# MMS Primer

All about the Modular Modeling System™

# Table of Contents

# 1. Introduction to MMS

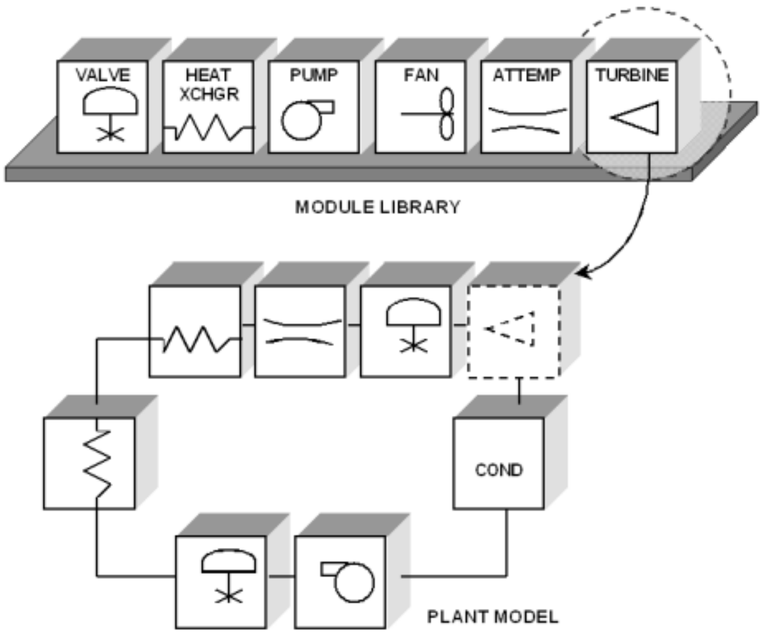
This chapter provides an introduction to the Modular Modeling System (MMS) through sections that describe MMS, an MMS model, MMS's purpose, its constituents, applicability, required modeler background, required equipment, languages used, and model types.

## What is MMS?

MMS is an acronym for Modular Modeling System. It is a Microsoft Windows based visual software system for modeling the dynamic characteristics of power plant systems and studying various design, performance, operation, training, interface, and maintenance aspects during the entire plant life cycle. Systems that can be modeled with MMS include, but are not limited to, thermal-hydraulic, nuclear, electrical, and control. MMS represents a powerful simulation environment featuring the Model Builder, component modules, online documentation, simulator building tools, module development capability, training, consultation, etc. MMS can be installed and used on a PC in conjunction with the Advanced Continuous Simulation Language (ACSL) [1], [2], [7] and Compaq[8] or Watcom FORTRAN compilers[3].

## What is an MMS Model?

An MMS model is a collection of modules graphically configured to represent the dynamic behavior of a power plant system. Section 1-1 in Part I of the MMS Basics Manual [6] further defines this concept.



The power plant modules represent the dynamic characteristics of power plant components based on first principles and include the following:

- **Fossil modules** (e.g., drum boiler, HRSG evaporator, superheater, economizer, gas turbine)
- **Nuclear modules** (e.g., reactor, pressurizer)
- **Balance of plant modules** (e.g., valve, pipe, steam turbine)

- **Electromechanical modules** (e.g., generator, motor, circuit breaker, transformer)
- **Control modules** (e.g., PID controller, low select, low/high alarm)

An MMS model is configured using the MMS Model Builder [4] in which modules can be selected online, placed on the model workspace, and connected to represent the model configuration. Each module typically has an input form in which physical and operating data of the component modeled are entered. An in-depth discussion of the background of MMS, derivation of first-principles equations, numerical integration, module connectivity, etc. is provided in the MMS Reference Manual [5].

The MMS model is further processed using the "Build Model" menu in the Model Builder to obtain an executable model that can be run to perform various analyses. Figure 8 shows the steps in the model building process (See also Section 1-1 of the MMS Reference Manual [5]). The executable model can also be run in conjunction with MMS Simulation Tools (Simulation Master, Graph Master, Action Center, and MMI Manager as described in Chapter 7) to perform applications, such as operator training and DCS checkout.

## Purpose of the MMS Model

An MMS model is used to do the following:

- **Study steady state and dynamic performance**
- **Change the parameters** (e.g., valve characteristics, controller gains) and forcing functions (e.g., valve position demand, controller set point) interactively to evaluate the responses
- **Perform linear analyses, such as stability analysis**
- **Study failure modes and effects**
- **Check out DCS configuration and I/O interface**
- **Verify system modifications**
- **Train operators**
- **Develop operating procedures**

## Structure of an MMS Model

An MMS model is programmed in ACSL, a high-level simulation language, and follows the ACSL model structure. It has several sections: Initial, Dynamic, Derivative, one or more Discrete sections, Terminal, and Subroutines [2]. The code generated from the graphical configuration in the MMS Model Builder is placed in the Derivative and/or Discrete sections. Subroutines may be included in the code or directly in the MMS project. (See Chapter 3 for details)

## Applicability

The MMS is applicable for modeling most power plant systems (nuclear, fossil, cogeneration, etc.) for dynamic analyses. The ACSL run-time environment provides powerful linear analysis capability.

## Modeler Background

For effective modeling, the person developing a model should be familiar with power plant systems, ACSL, FORTRAN, and the Windows operating system.

## Equipment Required

MMS can be installed on a variety of platforms. However, installation on a Windows XP/7/NT/2000 PC is the most common. The MMS software includes the MMS Model Builder, Component modules, and the MMS macro and FORTRAN libraries. ACSL is imbedded in the MMS Model Builder. Compaq or Watcom FORTRAN is required.

## Languages Used

MMS macros are written in ACSL. The ACSL translator converts the model program code into a FORTRAN program. Application-specific subroutines can be included in one of two ways: as source in the model epilog at the indicated place, or as either source, object, or library files in the model project settings. C routines can be included using inter-language capability.

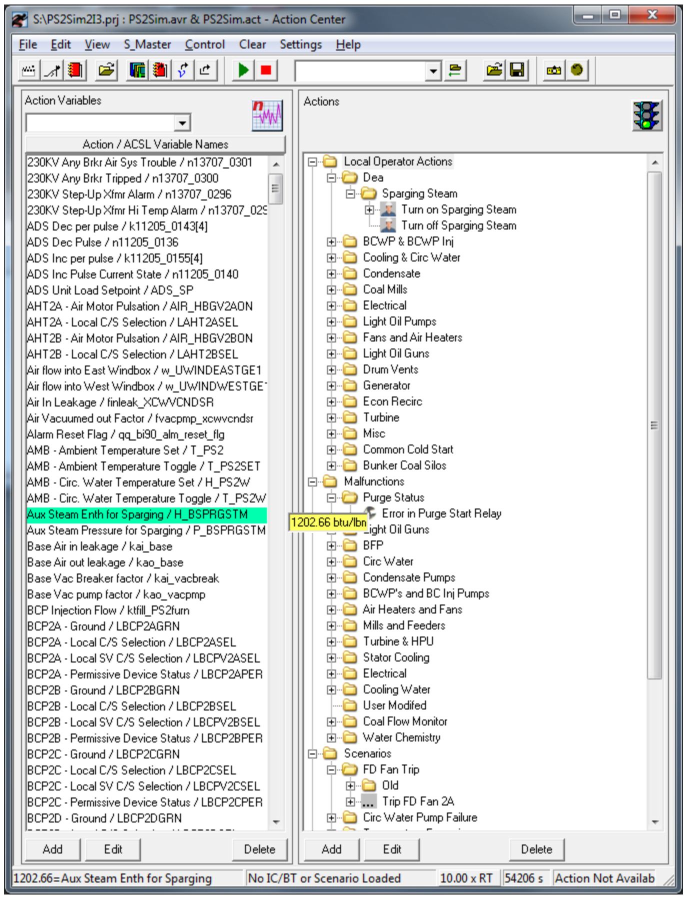
## Types of MMS Models

Depending on the application, many types of MMS models are used:

- **Real Time and Non-Real Time**
  Models used for studying control or operator interactions usually run in real time (i.e., same speed as the actual power plant system) and are built using the Real Time Capable (RTC) modules. The formulation of the RTC modules allows for the simulation of low flow and reverse flow conditions. An in-depth discussion of the basic theory of RTC modules, comprehensive description of each module, a list of MMS modules, ports, and module selection guide, gas and water/steam property subroutines, and a list of MMS error and warning messages is provided in the MMS Basics Manual [6].

- **With or Without Simulation Tools**
  Models may run in the traditional ACSL run-time environment in which the model output is in the form of selected variable values displayed on the screen periodically and user interaction is at pre-specified times while the model is stopped for specifying the interactions and plotting. Models can also be developed such that user interactions with the model are accomplished without stopping the model, through a Man-Machine Interface [MMI, also known as Human-Machine Interface (HMI) or Operator Interface (OI)]. The MMI can exist on the same or separate networked PCs. The contents of the MMI screen can be developed using Visual Basic. nView is an example of such MMI. nView is a very powerful simple interface with the model through variables that can be selected, monitored and set. Various MMI applications are interfaced with the model through MMI Manager. Interaction may also be provided from the Action Center, and Graph Master (G_Master). The Action Center may be used to initiate normal actions (e.g., opening a valve), malfunctions, scenarios, or standard pre-built tests. The G_Master application is used for monitoring trends of model variables. In addition, interactions can be provided from external devices such as hard panels, DCS, PLC using software device drivers, APIs, or I/O devices. In all such interactions, the interface with the model is through Simulation Master (S_Master). See Chapter 7 for more information on Simulation Tools.

Figures 1-5 present the MMS Action Center, G_Master, S_Master, nView, and MMI Manager applications.


S:\PS2Sim2I3.prj : PS2Sim.avr & PS2Sim.act - Action Center

Action Variables

Action / ACSL Variable Names

230KV Any Brkr Air Sys Trouble / n13707_0301
230KV Any Brkr Tripped / n13707_0300
230KV Step-Up Xfmr Alarm / n13707_0296
230KV Step-Up Xfmr Hi Temp Alarm / n13707_029
ADS Dec per pulse / k11205_0143[4]
ADS Dec Pulse / n11205_0136
ADS Inc per pulse / k11205_0155[4]
ADS Inc Pulse Current State / n11205_0140
ADS Unit Load Setpoint / ADS_SP
AHT2A - Air Motor Pulsation / AIR_HBGV2AON
AHT2A - Local C/S Selection / LAHT2ASEL
AHT2B - Air Motor Pulsation / AIR_HBGV2BON
AHT2B - Local C/S Selection / LAHT2BSEL
Air flow into East Windbox / w_UWINDEASTGE1
Air flow into West Windbox / w_UWINDWESTGE
Air In Leakage / finleak_XCWVCNDSR
Air Vacuumed out Factor / fvacpmp_xcwvcndsr
Alarm Reset Flag / qq_bi90_alm_reset_flg
AMB - Ambient Temperature Set / T_PS2
AMB - Ambient Temperature Toggle / T_PS2SET
AMB - Circ. Water Temperature Set / H_PS2W
AMB - Circ. Water Temperature Toggle / T_PS2W
Aux Steam Enth for Sparging / H_BSPRGSTM
Aux Steam Pressure for Sparging / P_BSPRGSTM
Base Air in leakage / kai_base
Base Air out leakage / kao_base
Base Vac Breaker factor / kai_vacbreak
Base Vac pump factor / kao_vacpmp
BCP Injection Flow / ktfill_PS2furn
BCP2A - Ground / LBCP2AGRN
BCP2A - Local C/S Selection / LBCP2ASEL
BCP2A - Local SV C/S Selection / LBCPV2ASEL
BCP2A - Permissive Device Status / LBCP2APER
BCP2B - Ground / LBCP2BGRN
BCP2B - Local C/S Selection / LBCP2BSEL
BCP2B - Local SV C/S Selection / LBCPV2BSEL
BCP2B - Permissive Device Status / LBCP2BPER
BCP2C - Ground / LBCP2CGRN
BCP2C - Local C/S Selection / LBCP2CSEL
BCP2C - Local SV C/S Selection / LBCPV2CSEL
BCP2C - Permissive Device Status / LBCP2CPER
BCP2D - Ground / LBCP2DGRN

