBackFunctionUserName;
      PEditInternalTagFunctionAddress : TCallBackFunctionEditInternalTag;

      PRaiseActionFunctionAddress : TCallBackFunctionRaiseAction;
      PRaiseAlarmFunctionAddress : TCallBackFunctionRaiseAlarm;

      PHoldTimerFlag : Boolean;
      PFlagStatusOnCommandRecieve : Boolean; // The Status of the pump on recieval of
a commnad is saved here
      PRepeatedStart : Boolean;
      PRepeatStartTimerFlag : Boolean;
      PRepeatedStop : Boolean;
      PRepeatStopTimerFlag : Boolean;

      PLastAddTime : double;
```

```
      PDrawIconsInHighContras : Boolean;
      PDrawIconsInHighContrasTimeFlagUpdate : double;
//      Timer : TTimer;
//      Procedure TimerFunction(Sender: TObject);

      Function InvertBits(AValue : Integer) : Integer;

      Function IsSimulating : Boolean;

      Procedure InitialiseerVeranderlikes;
      //Procedure Draw(APosition : TPoint ; ACanvas : TCanvas);
          Procedure DrawGraphic(var ABitmap : TBitmap);
          Procedure DrawDescription(var ABitmap: TBitmap);
          Procedure DrawStartPermission(var ABitmap: TBitmap);

      Procedure LoadPumpGraphic;
      Procedure ViewFormClose(Sender: TObject; var Action: TCloseAction);


    Public
      // Physical Properties
      FDescription : String;

      // Tags
      FStartTag : String[100];
      FStopTag : String[100];
      FAvailabilityTag : String[100];
      FStatusTag : String[100];
      FPriorityTag : String[100];
      FStartPermissionTag : String[100];
      FStopPermissionTag : String[100];
      FAutoManualTag : String[100];

      // Control Variables
      FResetOnStatusChange : boolean;
      FAttemptRestart : Boolean;
      FStartStopInManual : Boolean;
      FStartStopInAuto : Boolean;
      FRepeatStart : Boolean;
      FRepeatStop : Boolean;
      FRepeatStartTime : Double;
      FRepeatStopTime : Double;

      FStartType : TPumpControl;
      FStartValue : integer;
      FStartSpecBit : integer;
      FStartSpecBitGrey : integer;
      FStopType : TPumpControl;
      FStopValue : integer;
      FStopSpecBit : integer;
      FStopSpecBitGrey : integer;
      FAvailabilityType : TPumpControl;
      FAvailabilityValue : integer;
      FAvailabilitySpecBit : integer;
      FAvailabilitySpecBitGrey : integer;
      FStatusType : TPumpControl;
      FStatusValue : integer;
      FStatusSpecBit : integer;
      FStatusSpecBitGrey : integer;

      // Display Variables
      FColourStandby : TColor;
      FColourOffline : TColor;
      FColourOn : TColor;

      // Properties
      FPower : integer;
      FFlowRate : double;
      FEfficiency : integer;
      FType : String[50];
      FMinRunTime : double;
      FHoldDelay : integer;
      FLocked : boolean;
```

```
      FStartingPriority : integer;
      FStartingSet : Integer;

      //Form Properties
      FXPos : integer;
      FYPos : integer;

      FRunTime : double;

      Function  GetStatus : Boolean; override;
      Function  GetAvailability : Boolean; override;
      Function  GetAvailabilityNotLocked : Boolean;
      Function  GetStartPermission : Boolean; override;
      Function  GetStopPermission : Boolean; override;
      Function  GetLockedOut : Boolean; Override;
      Function  GetPower : Double; OverRide;                    {kW}
      Function  GetFlowRate : Double; override;
      Function  GetMinRunTime : Double; override;
      Function  GetStartedTime : Double; override;
      Function  GetStoppedTime : Double; override;

      Function  GetStatusString : String; override;

      Function  GetSimulationTemperature: Real; override;
      Procedure SetSimulationTemperature(const Value: Real); override;

      Function  GetPanelPermittedToStart : Boolean; Override;
      Procedure SetPanelPermittedToStart(AValue : Boolean); Override;

      Function  GetAttemptRestartOnStartFailure : Boolean; override;

      Function  GetStartingPriority : Integer; override;
      Procedure SetStartingPriority(AValue : Integer); override;
      Function  GetStartingSet : Integer; override;
      Procedure SetStartingSet(AValue : Integer); override;

      Procedure Start(AHoldDelay:Integer); override;
      Procedure Stop(AHoldDelay:Integer); override;
        Procedure TogglePumpPanelsRun;

      Function OtherPump(APointer : Pointer) : TPump;

      Function MixBytes(FG, BG, TRANS: byte): byte;
      Procedure MaskBitmap(ABitmap : TBitmap; AColor : TColor; AMask : integer);

      Property   FSetTagValueFunctionAddress   :   TCallbackFunctionSetTagValue   read
PSetTagValueFunctionAddress write PSetTagValueFunctionAddress;
      Property   FGetTagValueFunctionAddress   :   TCallbackFunctionGetTagValue   read
PGetTagValueFunctionAddress write PGetTagValueFunctionAddress;
      Property FIconDescriptionFunctionAddress : TCallbackFunctionIconDescription read
PIconDescriptionFunctionAddress write PIconDescriptionFunctionAddress;
      Property   FTagBrowserFunctionAddress   :   TCallBackFunctionTagBrowser   read
PTagBrowserFunctionAddress write PTagBrowserFunctionAddress;
      Property   FDLLBrowserFunctionAddress   :   TCallBackFunctionDLLBrowser   read
PDLLBrowserFunctionAddress write PDLLBrowserFunctionAddress;

      Property   FRefreshIconFunctionAddress   :   TCallBackFunctionRefreshIcon   read
PRefreshIconFunctionAddress write PRefreshIconFunctionAddress;
      Property FPlatformActionFunctionAddress : TCallBackFunctionPlatformAction read
PPlatformActionFunctionAddress write PPlatformActionFunctionAddress;

      Property   FOPCConnectedFunctionAddress   :   TCallBackFunctionOPCConnected   read
POPCConnectedFunctionAddress write POPCConnectedFunctionAddress;
      Property   FPlatformModeFunctionAddress   :   TCallBackFunctionPlatformMode   read
PPlatformModeFunctionAddress write PPlatformModeFunctionAddress;
      Property  FUserPrivilegeFunctionAddress  :  TCallBackFunctionUserPrivilege  read
PUserPrivilegeFunctionAddress write PUserPrivilegeFunctionAddress;
      Property     FUserNameFunctionAddress     :     TCallBackFunctionUserName     read
PUserNameFunctionAddress write PUserNameFunctionAddress;
      Property FEditInternalTagFunctionAddress : TCallBackFunctionEditInternalTag read
PEditInternalTagFunctionAddress write PEditInternalTagFunctionAddress;

      Property   FRaiseActionFunctionAddress   :   TCallBackFunctionRaiseAction   read
PRaiseActionFunctionAddress write PRaiseActionFunctionAddress;
```

```
      Property    FRaiseAlarmFunctionAddress    :    TCallBackFunctionRaiseAlarm    read
PRaiseAlarmFunctionAddress write PRaiseAlarmFunctionAddress;

      // Upwards Communication
      Procedure SetTagValue(ATag : String; AValue : Double);
      Function GetTagValue(ATag : String) : Double;
      Function IconDescription : String; override;
      Function DLLBrowser(ADLLTypeSET: TDLLTypeSET;
                          var ADLLRefs : TDLLPointerReferenceArray;
                          AMultiSelectable : Boolean = False;
                          AAutoReturnAll : Boolean = False;
                          ABrowserCaption : String = ''): Boolean;
      Function RefreshIcon : Boolean;
      Function OPCConnected : Boolean;
      Function PlatformMode : TPlatformMode;
      Function UserPrivilege : TUserPrivilege;
      Function UserName : String;
      // FUNCTIONS USED FOR DOWNWARD COMMUNICATION
      Procedure  Run(ANowTime  :  TDateTime;  ATimeMode  :  TTimeMode;  APlatformMode  :
TPlatformMode);
            Procedure DoVisualStuff;
            Procedure UpdateDrawIconsInHighContras;
      Procedure Draw(var ABitmap: TBitmap);

      Procedure ActivatePumpEditForm; override;
      Procedure ActivatePumpViewForm; override;

      Procedure SaveFile(AFileStream : TFileStream);
      Procedure LoadFile(AFileStream : TFileStream);
        Procedure LoadFileStructure1(AFileStream : TFileStream);
        Procedure LoadFileStructure2(AFileStream : TFileStream);
        Procedure LoadFileStructure3(AFileStream : TFileStream);
        Procedure LoadFileStructure4(AFileStream : TFileStream);
        Procedure LoadFileStructure5(AFileStream : TFileStream);
        Procedure LoadFileStructure6(AFileStream : TFileStream);
        Procedure LoadFileStructure7(AFileStream : TFileStream);
        Procedure LoadFileStructure8(AFileStream : TFileStream);
        Procedure LoadFileStructure9(AFileStream : TFileStream);
        Procedure LoadFileStructure10(AFileStream : TFileStream);
        Procedure LoadFileStructure11(AFileStream : TFileStream);