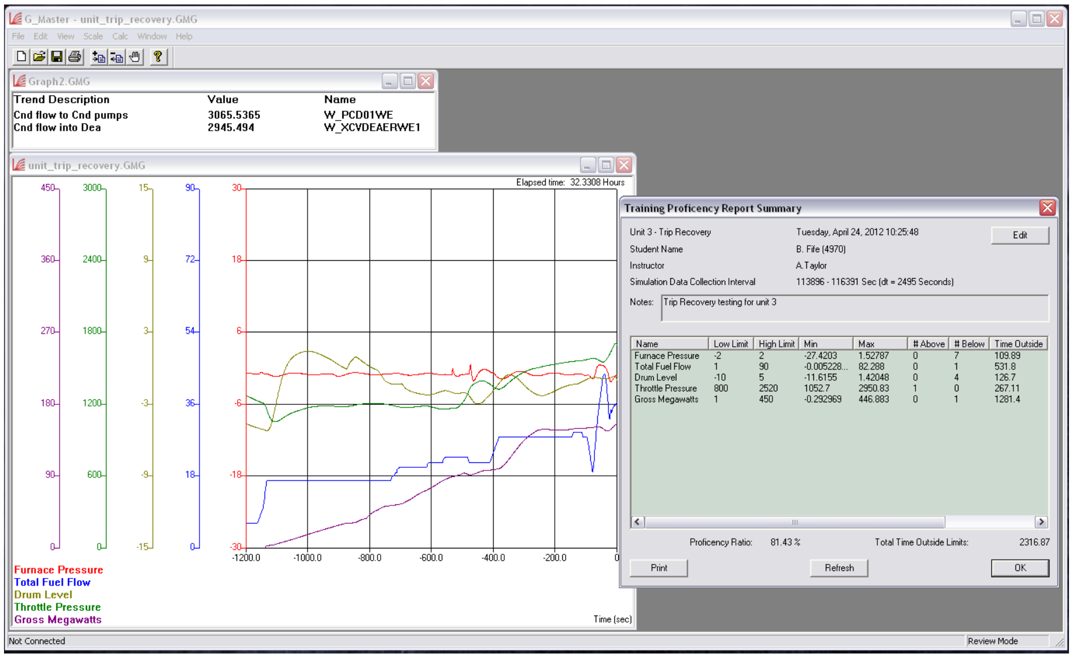
1202.66 btu/lbm

Actions

Local Operator Actions
Dea
Sparging Steam
Turn on Sparging Steam
Turn off Sparging Steam
BCWP & BCWP Inj
Cooling & Circ Water
Condensate
Coal Mills
Electrical
Light Oil Pumps
Fans and Air Heaters
Light Oil Guns
Drum Vents
Generator
Econ Recirc
Turbine
Misc
Common Cold Start
Bunker Coal Silos
Malfunctions
Purge Status
Error in Purge Start Relay
Light Oil Guns
BFP
Circ Water
Condensate Pumps
BCWP's and BC Inj Pumps
Air Heaters and Fans
Mills and Feeders
Turbine & HPU
Stator Cooling
Electrical
Cooling Water
User Modifed
Coal Flow Monitor
Water Chemistry
Scenarios
FD Fan Trip
Old
Trip FD Fan 2A
Circ Water Pump Failure

1202.66=Aux Steam Enth for Sparging | No IC/BT or Scenario Loaded | 10.00 x RT | 54206 s | Action Not Availab

**Figure 1:** MMS Action Center


G_Master - unit_trip_recovery.GMG

Graph2.GMG

| Trend Description | Value | Name |
|---|---|---|
| Cnd flow to Cnd pumps | 3065.5365 | W_PCD01WE |
| Cnd flow into Dea | 2945.494 | W_XCVDEAERWE1 |

unit_trip_recovery.GMG

Elapsed time: 32.3308 Hours

Training Proficency Report Summary

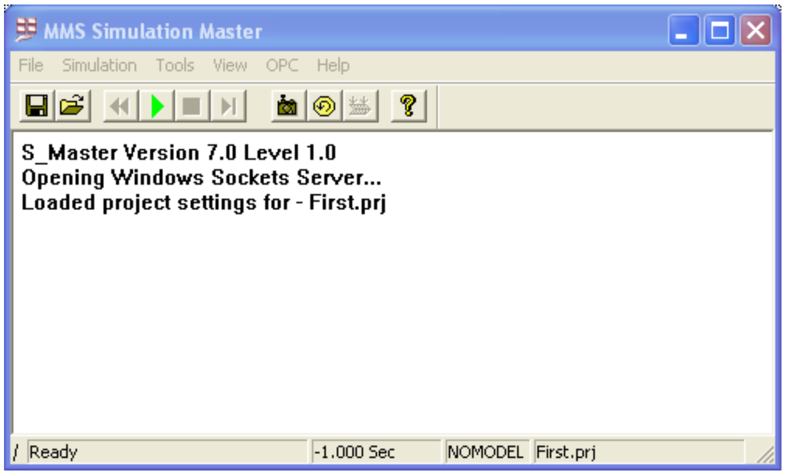
Unit 3 - Trip Recovery | Tuesday, April 24, 2012 10:25:48
Student Name | B. Fife (4970)
Instructor | A.Taylor
Simulation Data Collection Interval | 113896 - 116391 Sec (dt = 2495 Seconds)
Notes: Trip Recovery testing for unit 3

| Name | Low Limit | High Limit | Min | Max | # Above | # Below | Time Outside |
|---|---|---|---|---|---|---|---|
| Furnace Pressure | -2 | 2 | -27.4203 | 1.52787 | 0 | 7 | 109.89 |
| Total Fuel Flow | 1 | 90 | -0.005228... | 82.288 | 0 | 1 | 531.8 |
| Drum Level | -10 | 5 | -11.6155 | 1.42048 | 0 | 4 | 126.7 |
| Throttle Pressure | 800 | 2520 | 1052.7 | 2950.83 | 1 | 0 | 267.11 |
| Gross Megawatts | 1 | 450 | -0.292969 | 446.883 | 0 | 1 | 1281.4 |

Proficency Ratio: 81.43 %  Total Time Outside Limits: 2316.87

Not Connected | Review Mode

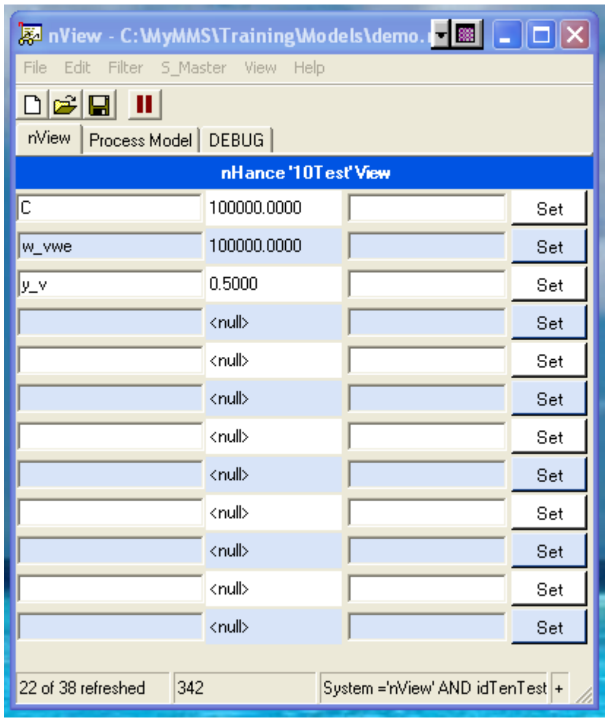
**Figure 2:** MMS Graph Master (G_Master)



```
S_Master Version 7.0 Level 1.0
Opening Windows Sockets Server...
Loaded project settings for - First.prj
```

Ready | -1.000 Sec | NOMODEL | First.prj

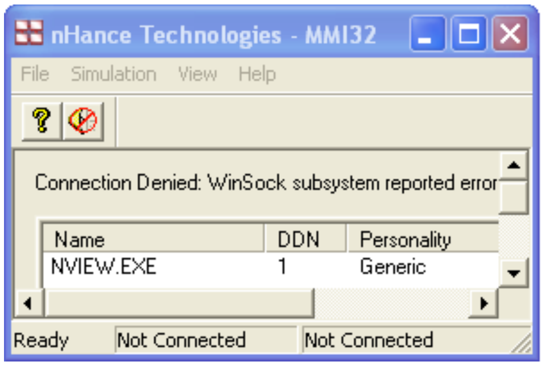
**Figure 3:** MMS Simulation Master (S_Master)


nView - C:\MyMMS\Training\Models\demo.
File Edit Filter S_Master View Help
nView
Process Model
DEBUG
nHance '10Test' View

| | | | |
|---|---|---|---|
| C | 100000.0000 | | Set |
| w_vwe | 100000.0000 | | Set |
| y_v | 0.5000 | | Set |
| | <null> | | Set |
| | <null> | | Set |
| | <null> | | Set |
| | <null> | | Set |
| | <null> | | Set |
| | <null> | | Set |
| | <null> | | Set |
| | <null> | | Set |
| | <null> | | Set |