    end;
```

Code block 8 - Pump

1.2.2. Pump Editor

The Pump editor unit encapsulates a user interface that enables the user to alter the settings associated with each pump. The following figure shows this editor:

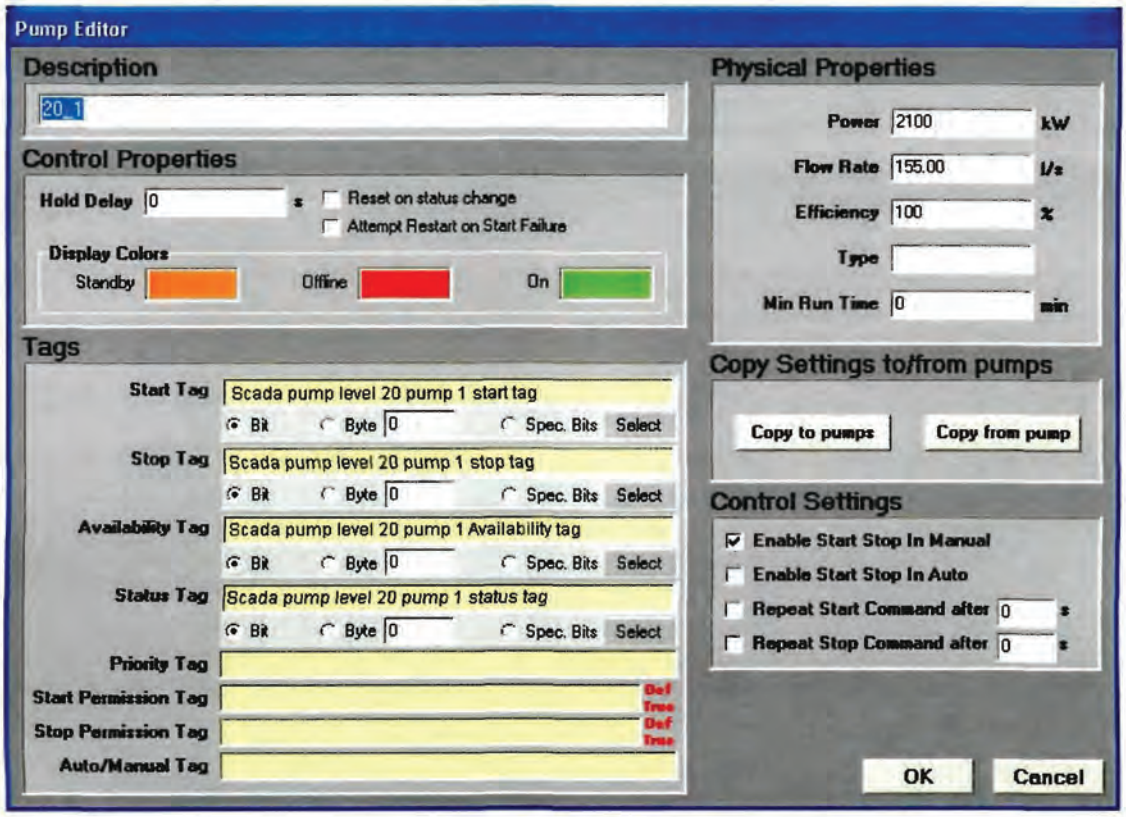


Figure 1-5 Pump DLL Editor

The following shows the interface of the Pump editor Unit:

```
unit PumpEditFrm;

interface

uses
  // This form is used to edit
  Pump,

  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, CheckLst;

type
  TPumpEditForm = class(TForm)
    ButtonOK: TButton;
    ButtonCancel: TButton;
    ColorDialog1: TColorDialog;
    pnlTags: TPanel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label9: TLabel;
    Label18: TLabel;
    edtPriorityTag: TEdit;
    edtStartPermissionTag: TEdit;
    Panel1: TPanel;
    radBitStart: TRadioButton;
    radByteStart: TRadioButton;
```

```
    edtByteStart: TEdit;
    radStartSpecBit: TRadioButton;
    Panel2: TPanel;
    radBitStop: TRadioButton;
    radByteStop: TRadioButton;
    edtByteStop: TEdit;
    radStopSpecBit: TRadioButton;
    edtAutoManualTag: TEdit;
    Panel3: TPanel;
    radBitStatus: TRadioButton;
    radByteStatus: TRadioButton;
    edtByteStatus: TEdit;
    radStatusSpecBit: TRadioButton;
    Panel4: TPanel;
    radBitAvailability: TRadioButton;
    radByteAvailability: TRadioButton;
    edtByteAvailability: TEdit;
    radAvailabilitySpecBit: TRadioButton;
    pnlControlProperites: TPanel;
    chkAttemptRestart: TCheckBox;
    pnlDescription: TPanel;
    edtDescription: TEdit;
    GroupColors: TGroupBox;
    Label8: TLabel;
    pnlStandbyColour: TPanel;
    Label10: TLabel;
    pnlOfflineColour: TPanel;
    Label11: TLabel;
    pnlOnColour: TPanel;
    Label1: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    btnStartSelectBits: TPanel;
    btnStopSelectBits: TPanel;
    btnAvailabilitySelectBits: TPanel;
    btnStatusSelectBits: TPanel;
    edtStartTag: TEdit;
    Label26: TLabel;
    Label27: TLabel;
    edtHoldDelay: TEdit;
    edtStopTag: TEdit;
    edtAvailabilityTag: TEdit;
    edtStatusTag: TEdit;
    Label22: TLabel;
    pnlPhysical: TPanel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    Label21: TLabel;
    edtPower: TEdit;
    edtFlowRate: TEdit;
    edtEfficiency: TEdit;
    edtType: TEdit;
    edtMinRunTime: TEdit;
    Label23: TLabel;
    Panel5: TPanel;
    btnCopyTo: TButton;
    btnCopyFrom: TButton;
    Label28: TLabel;
    Label29: TLabel;
    edtStopPermissionTag: TEdit;
    Label30: TLabel;
    Panel6: TPanel;
    CheckEnableStartStopInManual: TCheckBox;
    CheckEnableStartStopInAuto: TCheckBox;
    chkRepeatStart: TCheckBox;
    chkRepeatStop: TCheckBox;
    edtRepeatStartTime: TEdit;
    edtRepeatStopTime: TEdit;
```

```
    chkResetOnStatus: TCheckBox;

    Constructor Create(AOwner : TComponent; AParent : TObject); reintroduce;
    procedure FormShow(Sender: TObject);

    procedure ButtonOKClick(Sender: TObject);
    procedure ButtonCancelClick(Sender: TObject);
    procedure FormKeyPress(Sender: TObject; var Key: Char);
    procedure edtStartTagDblClick(Sender: TObject);
    procedure edtStopTagDblClick(Sender: TObject);
    procedure edtAvailabilityTagDblClick(Sender: TObject);
    procedure edtStatusTagDblClick(Sender: TObject);
    procedure edtPriorityTagDblClick(Sender: TObject);
    procedure edtStartPermissionTagDblClick(Sender: TObject);
    procedure pnlStandbyColourClick(Sender: TObject);
    procedure pnlOfflineColourClick(Sender: TObject);
    procedure pnlOnColourClick(Sender: TObject);
    procedure btnStartSelectBitsClick(Sender: TObject);
    procedure btnStopSelectBitsClick(Sender: TObject);
    procedure btnAvailabilitySelectBitsClick(Sender: TObject);
    procedure btnStatusSelectBitsClick(Sender: TObject);
    procedure btnCopyToClick(Sender: TObject);
    procedure btnCopyFromClick(Sender: TObject);
    procedure   edtDescriptionKeyDown(Sender:   TObject;   var   Key:   Word;   Shift:
TShiftState);
    procedure Label2DblClick(Sender: TObject);
    procedure edtStopPermissionTagDblClick(Sender: TObject);

  private
    POwner : TComponent;
    PParent : TObject;

    Function Pump : TPump;

    Function   ActivateSpecificBitEditorForm(var   AValue,   AGreyValue   :   integer)   :
boolean;

    Procedure LoadInformation;
    Procedure SaveInformation;
  public
  end;
```

Code block 9 - Pump Editor

1.2.3. Pump Viewer

The Pump Viewer gives active information about the pump to the user. The data relayed on the Pump Viewer is updated in real time. The following figure shows an example of the Pump Viewer:

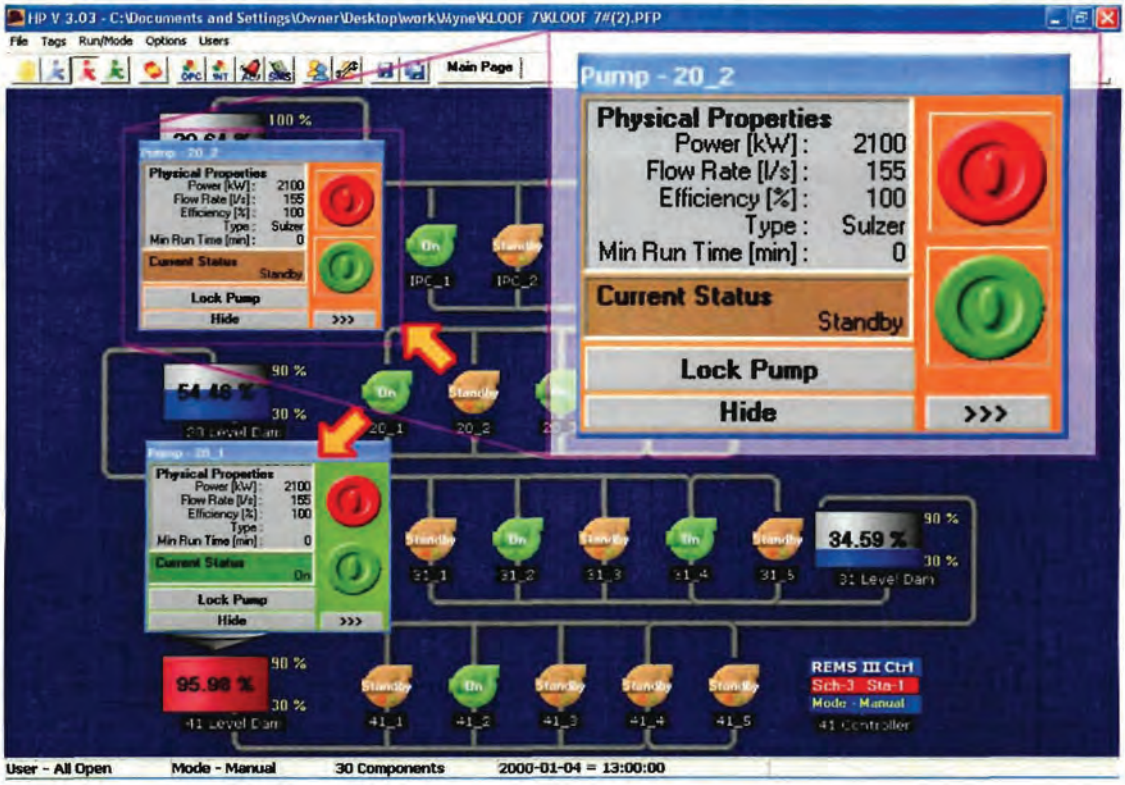


Figure 1-6 Pump Viewer

The following shows the Pump viewer unit interface:

```
unit PumpViewFrm;

interface

uses
  // This form is used to edit
  Pump,
  HVACIPlatformTypes,

  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Buttons, Mask;

type
  TPumpViewForm = class(TForm)
    ButtonHide: TPanel;
    pnlProperties: TPanel;
    Label10: TLabel;
    Label18: TLabel;
    Label22: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    lblPower: TLabel;
    lblFlowRate: TLabel;
    lblEfficiency: TLabel;
    lblType: TLabel;
    lblMinRunTime: TLabel;
    pnlStatus: TPanel;
    lblPumpStatus: TLabel;
    Timer1: TTimer;
    Label3: TLabel;
    Label23: TLabel;
```