22 of 38 refreshed | 342 | System ='nView' AND idTenTest +

**Figure 4:** MMS nView



Connection Denied: WinSock subsystem reported error

| Name | DDN | Personality |
|---|---|---|
| NVIEW.EXE | 1 | Generic |

Ready | Not Connected | Not Connected

**Figure 5:** MMI Manager

# 2. Overview of Model Building

This chapter covers the following:

1. **The steps used in building a model**
2. **The MMS Model Builder**
3. **FORTRAN compilers for MMS**
4. **Large-model tools (PC Split Program, Debugger)**
5. **Interfacing with the nView, Action Center, G_Master, MMI, and external devices through S_Master**
6. **Custom ACSL macros**
7. **MMS modules created by using MMS CompGen™.**

## Steps in Building a Model

Assuming that you have successfully installed and tested the MMS software, the procedure for building an MMS model is as follows:

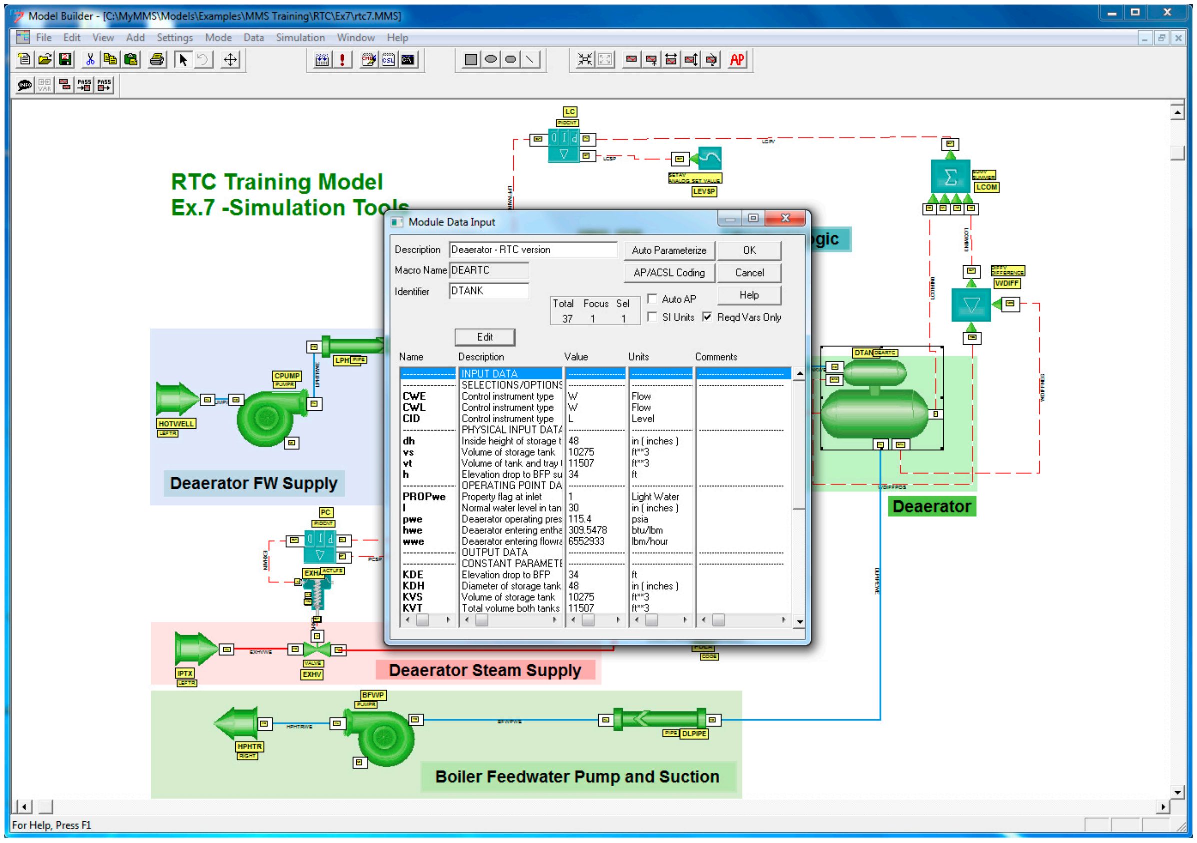
- **Understand the functionality of the system to be modeled and gather information.** This is a very important, although often procrastinated, step; the efficiency and success of the simulation project heavily depend on it.

  The information to be gathered includes items such as the following:

  - System description, function, operation
  - Scope of model (phenomena and components to be modeled, boundary conditions, model type, accuracy, testing, and use)
  - P&I drawings
  - Electrical one-line drawings
  - Control logic drawings
  - Validated heat balance data
  - Component data (e.g., valve type and capacity, pump head-flow characteristic, motor no-load and full load performance data, boiler heat transfer surface arrangement and geometry, safety valve settings)
  - Controller setpoints and gains

- **Configure the model using the MMS Model Builder.** **Review and change project settings ("Settings\Project Settings" menu), and select the appropriate prolog and epilog pair from the "Simulation\Code Generation Settings\Prolog/Epilog Code" menu, and edit as necessary. Select the modules, place them, interconnect them, enter the Auto-Parameterization (AP) input data, and autoparameterize. Finally, save the model (an .mms file). You may also save the**

**model configuration diagram as a metafile (.emf) for model documentation by clicking on "Files\Metafile Export" menu.**


Model Builder - [C:\MyMMS\Models\Examples\MMS Training\RTC\Ex7\rtc7.MMS]
File Edit View Add Settings Mode Data Simulation Window Help
RTC Training Model
Ex.7 -Simulation Tools
Deaerator FW Supply
Deaerator Steam Supply
Boiler Feedwater Pump and Suction
Deaerator
LC
LEVSP
LCOM
VWDIFF
HOTWELL
CPUMP
LPH
PC
EXHV
IPTX
HPHTR
BFWP
DLPIPE
For Help, Press F1