```
    btnStop: TSpeedButton;
    btnStart: TSpeedButton;
    btnLock: TPanel;
    Panel1: TPanel;
    lblPumpRunTime: TLabel;
    Label2: TLabel;
    btnResetRunTime: TButton;
    edtHours: TMaskEdit;
    edtMinutes: TMaskEdit;
    edtSeconds: TMaskEdit;
    Label1: TLabel;
    ButtonDisplayRunningHours: TPanel;

    Constructor Create(AOwner : TComponent; AParent : TObject); reintroduce;

    procedure ButtonHideClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure btnStartClick(Sender: TObject);
    procedure btnStopClick(Sender: TObject);
    procedure btnLockClick(Sender: TObject);
    procedure btnResetRunTimeClick(Sender: TObject);
    procedure ButtonDisplayRunningHoursClick(Sender: TObject);
  private
    { Private declarations }
  public
    POwner : TComponent;
    PParent : TObject;

    Function Pump : TPump;

    Procedure UpdateAll;
      Procedure UpdateColour;
      Procedure UpdatePhysicalProperties;
      Procedure UpdatePumpStatus;
      Procedure UpdatePumpRunTime;

    Procedure LoadInformation;
  end;
```

Code block 10 - Pump Viewer

1.3. Dam DLL

The Dam DLL, same as the Pump DLL, is code that connects to actual physical dams and represents the dams inside the coding environment.

The next figure shows a platform setup in which five instances of the Dam DLL is used to represent five dams in an actual pump system setup.

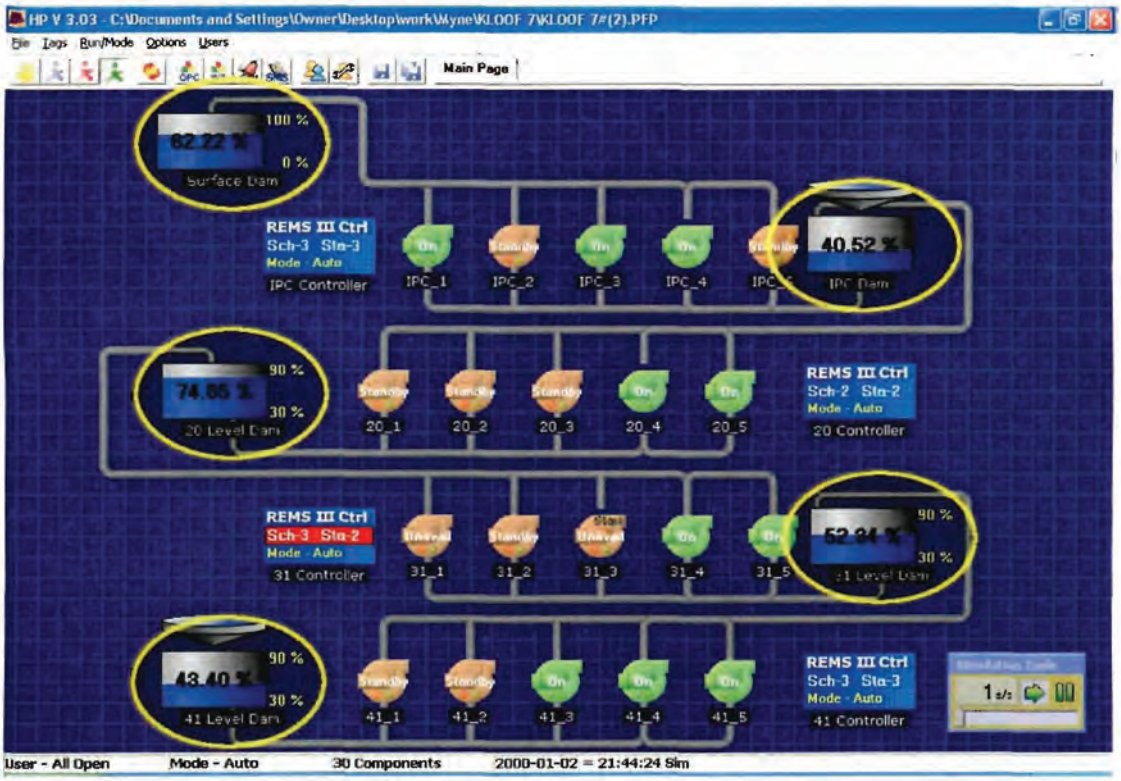


Figure 1-7 Dam DLL

1.3.1. Dam

The Dan Unit encapsulates all the units that make up the Dam DLL. This unit also manages all the functionalities of the Dam DLL and saves all the user defined settings associated with each actual dam.

The following figure shows the units in the Dam DLL and how they relate to one another:

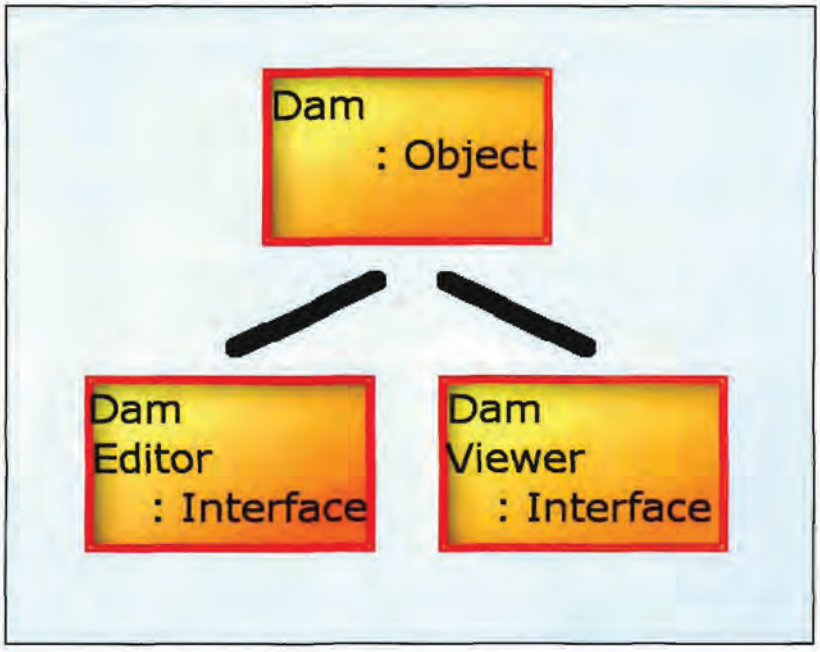


Figure 1-8 Unit structure of the Dam DLL

The following shows the interface of the Dam Unit:

```
unit Dam;

interface

Uses
    // Afgelei Van
    // Hierargie
    // Ander Tools
    //PlatformIcon,  // This is used for Upward Communication.   Function in Platform
Icon enables the DLL to usitlise functions in the Main app
    HVACIPlatformTypes,
    HVACIValue,
    DamInterface,
    PumpInterface,
    ValveInterface,
    FridgePlantInterface,
    // Windows Tools Self
    Forms,
    Controls,
    Dialogs,
    Graphics,       // TCanvas
    Types,          // TPoint
    SysUtils,       // ExtractFilePath
    Classes;
    // Windows Tools Automaties;

Type
    TDamSimulationFlowSource = Record
            DLL : TDLLPointerReference;
            ValveBoundDLL : TDLLPointerReference;
            ValveStatus : Boolean; {True - Valve Flow Should Be added if valve is open
// False - falve Flow Shoud Be added if valve is closed}
        end;

Type
    TDam = Class(TDamInterface)
        Constructor Create(AOwner: TComponent; AParent: TObject); Reintroduce;
```

```
        Destructor  Free; Reintroduce;
    Private
        POwner : TComponent;
        PParent : TObject;

        PLoadedDLLs : Boolean;

        PDamGraphic : TBitmap;
        PSettlerGraphic : TBitmap;

        PSimDamLevel : Double;
        PSimDamTemp : Double;

        // FUnction Display Variables
        PPrevDamLevel : Double;
        PPrevFlucDamLevel : Double;
        PPrevFlucDamLevelTime : Double; // The time the previous FLuck Dam Level was
recorded

        PDrawIconsInHighContras : Boolean;
        PDrawIconsInHighContrasTimeFlagUpdate : Double;

        PNowTime : TDateTime;
        PSimHoldTime : TDateTime;

        PTimeMode : TTimeMode;
        PPlatformMode : TPlatformMode;

        PSetTagValueFunctionAddress : TCallbackFunctionSETTagValue;
        PGetTagValueFunctionAddress : TCallbackFunctionGETTagValue;
        PIconDescriptionFunctionAddress : TCallbackFunctionIconDescription;
        PTagBrowserFunctionAddress : TCallBackFunctionTagBrowser;
        PDLLBrowserFunctionAddress : TCallBackFunctionDLLBrowser;

        PRefreshIconFunctionAddress : TCallBackFunctionRefreshIcon;
        PPlatformActionFunctionAddress : TCallBackFunctionPlatformAction;

        POPCConnectedFunctionAddress : TCallBackFunctionOPCConnected;
        PPlatformModeFunctionAddress : TCallBackFunctionPlatformMode;
        PUserPrivilegeFunctionAddress : TCallBackFunctionUserPrivilege;
        PUserNameFunctionAddress : TCallBackFunctionUserName;
        PEditInternalTagFunctionAddress : TCallBackFunctionEditInternalTag;

        PRaiseActionFunctionAddress : TCallBackFunctionRaiseAction;
        PRaiseAlarmFunctionAddress : TCallBackFunctionRaiseAlarm;

        PViewForm : TForm;
        PFreeViewForm : Boolean;

        Function OtherDam(APointer : Pointer) : TDam;
        Function Pump(APointer : Pointer) : TPumpInterface;
        Function Valve(APointer : Pointer) : TValveInterface;
        Function FridgePlant(APointer : Pointer) : TFridgePlantInterface;

        Function GetTime : Double;

        Procedure InitialiseerVeranderlikes;
        //Procedure Draw(APosition : TPoint ; ACanvas : TCanvas);
            Procedure DrawGraphic(var ABitmap : TBitmap);
            Procedure DrawDescription(var ABitmap: TBitmap);
            Procedure DrawLevel(var ABitmap: TBitmap);
            Procedure DrawTemp(var ABitmap: TBitmap);