Module Data Input

Description: Deaerator - RTC version
Macro Name: DEARTC
Identifier: DTANK

| Total | Focus | Sel |
|---|---|---|
| 37 | 1 | 1 |

☐ Auto AP
☐ SI Units ☑ Reqd Vars Only

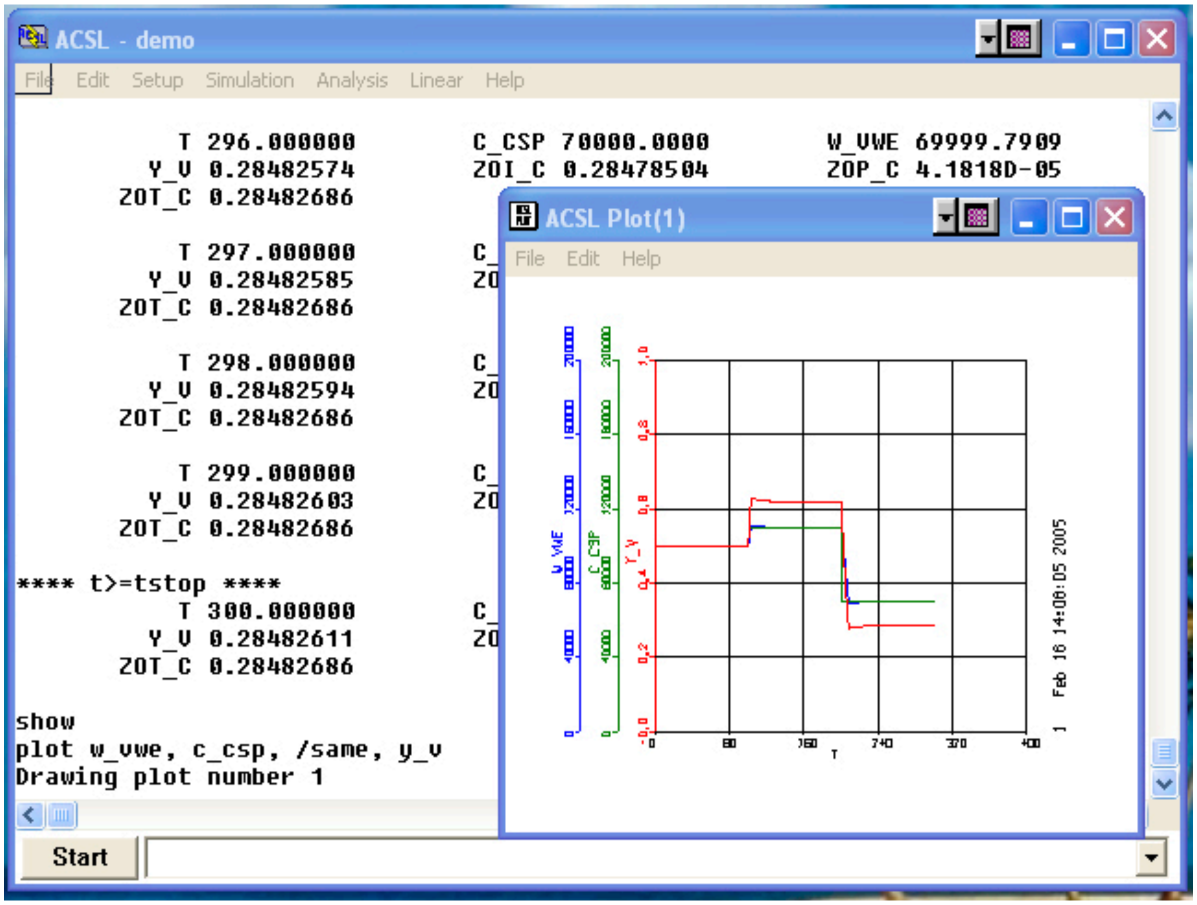
Auto Parameterize · OK · AP/ACSL Coding · Cancel · Help · Edit

| Name | Description | Value | Units | Comments |
|---|---|---|---|---|
| --------------- | INPUT DATA | --------------- | --------------- | --------------- |
| --------------- | SELECTIONS/OPTIONS | --------------- | --------------- | --------------- |
| CWE | Control instrument type | W | Flow | |
| CWL | Control instrument type | W | Flow | |
| CID | Control instrument type | L | Level | |
| --------------- | PHYSICAL INPUT DATA | --------------- | --------------- | --------------- |
| dh | Inside height of storage t | 48 | in ( inches ) | |
| vs | Volume of storage tank | 10275 | ft**3 | |
| vt | Volume of tank and tray | 11507 | ft**3 | |
| h | Elevation drop to BFP su | 34 | ft | |
| --------------- | OPERATING POINT DA | --------------- | --------------- | --------------- |
| PROPwe | Property flag at inlet | 1 | Light Water | |
| l | Normal water level in tan | 30 | in ( inches ) | |
| pwe | Deaerator operating pres | 115.4 | psia | |
| hwe | Deaerator entering entha | 309.5478 | btu/lbm | |
| wwe | Deaerator entering flowra | 6552933 | lbm/hour | |
| --------------- | OUTPUT DATA | --------------- | --------------- | --------------- |
| --------------- | CONSTANT PARAMETE | --------------- | --------------- | --------------- |
| KDE | Elevation drop to BFP | 34 | ft | |
| KDH | Diameter of storage tank | 48 | in ( inches ) | |
| KVS | Volume of storage tank | 10275 | ft**3 | |
| KVT | Total volume both tanks | 11507 | ft**3 | |

**Figure 6:** MMS Model Builder featuring the AP window

- **Build an executable model.**
  Specify the project options (i.e., user-defined fortran, object, and library files to be included in the model) from the "Settings \ Project Settings" menu. Click on the "Build Model" button. The Model Builder will translate, compile, and link the model files and libraries to generate the executable model. Unless it is a simple model and you have not added any custom coding, it is quite unusual that the model will run perfectly the first time through. There may be translation errors and/or compile errors. If so, edit appropriate files to eliminate these until the model builds satisfactorily. Edit the default command file automatically generated during the first model save to include commonly used model startup commands (e.g., s NRWITG=.t. ; output t, xyz,… ) and command macros (called PROCEDs) [2].

- **Run the model.**
  Run the model by clicking the "Execute Model" button. The ACSL run-time window shown in Figure 7 will appear. The commands in the command file will be executed. Type commands to test the model. If you encounter any run-time errors, resolve them with or without the debugger. Once the model runs satisfactorily, it can also be run in the SimTools environment if the MMS – RTC w/ Simulator Link" prolog/epilog was chosen during model building.



```
        T 296.000000         C_CSP 70000.0000        W_VWE 69999.7909
      Y_V 0.28482574         ZOI_C 0.28478504        ZOP_C 4.1818D-05
    ZOT_C 0.28482686

        T 297.000000         C_
      Y_V 0.28482585         ZO
    ZOT_C 0.28482686

        T 298.000000         C_
      Y_V 0.28482594         ZO
    ZOT_C 0.28482686

        T 299.000000         C_
      Y_V 0.28482603         ZO
    ZOT_C 0.28482686

**** t>=tstop ****
        T 300.000000         C_
      Y_V 0.28482611         ZO
    ZOT_C 0.28482686

show
plot w_vwe, c_csp, /same, y_v
Drawing plot number 1
```

**Figure 7:** ACSL Executive

## MMS Model Builder

### Model configuration

The MMS Model Builder is a very powerful visual environment for building models graphically by placing and linking icons on the screen. Each icon represents an MMS module or component. Included in the MMS Model Builder are many modules that constitute the building blocks for model development. The location of modules, help files, text editor, etc., are specified in the "Settings\Directories/Paths" menu.

### Modules

Each module consists of four elements: an Icon, an Auto Parameterization (AP) form and Dynamic Link Library (DLL), an ACSL template, and an online help file. The icon is a graphic representation of the component or function and includes ports for connecting the module to other modules. The AP form is a table for entering the geometric data, response characteristics, and operating data. The DLL takes the input data and calculates the values of parameters appearing in the model. The ACSL template is the code for the component's mathematical representation that gets included in the model source file (.csl) during code generation after the substitutions and assertions in the template are carried out; to see the code, click the "Expand" button in the "AP/ACSL Coding" window. Finally the online help file facilitates understanding of the module (function, ports, invocation, parameters and variables, theory, references, etc.).

The Comments column in the Data Input form allows the user to add notations about the source of data, etc. The toggling of units in the AP form is a powerful convenience to enter or view data in either US or SI units. The copy, paste, and automatic module Identification Tag (ID) generation capability are useful tools for efficiently developing a model. For example, a group of components that appear several times in the system can be developed and tested as a template, replicated to make similar groups, and then edited as necessary. All internal connections in the group are preserved in copying, thus minimizing configuration time. When developing the group, annotations can be added to indicate measurement points, subsystem names, etc. The ACSL template of a module may also be modified if necessary ***(caution: in doing so, the user takes responsibility for proper functioning of the module and the effect on other modules).***

If the system being modeled has features that are not modeled by the MMS modules, the user can use the CODE module and provide ACSL coding to model such features. A code block is a special module with universal port types that can be connected to any other port (and therefore module). For example, if the measurement required by the control logic is not available as a port on a component module, a CODE module can be used to obtain the desired measurement and pass it on to the control logic.

## Model building

When you build the model by clicking the "Build Model" button, the model source file is generated, translated, compiled, and linked to create the executable model. The code generated from the Model Builder's graphic representation of the process is placed in between the prolog and epilog sections of the model. Effectively, the code gets placed in the Derivative section. The Derivative section includes continuous-time derivative calculations, such as boiler dynamic states and analog controllers. The prolog and epilog sections are generic in nature. Several standard versions of these sections are included in the MMS Model Builder (e.g., MMS only, MMS-RTC, and MMS-RTC with simulator link). The user can select from the pre-defined Prolog/Epilog combination, and then edit if necessary. A log of model development can be maintained in the prolog in the form of comments. Custom macros can be included in the prolog, while custom subroutines can be included in the epilog.

The .csl file generated in the MMS Model Builder is processed by the Model Builder to create the executable model (.prx file). The process includes translating to Fortran, compiling along with specified subroutines, linking with specified .obj and library files. The process uses the MMS macro library, and the MMS FORTRAN library.

When a model is saved for the first time, a default command file for the model is generated, which can be edited to include commands for displaying and plotting a list of model variables, and procedurals (command macros) for carrying out transients. A convenient method is to use the Setup/Output menu in the ACSL runtime window. This menu shows the names of all model variables from which the user can select.

## Model execution

Once the executable model is successfully generated and it executes satisfactorily, it can be run to a steady state (marked by magnitudes of derivatives of dynamic states relative to the magnitudes of states being less than some low value), and eigenvalues of the system can be determined at that steady state by executing "Analyze /eigen" command. Eigenvalues are inverse of time constants in the complex variable (real/imaginary) domain. A linearized model may also be determined for further analysis/design by using the "Analyze /jacobian" command.

If SimTools Options are selected in the "Settings\Project Settings\Project Options" menu and the "RTC w/ Simulator Link" prolog/epilog is selected, the model can also be executed from Simulation Master with "on the fly" interactions from G_Master, nView or other MMI.

## FORTRAN Compilers

The 32-bit FORTRAN compilers supported by MMS on the Windows XP/NT/2000 platform are:

- **Compaq FORTRAN versions 6.6**
- **Open Watcom**
- **Watcom FORTRAN versions 10.6 and 11.0**

## FORTRAN Debugger

The FORTRAN environment debugger is of great help in debugging a model, particularly a large one. However, proficiencies in FORTRAN and the use of debugging techniques are required to fully take advantage of this tool. The "Settings\Project Settings\Project Options" menu includes the "Generate Debug Code" option to facilitate building model suitable for debugger use.

## PC Split

To facilitate building a large model in a time-efficient manner, MMS includes the " Use PC Split" option in the "Settings \ Project Settings \ Project Options" menu to split the FORTRAN code into several subroutines. This procedure is required for an MMS model with the "RTC w' Simulator Link" prolog/epilog to run in the simulator environment in conjunction with Sim Tools (e.g., S_Master, nView, MMI Manager, Action Center, etc. See Chapter 7 on Simulation Tools)

The use of PC Split allows large MMS models to be compiled in significantly less CPU time. In addition to splitting up the FORTRAN code, PC Split analyzes the split segments and produces files that are later used to build a database (.sdb file) of the variables in the model.

## Model Building for Use with SimTools

You can build an MMS model for use with SimTools by clicking the "Set Standard SimTools Options" button in the "Settings \ Project Settings \ Project Options" window. Such a model permits on-the-fly interactions with the user as in a training simulator.

The Standard SimTools Options include the PC Split procedure and generation of Custom Include files for each split piece. The splitting and custom include file generation process produces a series of files that, along with the MAP file produced by the Fortran linker, are used to generate the S_Master database file (.SDB file). The S_Master database file is a listing of all variables in the simulation and their locations. S_Master loads the .SDB file into RAM to access the model variables and service client requests. Since the .SDB file is not required for execution of the model in the ACSL runtime environment, model testing can also be carried out by clicking on the "Execute Model" [!] button in the Model Builder.

Figure 8 describes the MMS model building process. The left-hand side of Figure 8 indicates the steps in building an MMS model for ACSL runtime environment, while the right-hand side indicates the additional steps for SimTools' runtime environment.

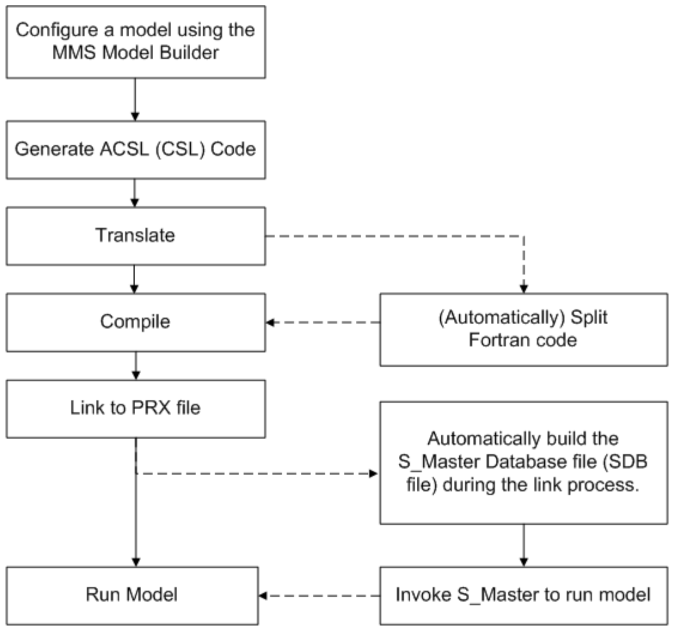

Configure a model using the MMS Model Builder
Generate ACSL (CSL) Code
Translate
Compile
(Automatically) Split Fortran code
Link to PRX file
Automatically build the S_Master Database file (SDB file) during the link process.
Run Model
Invoke S_Master to run model