        Function CalculateSimulatedDamLevel: Double;
            Procedure TestValidityofAllInAndOutFlows;
                        Procedure    ClearOutAllMissingDLLReferences(ASimFlows    :
Array of TDamSimulationFlowSource);
                Function  CalculateSimInflowRate : Double; {L/S}
                Function  CalculateSimOutFlowRate : Double; {L/S}
            Function  CalculateNewSimulatedDamTempreture(ADeltaSec : Double) : Double;
            Procedure BumpTempratureIntoREcievingPumpsAndValves(ATemp : Double);

        Procedure LoadDamGraphic;
        Procedure LoadDLLs;
```

```
        Procedure ViewFormHide(Sender: TObject);
    Public
        // Physical Properties
        FDescriptionOld     : String;
        FDescription     : String150;
        FVolume          : Integer; {m^3}
        FMaxLevel        : THVACIValue;
        FMinLevel        : THVACIValue;

        // Tags
        FLevelTag : THVACIValue;
        FTempTag : THVACIValue;

        // Fluctuations
        FFilterFluctuations : Boolean;
        FFilterWindow : Double;

        // Display Properties
        FDisplayMinMax : Boolean;
        FDisplaySettler : Boolean;

        // Simulation
        FSimulatedFlow : THVACIValue;
        FSimulatedFlowTemp : THVACIValue;
        FSimulatedFlowSign : Integer;

        FSimulatedOverrideSimualtedValueWithLevelTag : Boolean;

        FSimulatedFlowInOther : Double;
        FSimulatedFlowOutOther : Double;
        FInitialSimLevel : Double;
        FInitialSimTemp : Double;

        FSimInFlows : Array of TDamSimulationFlowSource;
        FSimOutFlows : Array of TDamSimulationFlowSource;

        // Tools Handeling the FSinInFlows ans FSimOutFlows
        Function  NumberOfSimInFlows : Integer;
        Function  NumberOfSimOutFlows : Integer;

        Function  GetMaxLevel : Double; override;
        Function  GetMinLevel : Double; override;
        Function  GetLevel : Double; override;
        Function  GetVolume : Double; override;

        Procedure ResetDamLevel;
        Procedure ResetAllDamLevels;

        // Simulation Communication
        Procedure SetInFlow(AValue : double); override;
        Procedure SetOutFlow(AValue : double); override;
        Function  IsSimulating : Boolean;
        Function  GetTemperature : Double; Override;

        Property  FSetTagValueFunctionAddress  :  TCallbackFunctionSetTagValue  read
PSetTagValueFunctionAddress write PSetTagValueFunctionAddress;
        Property  FGetTagValueFunctionAddress  :  TCallbackFunctionGetTagValue  read
PGetTagValueFunctionAddress write PGetTagValueFunctionAddress;
        Property  FIconDescriptionFunctionAddress  :  TCallbackFunctionIconDescription
read PIconDescriptionFunctionAddress write PIconDescriptionFunctionAddress;
        Property  FTagBrowserFunctionAddress  :  TCallBackFunctionTagBrowser  read
PTagBrowserFunctionAddress write PTagBrowserFunctionAddress;
        Property  FDLLBrowserFunctionAddress  :  TCallBackFunctionDLLBrowser  read
PDLLBrowserFunctionAddress write PDLLBrowserFunctionAddress;

        Property  FRefreshIconFunctionAddress  :  TCallBackFunctionRefreshIcon  read
PRefreshIconFunctionAddress write PRefreshIconFunctionAddress;
        Property FPlatformActionFunctionAddress : TCallBackFunctionPlatformAction read
PPlatformActionFunctionAddress write PPlatformActionFunctionAddress;

        Property  FOPCConnectedFunctionAddress  :  TCallBackFunctionOPCConnected  read
POPCConnectedFunctionAddress write POPCConnectedFunctionAddress;
```

```
        Property  FPlatformModeFunctionAddress  :  TCallBackFunctionPlatformMode  read
PPlatformModeFunctionAddress write PPlatformModeFunctionAddress;
        Property  FUserPrivilegeFunctionAddress  :  TCallBackFunctionUserPrivilege  read
PUserPrivilegeFunctionAddress write PUserPrivilegeFunctionAddress;
        Property    FUserNameFunctionAddress    :    TCallBackFunctionUserName    read
PUserNameFunctionAddress write PUserNameFunctionAddress;
        Property  FEditInternalTagFunctionAddress  :  TCallBackFunctionEditInternalTag
read PEditInternalTagFunctionAddress write PEditInternalTagFunctionAddress;

        Property  FRaiseActionFunctionAddress  :  TCallBackFunctionRaiseAction  read
PRaiseActionFunctionAddress write PRaiseActionFunctionAddress;
        Property  FRaiseAlarmFunctionAddress  :  TCallBackFunctionRaiseAlarm  read
PRaiseAlarmFunctionAddress write PRaiseAlarmFunctionAddress;

        // Upwards Communication
        Procedure SetTagValue(ATag : String; AValue : Double);
        Function GetTagValue(ATag : String) : Double;
        Function SetIconDescription(AValue : String) : String;
        Function IconDescription : String; override;
        Function TagBrowser : String;
        Function DLLBrowser(ADLLTypeSET: TDLLTypeSET;
                            var ADLLRefs : TDLLPointerReferenceArray;
                            AMultiSelectable : Boolean = False;
                            AAutoReturnAll : Boolean = False;
                            ABrowserCaption : String = ''): Boolean;
        Function RefreshIcon : Boolean;

        Function UserName : string;
        // FUNCTIONS USED FOR DOWNWARD COMMUNICATION
        Procedure  Run(ANowTime  :  TDateTime;  ATimeMode  :  TTimeMode;  APlatformMode  :
TPlatformMode);
            Procedure UpdateDrawIconsInHighContrasFlag;
        Procedure Draw(var ABitmap: TBitmap);

        Procedure ActivateDamEditForm; override;
        Procedure ActivateDamViewForm; override;

        Procedure SaveFile(AFileStream : TFileStream);
        Procedure LoadFile(AFileStream : TFileStream);
          Procedure LoadFileStructure1(AFileStream: TFileStream);
          Procedure LoadFileStructure2(AFileStream: TFileStream);
          Procedure LoadFileStructure3(AFileStream: TFileStream);
          Procedure LoadFileStructure4(AFileStream: TFileStream);
          Procedure LoadFileStructure5(AFileStream: TFileStream);
          Procedure LoadFileStructure6(AFileStream: TFileStream);
          Procedure LoadFileStructure7(AFileStream: TFileStream);
          Procedure LoadFileStructure8(AFileStream: TFileStream);

    end;
```

Code block 11 - Dam

1.3.2. Dam Editor

The Dam editor Unit encapsulates a user interface that enables the user to alter the settings regarding each Dam instance that is created. The next figure shows the Dam DLL Editor.

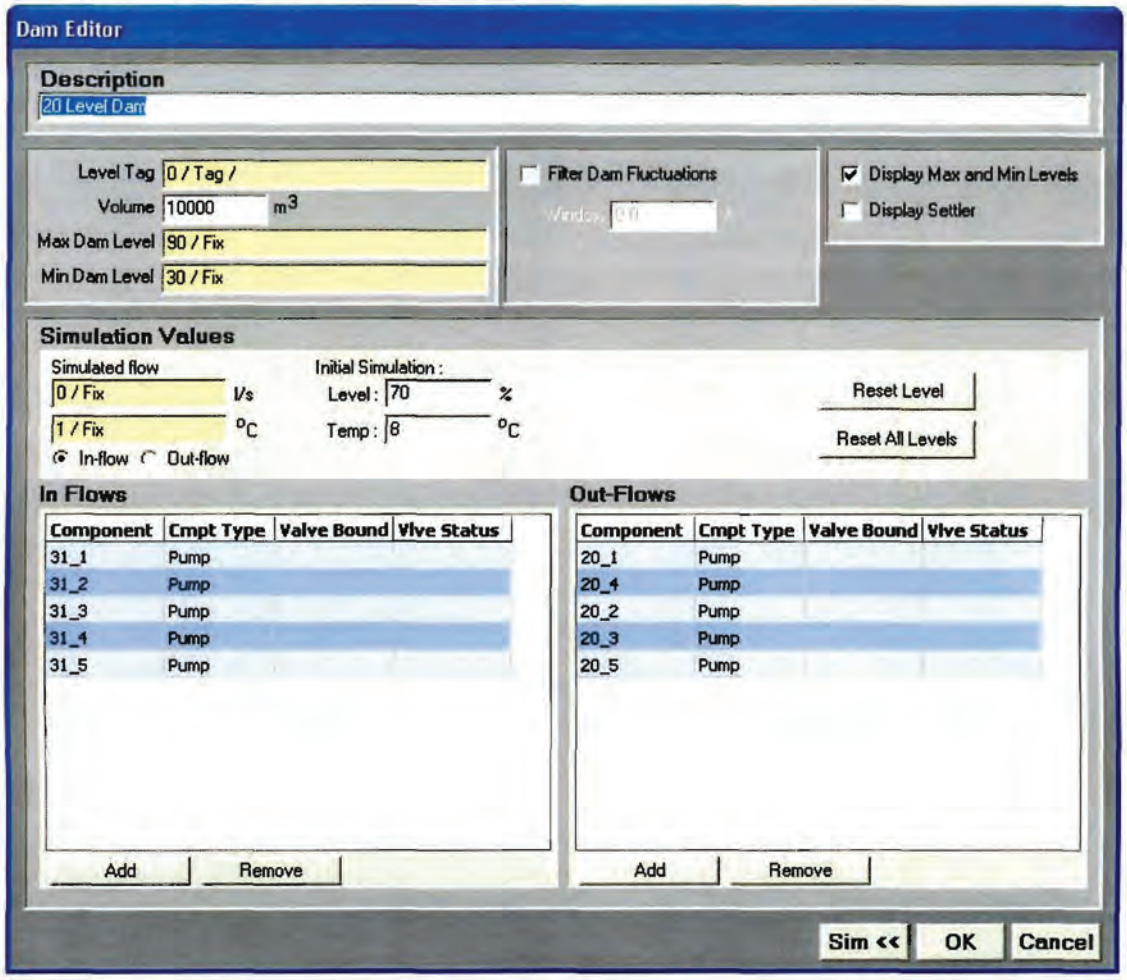


Figure 1-9 Dam DLL Editor

The following shows the interface of the Dam editor Unit:

```
unit DamEditFrm;

interface

uses
    // This form is used to edit
    Dam,
    // Include Tools Self
    Math,                          // Min

    HVACIPlatformtypes,
    HVACIValue,

    PumpInterface,
    ValveInterface,
    DamInterface,
    FridgePlantInterface,

    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, ExtCtrls, Grids, BaseGrid, AdvGrid, frmctrllink;

type
  TDamEditForm = class(TForm)
```

```
    ButtonOK: TButton;
    ButtonCancel: TButton;
    PanelSimulationValues: TPanel;
    pnlInFlows: TPanel;
    btnAddPumpIn: TButton;
    btnRemovePumpIn: TButton;
    Panel1: TPanel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    edtSimFlowIn: TEdit;
    edtInitialSimLevel: TEdit;
    radInFlow: TRadioButton;
    radOutFlow: TRadioButton;
    btnReset: TButton;
    Panel2: TPanel;
    btnAddPumpOut: TButton;
    btnRemovePumpOut: TButton;
    Label18: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    pnlDescription: TPanel;
    edtDescription: TEdit;
    Panel3: TPanel;
    Label3: TLabel;
    Label4: TLabel;
    Label2: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    edtVolume: TEdit;
    edtLevelTag: TEdit;
    edtMaxDamLevel: TEdit;
    edtMinDamLevel: TEdit;
    Label1: TLabel;
    btnSim: TButton;
    GridInFlows: TAdvStringGrid;
    GridOutFlows: TAdvStringGrid;
    btnResetAll: TButton;
    EditSimulationFlowInTemp: TEdit;
    Label9: TLabel;
    Label16: TLabel;
    EditInitialSimulationTemp: TEdit;
    Label22: TLabel;
    Label23: TLabel;
    Label21: TLabel;
    Label24: TLabel;
    Panel5: TPanel;
    chkFilterFluctuations: TCheckBox;
    lblFilterPerc: TLabel;
    edtFilterWindow: TEdit;
    lblFilterWindow: TLabel;
    CheckOverrideSimulatedValueWithLevelTag: TCheckBox;
    chkDisplayMaxMin: TCheckBox;
    chkDisplaySettler: TCheckBox;
    Label8: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    EditTempValue: TEdit;

    Constructor Create(AOwner : TComponent; AParent : TObject); reintroduce;
    procedure FormKeyPress(Sender: TObject; var Key: Char);
    procedure FormShow(Sender: TObject);

    procedure ButtonOKClick(Sender: TObject);
    procedure ButtonCancelClick(Sender: TObject);
    procedure edtLevelTagDblClick(Sender: TObject);
    procedure edtMaxDamLevelDblClick(Sender: TObject);
    procedure edtMinDamLevelDblClick(Sender: TObject);
    procedure edtSimFlowInDblClick(Sender: TObject);
    procedure btnResetClick(Sender: TObject);
    procedure btnSimClick(Sender: TObject);
```

```
    procedure GridOutFlowsClickCell(Sender: TObject; ARow, ACol: Integer);

    procedure btnResetAllClick(Sender: TObject);
    procedure EditSimulationFlowInTempDblClick(Sender: TObject);
    procedure btnAddPumpInClick(Sender: TObject);
    procedure btnRemovePumpInClick(Sender: TObject);
    procedure GridInFlowsClickCell(Sender: TObject; ARow, ACol: Integer);
    procedure btnAddPumpOutClick(Sender: TObject);
    procedure btnRemovePumpOutClick(Sender: TObject);
    procedure chkFilterFluctuationsClick(Sender: TObject);
    procedure EditTempValueDblClick(Sender: TObject);
  private
    POwner : TComponent;
    PParent : TObject;

    PTempLevelValue : THVACIValue;
    PTempTempValue : THVACIValue;
    PTempMaxValue : THVACIValue;
    PTempMinValue : THVACIValue;
    PTempFlow : THVACIValue;
    PTempFlowTemp : THvaciValue;

    PSimInFlows : Array of TDamSimulationFlowSource;
    PSimOutFlows : Array of TDamSimulationFlowSource;

    // All tools handeling the Inflow
    Procedure AddSimInflows;
    Procedure DeleteSimInflow(AIndex : Integer);
    Procedure AddSimInFlowValve(AIndex : Integer);
    Procedure DeleteSimInflowValve(AIndex : Integer);
    Procedure SwitchSimInflowValveStatus(AIndex : Integer);
    Function  NumberOfPSimInFlows : Integer;

    // All tools handeling the Outflow
    Procedure AddSimOutflows;
    Procedure DeleteSimOutflow(AIndex : Integer);
    Procedure AddSimOutflowValve(AIndex : Integer);
    Procedure DeleteSimOutflowValve(AIndex : Integer);
    Procedure SwitchSimOutflowValveStatus(AIndex : Integer);
    Function  NumberOfPSimOutflows : Integer;

    Function  Dam : TDam; OverLoad;
    Function  OtherDam(APointer : Pointer) : TDamInterface;
    Function  Pump(APointer : Pointer) : TPumpInterface;
    Function  Valve(APointer : Pointer) : TValveInterface;
    Function  FridgePlant(APointer : Pointer) : TFridgePlantInterface;

    Procedure FixStringGrids;

    Procedure LoadInformation;
        Procedure LoadSimFlowsFromDam;
        Procedure LoadSimFlowsToStringGrid;
            Function  ValveStatusToString(AValveStatus : Boolean) : String;
    Procedure SaveInformation;
        Procedure SaveSimFlowsToDam;
  public
  end;
```

Code block 12 - Dam Editor

1.3.3. Dam Viewer

The Dam viewer Unit encapsulates a user interface that gives real time information to the user about each dam instance. The following figure shows the Dam viewer as seen by the user in normal operating conditions:

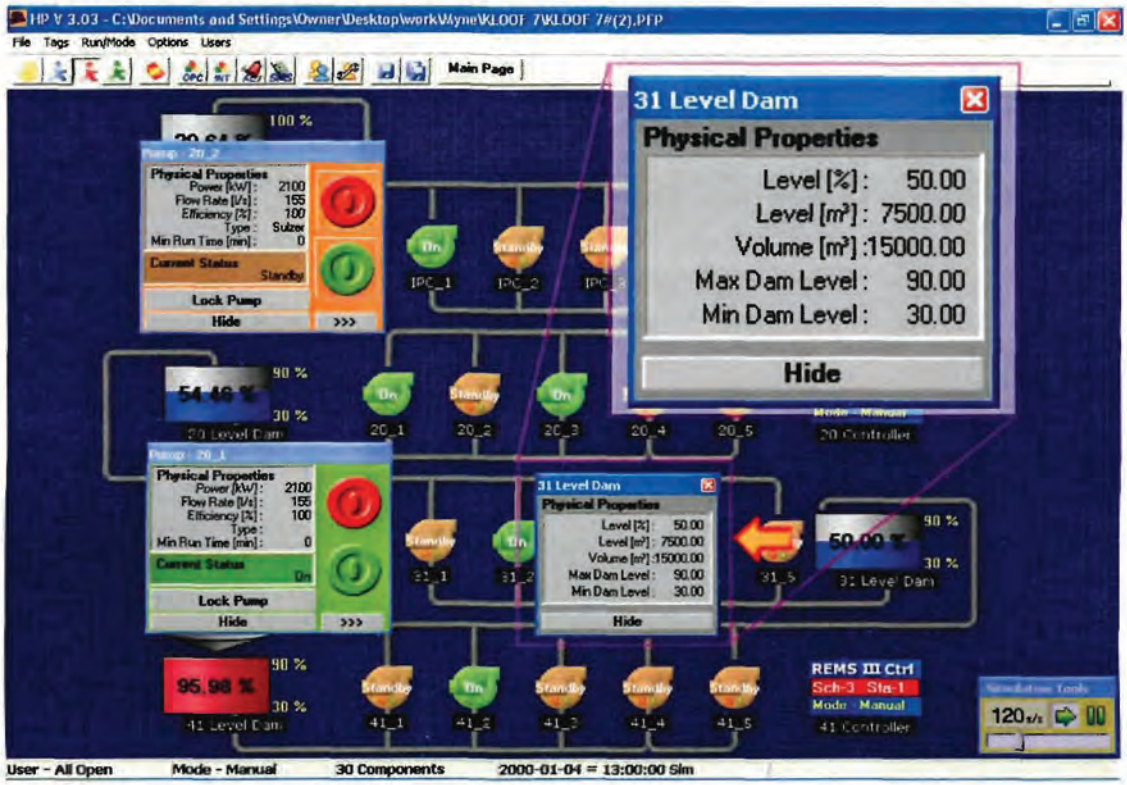


Figure 1-10 Dam Viewer

The following show the interface of the Dam viewer Unit:

```
unit DamViewFrm;

interface

uses
  // This form is used to edit
  Dam,

  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TDamViewForm = class(TForm)
    Label13: TLabel;
    PanelDownStreamDam: TPanel;
    Label14: TLabel;
    Label18: TLabel;
    Label22: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    lblLevelPerc: TLabel;
    lblLevel: TLabel;
    lblVolume: TLabel;
    lblMaxDamLevel: TLabel;
    lblMinDamLevel: TLabel;
    btnHide: TPanel;
    Timer1: TTimer;
    Constructor Create(AOwner : TComponent; AParent : TObject); reintroduce;
    procedure btnCloseClick(Sender: TObject);
    procedure btnHideClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
```

```
  private
    { Private declarations }
  public
    POwner : TComponent;
    PParent : TObject;

    Function Dam : TDam;

    Procedure LoadInformation;