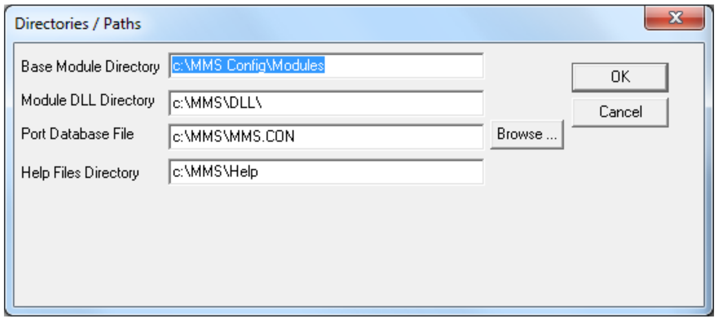
**Figure 8:** Model building process

## Custom ACSL Macros

Although most of the macros required for power plant related modeling and invoked in the MMS module ACSL templates are provided in the MMS macro library, it is quite possible that a custom macro may be required in an application, e.g., to make a template for code that repeats several times in that application under different names. A macro can be developed using the ACSL macro language. (See chapter on Macro Language in Reference 2). To provide access to a custom macro in a model, "include" the macro file or macro text in the prolog at the indicated location.

## MMS Modules (MMS CompGen)

Although most of the modules required for power plant related modeling are provided in the MMS Model Builder, it is quite possible that a custom module may be required in an application. MMS CompGen is a powerful visual environment for developing a user specific module. Different aspects of module development require different developer backgrounds. To develop an icon with ports for interconnection, familiarity with the different types of ports and the types of variables "riding" over the port stream is necessary. To use the Auto Parameter (AP) variables in the ACSL template, familiarity with the substitutions and assertions provided in MMS CompGen is required. To program the AP calculations in C++ in the AP code window, familiarity with C++ language is necessary. The AP code is compiled to produce a .DLL file. To specify path for custom modules in the Model Builder, add the custom module and .DLL directory in the "Settings \ Directories/Paths" menu as follows:



Directories / Paths

Base Module Directory: c:\MMS Config\Modules

Module DLL Directory: c:\MMS\DLL\

Port Database File: c:\MMS\MMS.CON

Help Files Directory: c:\MMS\Help

**Figure 9:** Directories/Paths

MMS CompGen [4] online help includes the necessary information for creating or modifying a module. Advanced CompGen help is also available online in "MMS Library" [9]. The following figure depicts the MMS CompGen™ application.

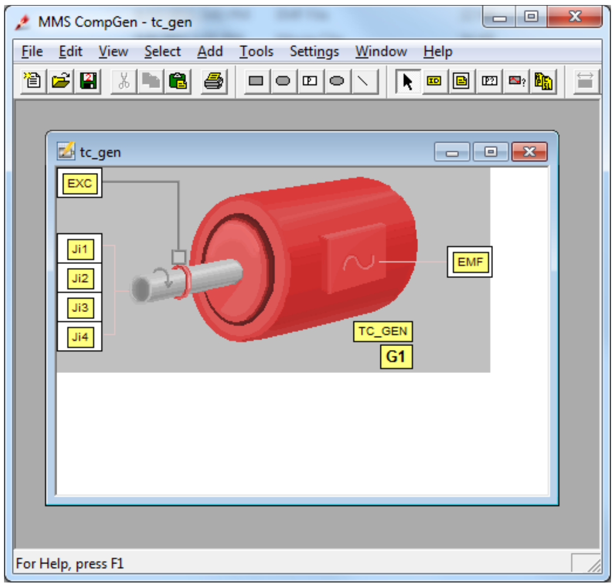


**Figure 10:** MMS CompGen

# 3. The MMS Model Builder

The MMS Model Builder contains a number of features designed to increase the user's efficiency in building a simulation. This chapter describes the MMS Model Builder and touches on some of its features. In addition, this chapter presents a few efficiency tips, and outlines a few modeling techniques, such as specifying boundary conditions, combining sub-models, and data collection.

## Module Library Contents

The modules in the Module Library are classified in several functional groups, e.g., RTC, Non-RTC, Control, ElecMech (Electromechanical), Common, and Trial. A list of the MMS modules is provided in Appendix A of Reference [6]. A description of the MMS modules is provided in Part II of Reference [6]. All modules include online help featuring the macro invocation, boundary conditions, theory, etc.

**The RTC Group** includes real-time capable modules that use an iterative pressure/flow solution and are required but not limited to applications that require real-time response (e.g., training simulator). A further classification is provided in the group as: BOP, Boundary, Fossil, and FossilAG. The FossilAG modules include air/gas side pressure calculation, whereas the Fossil modules do not. LEFTR and RIGHT modules represent the pressure source and sink boundaries of a hydraulic network in an RTC model. Module UJUNC (Universal Junction) is used to model flow splits and junctions.

**The Non-RTC Group** includes modules that are the forerunners of the RTC group. In these modules, the pressures at key locations are represented as ACSL dynamic states. A further classification is provided in the group as: AirGas, BOP, Boundary, Fossil, and Nuclear. The AirGas modules include air/gas side pressures as dynamic states, whereas the Fossil modules do not.

**The Control Group** includes modules that represent SAMA function blocks (e.g., PID, signal monitor [hi/lo alarm], and function generator).

**The ElecMech Group** includes electromechanical modules, such as electric generators, induction motor, transformer, circuit breaker, busses, etc., which can be used to model an electrical distribution system. The procedure for building an electrical distribution system model is described in MMS Library [9].

**The Common Group** includes boundary conditions, offpage connectors, etc. that are common to the RTC, Non-RTC, and Control groups. It also includes a special module called CODE (or code block), which has universal ports that can be connected to any other port and where the user can define custom code in ACSL.

**The Trial Group** includes new modules that are undergoing actual in-use testing and may eventually be transferred to one of the other groups. These modules don't have complete online help. The user should use caution when using these modules and may need support from MMS technical staff.

## Units

The default units are US Customary Units. SI units can be selected globally by using the "Settings/SI Units" menu. For entering or seeing the values in one or the other units, the SI Units box on individual module data input (AP) window may be toggled.

## Module Ports and Interconnectibility, Sizing, Orientation, Interconnection Options

Each module has one or more ports for interconnecting with other modules. There are several common port names and port types for different flow/signal streams, e.g., WE (water entering) of resistive-in type, and SL (steam leaving) of storage-out type. The port type can be determined graphically by using the INFO bubble button or from the help file. Port types that satisfy certain connection rules can be connected together (e.g., resistive-in with storage-out). When a port is double-clicked for interconnecting with others, all connectible ports in the model turn blue, thus prompting the user. A list of port types and allowed port type connectibility is provided in Figure 11. Appendix A of Reference [6] includes a table of port types and modules that have at least one port to which the port type can be connected. To see the port names conveniently, the "Settings\Port ID Font" menu may be used to select a larger and/or bold font. Also it may be helpful to turn off the module ID, etc., by using the "View\Module ID" menu.