    Procedure UpdateAll;
      Procedure UpdateProperties;
  end;
```

Code block 13 - Dam Viewer

1.4. Pump group controller DLL

The Pump group controller DLL encapsulates the code that governs the control engine that is responsible for the actual load shifting and reduced running costs.

The Pump group controller DLL is named so because an instance of this DLL type is created for each actual pump group. A pump group represents a pump station consisting of a series of pump all pumping water from the same source dam to the same destination dam.

The Pump group controller DLL consists of the following units as shown in the next figure and discussed thereafter.

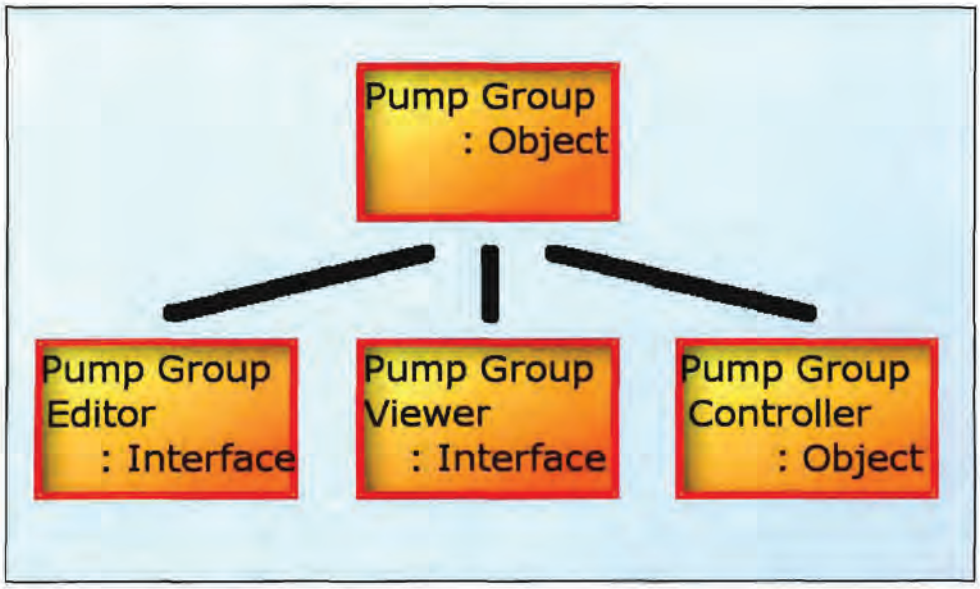


Figure 1-11 Unit structure of the Pump group controller DLL

1.4.1. Pumpgroup

The Pumpgroup Unit embodies all the settings and data that is associated with the Pump group controller DLL. This unit acts as communication between the Pump group controller DLL and the Platform.

```
unit PumpGroup;

interface
```

```
Uses
    // Afgelei van
    PumpGroupInterface,
    // Herargie
    PumpGroupController,

    // Ander Tools
    DamInterface,  // Weet hoe lyk die Dam DLL      Bere net verwysings na die Damme
    PumpInterface, // Weet hoe lyk die Pump DLL     Bere net verwysings na die Pompe

    PumpGroupViewFrm,

    HVACIPlatformTypes,
    PumpGroupIIITypes,

    HVACIValue,
    // Windows Tools Self
    Types,
    Graphics,
    Forms,
    Dialogs,
    Classes,
    // Windows Tools Automaties
    SysUtils;


Type
    TPumpGroup = Class(TPumpGroupInterface)
        Constructor Create(AOwner:TComponent;AParent:TObject); Reintroduce;
        Destructor  Free; Reintroduce;
    private
        FOwner : TComponent;
        FParent : TObject;

        PPumpGroupViewForm : TPumpGroupViewForm;
        PPumpGroupViewFormPosition : TPoint;

        PPumpGroupGraphic : TBitmap;

        PDLLPointersUpdated : Boolean;

        PTime : Double;
        PTimeMode : TTimeMode;
        PPlatformMode : TPlatformMode;

        PDisplayedSchedule : Integer;
        PDisplayedStatus : Integer;
        PDisplayedMode : TPlatformMode;

        PDrawIconsInHighContras : boolean;
        PDrawIconsInHighContrasTimeFlagUpdate : double;


        Procedure InitialiseerVeranderlikes;
          Procedure LoadPumpGroupGraphic;

        Procedure UpdateOtherDLLPointers;

        // These Two Function is for the HVACIValue
            // TOnActivateTagBrowserEvent = Function() : String of Object;//
            // TOnGetTagValueEvent = Function(ATag : String) : Double of Object;

        Function OnActivateTagBrowser() : String;
        Function OnGetTagValue(ATag : String) : Double;
        Function OnGetTime() : Double;
        Function  OnPromptTagEditor(ATagName  :  String;  AActivateEditor  :  Boolean  =
True) : Boolean;
    public
        // DLL Working Stuff
        FSetTagValueFunctionAddress :       TCallbackFunctionSETTagValue;
        FGetTagValueFunctionAddress :       TCallbackFunctionGETTagValue;
        FIconDescriptionFunctionAddress :   TCallbackFunctionIconDescription;
```

```
    FTagBrowserFunctionAddress :          TCallBackFunctionTagBrowser;
    FDLLBrowserFunctionAddress :          TCallBackFunctionDLLBrowser;

    FRefreshIconFunctionAddress :         TCallBackFunctionRefreshIcon;
    FPlatformActionFunctionAddress :      TCallBackFunctionPlatformAction;

    FOPCConnectedFunctionAddress :        TCallBackFunctionOPCConnected;
    FPlatformModeFunctionAddress :        TCallBackFunctionPlatformMode;
    FUserPrivilegeFunctionAddress :       TCallBackFunctionUserPrivilege;
    FUserNameFunctionAddress :            TCallBackFunctionUserName;
    FEditInternalTagFunctionAddress :     TCallBackFunctionEditInternalTag;

    FRaiseActionFunctionAddress :         TCallBackFunctionRaiseAction;
    FRaiseAlarmFunctionAddress :          TCallBackFunctionRaiseAlarm;

    FPumpGroupController : TPumpGroupController;

    // PumpGroupIII Eienskappe
    // GeneralSettrings
        FMaxNumberOfPumps : THVACIValue;
        FMinNumberOfPumps : THVACIValue;
        FTimeDelayStartUp : Byte;
        FTimeDelayStop : Byte;
        FToggleDelay : Integer;
        FControlType : TPumpGroupControlType;
        FScheduleTag : String100;
    // UPSTREAM ////////////////////////////////////////////
        FUpstreamMaxDamLevel : THVACIValue;
        FUpstreamOffsetTop : THVACIValue;
        FUpstreamConstrolRange : THVACIValue;
        FUpstreamOffsetBottom : THVACIValue;
        FUpstreamMinDamLevel : THVACIValue;

        FUpstreamDamPointer : TDLLPointerReference;
    // DownStream //////////////////////////////////////////
        FDownStreamMaxDamLevelControl : Boolean;
        FDownStreamOffsetTop : THVACIValue;
        FDownStreamMaxDamLevel : THVACIValue;
        FDownStreamControlRangeTop : THVACIValue;
        FDownStreamControlRangeBottom : THVACIValue;
        FDownStreamMinDamLevel : THVACIValue;
        FDownStreamOffSetBottom : THVACIValue;
        FDownStreamMinDamLevelControl : Boolean;

        FDownStreamMaxDamPointer : TDLLPointerReference;
        FDownStreamMinDamPointer : TDLLPointerReference;
    // MD Controller
        FMDControlActivated          : Boolean;
        FMDControlMaxNumberOfPumps   : Byte;
        FMDControlActivationParametersIndexes : Array of Byte;
        FMDControlDEActivationParametersIndexes : Array of Byte;
    // Pump
        FPumps : Array of TDLLPointerReference;
        FPumpStoppingMethod : TPumpStoppingMethod;
    // Pump Set Priority
        FPumpSetMax : array[1..4] of THVACIValue;
        FMaxPumpsAtSet : boolean;
    // Peak Anticipation
        FMorningPeakAnticipateMinutes : THVACIValue;
        FEveningPeakAnticipateMinutes : THVACIValue;

    // Al fie Interface Overrides


  Function GetPumpStoppingMethod : TPumpStoppingMethod; Override;
  Function GetPumpSetMax(AIndex : integer) : THVACIValue; Override;
  Function GetMaxPumpsAtSet : Boolean; Override;
  Procedure SetPumpStoppingMethod(AValue : TPumpStoppingMethod); Override;
  Procedure SetPumpSetMax(AIndex : integer; AValue : THVACIValue); Override;
  Procedure SetMaxPumpsAtSet(AValue : Boolean); Override;

    // ALL PROCEDURE USED TO COMMUNICATE TO PUMPS
        Function  AddPump() : Boolean;
```

```
            Procedure DeletePump(AIndex : Integer);
            Function  NumberOfPumps() : Integer; Override;
            Function  GetPump(AIndex : Integer) : Pointer; Override;
            Function  Pump(APointer : Pointer) : TPumpInterface; overload; Override;
            Function  Pump(AIndex : Integer) : TPumpInterface; overload; Override;
            Function     Pump(ADescription  :  String)  :  TPumpInterface;  overload;
Override;

        // ALL PROCEDURES USED TO HANDLE DAM POINTERS
            Function  SelectADam : TDLLPointerReference;
            Function  DAM(ADamPointer : Pointer) : TDamInterface;
              Function UpstreamDam : TDamInterface; Override;
              Function DownstreamMaxDam : TDamInterface;
              Function DownstreamMinDam : TDamInterface;

        // MD MD MD MD die MD controll goeters
            Procedure AddFMDControlActivationParametersIndexes(AIndex : Integer);
            Procedure AddFMDControlDEActivationParametersIndexes(AIndex : Integer);

        // Reading Of Tags
            Function  GetUpsteamDamLevel : Double;
            Function  GetDownsteamMaxDamLevel : Double;
            Function  GetDownsteamMinDamLevel : Double;
        // Setting Of Tags
            Procedure SetScheduleTag(ASchedule : Byte);

        // Function that is declered in the Pump Group Interface.
        Function IconDescription : String; Override;

        Function ControlSchedule : Integer; Override;
        Function ControlStatus : Integer; Override;
        Function ControlUpperbound : Real; Override;

Function  MaxNumberOfRunningPumps : Integer; Override;
Function  MinNumberOfRunningPumps : Integer; Override;
Function  RTPPrice : Double; Override;

Function  USMaxDamLevel : Double; Override;
Function  USMinDamLevel : Double; Override;
Function  USUpperbound : Double; Override;
Function  USDamLevel : Double; Override;
Function  USSchedule : Integer; Override;

Function  DSMaxDamLevel : Double; Override;
Function  DSMinDamLevel : Double; Override;
Function  DSDamLevel : Double; Override;
Function  DSSchedule : Integer; Override;

        Function ControlCurrentElectricityPrice : Real; Override;

        Function DownstreamDam : TDamInterface; Override;

        Procedure ActivatePumpGroupEditForm(); Override;
        Procedure ActivatePumpGroupViewForm(); Override;
            Procedure GetPumpGroupViewFormPosition;
        Procedure ActivatePumpPriorityOrganiserForm;

        Procedure  Run(ANowTime  :  TDateTime;  ATimeMode  :  TTimeMode;  APlatformMode  :
TPlatformMode);
            Procedure UpdateDisplayInfo;
            Procedure UpdateDrawIconsInHighContras;
        Procedure Draw(var ABitmap: TBitmap);
            Procedure DrawGraphic(var ABitmap: TBitmap);



        Function SaveFile(AFileStream:TFileStream):Boolean;
        Function LoadFile(AFileStream:TFileStream):Boolean;
          Function LoadFileStructureVersion1(AFileStream:TFileStream):Boolean;
          Function LoadFileStructureVersion2(AFileStream:TFileStream):Boolean;
          Function LoadFileStructureVersion3(AFileStream:TFileStream):Boolean;
          Function LoadFileStructureVersion4(AFileStream:TFileStream):Boolean;
          Function LoadFileStructureVersion5(AFileStream:TFileStream):Boolean;
```

```
        Function LoadFileStructureVersion6(AFileStream:TFileStream):Boolean;
    end;
```

Code block 14 - Pumpgroup

1.4.2. Pumpgroup Controller

The Pumpgroup Controller Unit contains the code that drives the control philosophy behind the load shifting and running cost reductions. The unit is responsible for calculating the optimised schedule by which the pumps in the pump group is controlled.

```
unit PumpGroupController;

interface

Uses
    SysUtils,
//    RemsIIITypes,
    WhiteUnit,
    WhiteIntegerList,
    HVACIPlatformTypes,
    PumpGroupIIITypes,

    Math,

    Dialogs,        // Show Message
    Classes;

Type
    TPumpGroupController = Class(TObject)
        Constructor Create(AOwner:TComponent;AParent:TObject);
        Destructor  Destroy; Reintroduce;
    private
        FOwner : TComponent;
        FParent : TObject;

        PTime : Double;
        PTimeMode : TTimeMode;
        PPlatformMode : TPlatformMode;

        PLastActionString       : String;

        //  Get       Functions Calculates the value AND PASSES IT ON
        //  Calculate Functions Calculates the Value AND INSERTS it into the variable
//      Function  ResetController : Boolean;
//      Function  DoControlRun : Double;
            Procedure CalculateCurrentRTPPrice;
            Procedure CalculateUpperBound;

            Procedure CalculateUpstreamSchedule;
            Procedure CalculateLocalDownstreamControllersEnabled;
            Procedure CalculateDownStreamSchedule;

            Procedure CalculateMinNumberOfPumps;
            Procedure CalculateMaxNumberOfPumps;
                Function  IsMDCOntrollerActivated : Boolean;
                    Function      IsMDCOntrollerActivatedAcordingToActiveTimePoints   :
Boolean;
                    Function    IsMDCOntrollerActivatedAcordingToActivationParameters :
Boolean;
                    Function    IsMDCOntrollerActivatedAcordingToDEActivationParameters
: Boolean;

            Procedure CalculateSchedule;
```

```
                Procedure CalculateScheduleControlPrivilegeDownStreamDam;
                Procedure CalculateScheduleControlPrivilegeUpStreamDam;
                Procedure CalculateScheduleControlPrivilegeAdditive;

            Procedure CalculateCurrentStatus;
            Function CalculateCurrentSetStatus(ASet : integer) : Integer;
            Procedure ImplimentSchedule;
                Procedure StartAPump;
                Procedure StopAPump;
                    Procedure StartSelectedPump(AIndex : Integer);
                    Procedure StopSelectedPump(AIndex : Integer);
                    Function  FindSetInWhichPumpShouldBeStarted : Integer;
                    Function  FindSetInWhichPumpShouldBeStopped : Integer;
                        Function  FindNumberOfStartingSets : Integer;
                        Function   FindNumberOfPumpsRunningInSet(ASetNumber : Integer)
: Integer;
                        Function   FindNumberOfPumpsAvailableToStartInSet(ASetNumber :
Integer) : Integer;
                        Function    FindNumberOfPumpsAvailableToStopInSet(ASetNumber :
Integer) : Integer;

            Procedure CheckForRestartAtempts;

        // Other Tools
        Function  GetCurrentHour : Byte;
        Function  GetCurrentMinute : Byte;
        // Tools to the oudside World
        Function  GetNowTime : Double;
        Function  GetPRTPrice(AHour:Byte) : Double;
        Function  GetMinRTPPrice : Double;
        Function  GetMaxRTPPrice : Double;

        procedure SetLastActionString(const Value: String);

        procedure SetPlatformMode(const Value: TPlatformMode);

    public
        // Dam Levels
        FScheduleUpstream                   : Integer;
        FScheduleDownstream                 : Integer;
        FScheduleUpstreamImplimented        : Integer;
        FScheduleDownstreamImplimented      : Integer;

        FSchedule               : Integer;

        // Upstream Stuff
        FUpstreamDamLevel       : Double;
        FCurrentRTPPrice        : Double;
        FUpperBound             : Double;


        // DOwnStream Stuff
        FLocalDownStreamMaxDamLevelControl : Boolean; // dit is wat gestel is op die
intervlak - Is control geaktiveer
        FLocalDownStreamMinDamLevelControl : Boolean;
        FLocalDownStreamMaxDamLevelPermitted : Double; // dit is wat gestel is op die
intervlak
        FLocalDownStreamMinDamLevelPermitted : Double;
        FLocalDownStreamMaxDamLevelReal : Double; // dit is wat die dammer regtig in
die werklikheid is
        FLocalDownStreamMinDamLevelReal : Double;


        //  General Stuff
        FMaxNumberOfPumps       : Integer;
        FMinNumberOfPumps       : Integer;
        FCurrentStatus          : Integer;
        FTimePumpStarted        : Double;
        FMustResartPumpIndex    : Integer;
        FTimePumpStopped        : Double;

        // MDController
        FMDControllerActive     : Boolean;
```

```
        FTimeControllerStarted  : Double;

        Property    FPlatformMode    :    TPlatformMode    Read    PPlatformMode    write
SetPlatformMode;

        Property    FLastActionString    :    String    Read    PLastActionString    write
SetLastActionString;

        Function  ResetController : Boolean;
        Procedure     DoControlRun(ANowTime:     TDateTime;     ATimeMode:     TTimeMode;
APlatformMode: TPlatformMode);

        Procedure ScheduleHoppingUp;
        Procedure ScheduleHoppingDown;
    end;
```

Code block 15 - Pumpgroup Controller

1.4.3. Pumpgroup Editor

The Pumpgroup editor unit encapsulate a user interface that is used to alter the settings of the Pump group controller DLL. The following figure shows how the user sees this editor.

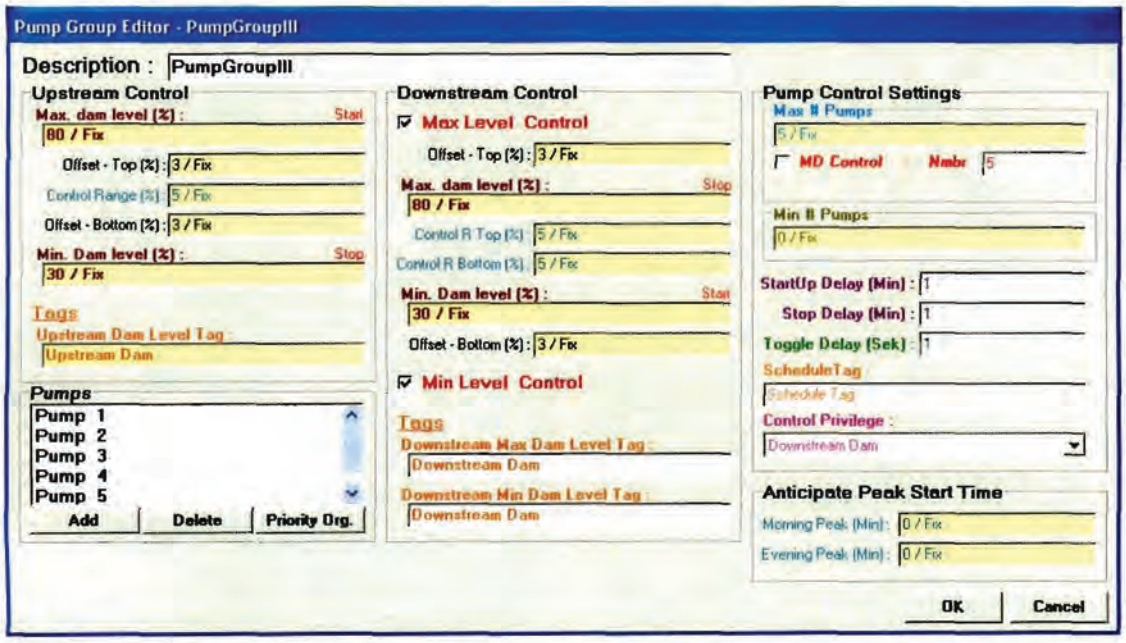


Figure 1-12 Pump group controller editor

The interface of the Pump croup editor unit is shown as follows:

```
unit PumpGroupEditFrm;

interface

uses
  PumpGroup,
```

```
  // Ather Tools
  PumpGroupIIITypes,

    DamInterface,
    PumpInterface,


  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, ExtCtrls;

type
  TPumpGroupEditForm = class(TForm)
    GroupUpstream: TGroupBox;
    GroupBox2: TGroupBox;
    EditUpstreamOffsetTop: TEdit;
    EditUpstreamControlRange: TEdit;
    EditUpstreamOffsetBottom: TEdit;
    EditUpstreamMaxDamLevel: TEdit;
    EditUpstreamMinDamLevel: TEdit;
    Label6: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label7: TLabel;
    EditDownStreamMaxDamLevel: TEdit;
    Label12: TLabel;
    EditDownStreamMinDamLevel: TEdit;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    EditDownStreamOffsetBottom: TEdit;
    EditDownstreamOffsetTop: TEdit;
    Label16: TLabel;
    Label19: TLabel;
    EditPumpGroupDescription: TEdit;
    CheckDownStreamMaxLevelControl: TCheckBox;
    CheckDownStreamMinLevelControl: TCheckBox;
    Label23: TLabel;
    Label26: TLabel;
    EditUpstreamDamLevelTag: TEdit;
    Label27: TLabel;
    Label28: TLabel;
    EditDownStreamMaxDamLevelTag: TEdit;
    GroupPumpGroupSettings: TGroupBox;
    Label21: TLabel;
    EditTimeDelayStartup: TEdit;
    Label30: TLabel;
    EditToggleDelay: TEdit;
    ComboControlPrivilege: TComboBox;
    Label31: TLabel;
    GroupBoxPumpHandler: TGroupBox;
    ButtonPumpAdd: TButton;
    ButtonPumpDelete: TButton;
    ButtonOK: TButton;
    ButtonCancel: TButton;
    Label32: TLabel;
    EditScheduleTag: TEdit;
    Label33: TLabel;
    EditDownStreamMinDamLevelTag: TEdit;
    Label34: TLabel;
    ListBoxPumps: TListBox;
    ButtonPriorityOrganiser: TButton;
    Label36: TLabel;
    EditTimeDelayStop: TEdit;
    Label37: TLabel;
    GroupBox1: TGroupBox;
    EditMaxNumberPumps: TEdit;
    CheckMaxNumberOfPumpsMDControl: TCheckBox;
    EditMaxNumberOfPumpsMDControl: TEdit;
    GroupBox3: TGroupBox;
    EditMinNumberPumps: TEdit;
    EditDownStreamControlRangeTop2: TEdit;
```

```
    EditDownStreamControlRangeBottom2: TEdit;
    Label3: TLabel;
    Label5: TLabel;
    lblDisabledMaxPumps: TLabel;
    GroupBox4: TGroupBox;
    Label29: TLabel;
    Label35: TLabel;
    EditAnticipateMorningPeak: TEdit;
    EditAnticipateEveningPeak: TEdit;

    Constructor Create(AOwner:TComponent;AParent:TObject); Reintroduce;
    procedure FormShow(Sender: TObject);

    procedure CheckDownStreamMaxLevelControlClick(Sender: TObject);
    procedure CheckDownStreamMinLevelControlClick(Sender: TObject);
    procedure ButtonPumpAddClick(Sender: TObject);
    procedure ButtonCancelClick(Sender: TObject);
    procedure ButtonOKClick(Sender: TObject);

    procedure EditScheduleTagDblClick(Sender: TObject);
    procedure ButtonPumpDeleteClick(Sender: TObject);

    procedure ListBoxPumpsDblClick(Sender: TObject);
    procedure EditMaxNumberPumpsClick(Sender: TObject);
    procedure ButtonPriorityOrganiserClick(Sender: TObject);

    procedure CheckMaxNumberOfPumpsMDControlClick(Sender: TObject);
    procedure EditMinNumberPumpsClick(Sender: TObject);
    procedure EditUpstreamOffsetTopClick(Sender: TObject);
    procedure EditUpstreamControlRangeClick(Sender: TObject);
    procedure EditUpstreamOffsetBottomClick(Sender: TObject);
    procedure EditUpstreamDamLevelTagClick(Sender: TObject);
    procedure EditDownstreamOffsetTopClick(Sender: TObject);
    procedure EditDownStreamOffsetBottomClick(Sender: TObject);
    procedure EditDownStreamMaxDamLevelTagClick(Sender: TObject);
    procedure EditDownStreamMinDamLevelTagClick(Sender: TObject);
    procedure EditUpstreamMaxDamLevelClick(Sender: TObject);
    procedure EditUpstreamMinDamLevelClick(Sender: TObject);
    procedure EditDownStreamControlRangeTop2Click(Sender: TObject);
    procedure EditDownStreamControlRangeBttom2Click(Sender: TObject);
    procedure EditDownStreamMinDamLevelClick(Sender: TObject);
    procedure EditDownStreamMaxDamLevelClick(Sender: TObject);
    procedure EditPumpGroupDescriptionChange(Sender: TObject);
    procedure EditPumpGroupDescriptionKeyDown(Sender: TObject;
      var Key: Word; Shift: TShiftState);
    procedure FormPaint(Sender: TObject);
    procedure EditAnticipateMorningPeakClick(Sender: TObject);
    procedure EditAnticipateEveningPeakClick(Sender: TObject);

  private
    FParent : TObject;
    FOwner : TComponent;

    Function  PumpGroup:TPumpGroup;

    Function  LoadPumpGroupInformation:Boolean;
        Procedure LoadDamNames;
        Function  LoadPumps:Boolean;
    Function  SavePumpGroupInformation:Boolean;

  public
    { Public declarations }
  end;
```

Code block 16 - Pumpgroup Editor

1.4.4. Pumpgroup Viewer

The Pumpgroup viewer unit encapsulates an user interface that gives information to the user during active control. The next figure shows an example of such an information panel.

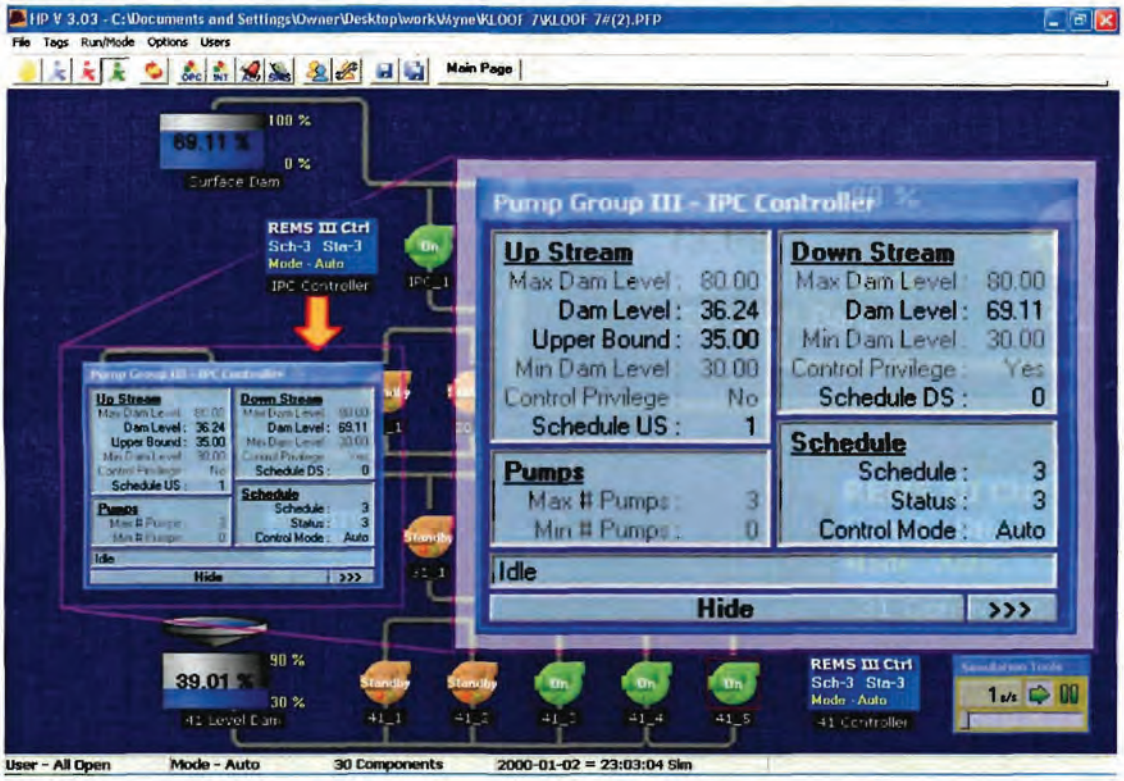


Figure 1-13 Pump group controller DLL information panel

The following shows the interface of the Pumpgroup viewer unit:

```
unit PumpGroupViewFrm;

interface

uses
    // Form Over
//    PumpGroup,
    // Other Tools
    PumpGroupIIITypes,
    HVACIPlatformTypes,

  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls;

type
  TPumpGroupViewForm = class(TForm)
    PanelUpstreamDam: TPanel;
    ButtonHide: TPanel;
    PanelDownStreamDam: TPanel;
```

```
    Label1: TLabel;
    Label4: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    LabelDownStreamDamLevelCaption: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    LabelUpstreamDamLevelMax: TLabel;
    LabelUpstreamDamLevel: TLabel;
    LabelUpstreamDamLevelMin: TLabel;
    LabelUpstreamSchedule: TLabel;
    LabelUpstreamControlPrivilege: TLabel;
    LabelDownStreamDamLevelMax: TLabel;
    LabelDownStreamDamLevel: TLabel;
    LabelDownStreamDamLevelMin: TLabel;
    LabelDownStreamControlPrivilege: TLabel;
    LabelDownStreamSchedule: TLabel;
    PanelOther: TPanel;
    Label21: TLabel;
    Label22: TLabel;
    LabelOtherSchedule: TLabel;
    LabelOtherStatus: TLabel;
    Timer1: TTimer;
    Label13: TLabel;
    LabelOtherControlMode: TLabel;
    PanelPumps: TPanel;
    Label14: TLabel;
    Label15: TLabel;
    LabelPumpsMaxNumberOfPumps: TLabel;
    LabelPumpsMinNumberOfPumps: TLabel;
    Label16: TLabel;
    LabelUpstreamUpperbound: TLabel;
    ButtonExpandForm: TPanel;
    ButtonScheduleHopUp: TPanel;
    ButtonSchedulehopDown: TPanel;
    ButtonResetController: TPanel;
    Label11: TLabel;
    Label7: TLabel;
    Label17: TLabel;
    Label18: TLabel;
    Panel1: TPanel;
    LabelLastActionString: TLabel;


    Constructor Create(AOwner:TComponent;AParent:TObject); Reintroduce;
    procedure FormShow(Sender: TObject);
    procedure FormHide(Sender: TObject);

    procedure Timer1Timer(Sender: TObject);
    procedure ButtonHideClick(Sender: TObject);

    procedure ButtonExpndFormClick(Sender: TObject);
    procedure ButtonScheduleHopUpClick(Sender: TObject);
    procedure ButtonSchedulehopDownClick(Sender: TObject);
    procedure ButtonResetControllerClick(Sender: TObject);

  private
    POwner : TComponent;
    PParent : TObject;

    Procedure UpdateAll;
        Procedure UpdateUpstreamControl;
        Procedure UpdateDownStreamControl;
        Procedure UpdateOther;
        Procedure UpdatePumps;
        Procedure UpdateLastActionString;
            Function LocalBoolToString(ABool : Boolean) : String;

    Procedure HideForm;
  public
```

```
end;
```

Code block 17 - Pumpgroup Viewer