There are several classes of ports:  flow stream, signal, electromechanical, and universal. The flow stream ports include resistive-in and -out, storage-in and -out, etc.  Signal ports include analog-in and -out, digital-in and -out, logical-in and –out (i.e., True/False), control-in and -out, etc. Electromechanical ports include power-in and -out, speed-in or -out, electrical-in and -out.  Universal ports include universal-in and -out and can connect with any -out and –in ports, respectively.  In general, "in" ports connect with "out" ports and vice versa.  Before RTC modules were added in MMS, all flow stream connections had to follow a connection rule for the sake of modularity.  There were two types of modules, resistive and storage.  The resistive modules (e.g., valve, pump, and pipe) required pressure at both ends to calculate flow.  The storage modules (e.g., deaerator, steam drum in a boiler) required flow in and out to calculate pressure.  The rule required that storage modules be connected to resistive and vice versa for proper pressure flow calculation, i.e., resistive ports could not be cascaded together without intervening storage ports and vice versa .  Connective modules (called connective nodes) for incompressible and compressible flow were introduced at that time to facilitate interconnectibility.  However the connective nodes introduced very large eigenvalues (i.e., fast dynamics) and required small time steps during fast changes in the modeled system.  This made them unsuitable for real-time applications, such as training simulators.  The RTC modules overcame this by using an iterative pressure-flow solution method.  Section 3.2, Part I of Reference [6] includes a discussion of connection rules for various types of modules.

The module icons can be sized to suit the model scope.  A part of the model can also be zoomed (View/Zoom menu) for ease of seeing details and making connections.  Module icons can also be turned to orient them suitably ("Edit\Module" menu or 3 buttons on the tool bar ).

The interconnection attributes include color, type, and thickness of connecting lines which can be used to color-code the streams (e.g., blue for water, yellow for steam, dotted red for air/gas, chain-dotted purple for control signals, etc.).

| Port Type | Connection Name | Allowable Connections | | | | | |
|---|---|---|---|---|---|---|---|
| AGBin | Air/Gas Dynamics Boundary In | AGRin | AGSin | UO | | | |
| AGBout | Air/Gas Dynamics Boundary Out | AGRout | AGSout | UI | | | |
| AGRin | Air/Gas Resistive Inlet | AGBin | AGSout | UO | | | |
| AGRout | Air/Gas Resistive Outlet | AGBout | AGSin | UI | | | |
| AGSin | Air/Gas Storage In | AGBin | AGRout | UO | | | |
| AGSout | Air/Gas Storage Out | AGBout | AGRin | UI | | | |
| AI | Analog In | AO | Cout | MO | UO | | |
| Ain | Air In | Aout | Gout | UO | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| AO | Analog Out | AI | Cin | UI | | | |
| Aout | Air Out | Ain | Gin | UI | | | |
| CI | Compressible In | CNO | MNO | UO | | | |
| Cin | Control In | AO | Cin | Cout | DO | MO | UO |
| CNO | Compressible Node Out | CI | UI | | | | |
| CO | Compressible Out | NI | UI | | | | |
| Cout | Control Out | AI | Cin | UI | | | |
| DI | Digital Input | DO | UO | | | | |
| DO | Digital Output | DI | UI | | | | |
| Ein | Electrical In | Eout | UO | | | | |
| Eout | Electrical Out | Ein | UI | | | | |
| Fin | Fuel In | Fout | UO | | | | |
| Fout | Fuel Out | Fin | UI | | | | |
| GCVin | Gas Composition Vector In | GCVout | UO | | | | |
| GCVout | Gas Composition Vector Out | GCVin | UI | | | | |
| Gin | Gas In | Aout | Gout | UO | | | |
| Gout | Gas Out | Ain | Gin | UI | | | |
| JIN | Power in/Speed Out | JOUT | UO | | | | |
| JOUT | Power Out/Speed In | JIN | UI | | | | |
| KCVin | Kerosene Composition Vector In | KCVout | UO | | | | |
| KCVout | Kerosene Composition Vector Out | KCVin | UI | | | | |
| MMSBin | MMS Boundary In | RIN | SIN | UO | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| MMSBot | MMS Boundary Out | ROUT | SOUT | UI | | | |
| MNO | SPLITM output to CI or PI | CI | PI | UI | | | |
| MO | Motor Out | AI | Cin | UI | | | |
| NI | Pressure Node Inlet | CO | PO | SWO | UO | | |
| NO | Pressure Node Out | PI | UI | | | | |
| PI | Pipe Inlet | MNO | NO | PO | UO | | |
| PO | Pipe Outlet | NI | PI | UI | | | |
| Qin | Energy Stream In | Qout | UO | | | | |
| Qout | Energy Stream Out | Qin | UI | | | | |
| RIN | Resistive In | MMSBin | SOUT | UO | | | |
| ROUT | Resistive Out | MMSBot | SIN | UI | | | |
| SIN | Storage In | MMSBin | ROUT | UO | | | |
| SOUT | Storage Out | MMSBot | RIN | UI | | | |
| SWO | SWH Outlet Port | NI | UI | | | | |
| TFI | True/False In | TFO | UO | | | | |
| TFO | True/False Out | TFI | UI | | | | |
| TVI | TURBV Parameters In | TVO | UO | | | | |
| TVO | TURBV Parameters Out | TVI | UI | | | | |
| UI | Universal In | Connects to all out ports | | | | | |
| UO | Universal Out | Connects to all In ports | | | | | |

**Figure 11: Port types and allowed connections**

## Sub-models with Off-Page Connectors

Large models can be divided into smaller sub-models, each of which can be contained in a separate MMS file.  These sub-models can then be connected together using off-page connectors.  This feature is useful for building and testing sub-models independently using boundary conditions.  Later, the boundary conditions can be replaced with off-page connectors to build the integrated model. Care must be taken when using off-page connectors, however, to make sure module ID’s are not repeated in the different sub-models and that all off-page connector streams contain matching ID’s.

To build a large model, the submodels should have the "Full Model" option unchecked in the "Simulation \ Code Generation Settings \ Options" menu and should be processed using "Simulation \ Generate csl" menu.  These sub csl files should be included in the Derivative section (for continuous processes) or in a Discrete section (for discrete processes) of the parent model by Include statements at the end of prolog or beginning of epilog, or at the indicated location in the epilog, respectively.

## Annotations

Annotations can be used to add notes, captions, subsystem names, etc. on the model graphic. Sensors can also be shown using circle, line, and symbol annotation (e.g., P for pressure).

## ID Incrementing

If the model includes identical or similar subsystems, one such system can be modeled using an appropriate incrementation scheme (Identifier and Inserts boxes in AP form).  After this system is debugged, it can be copied and pasted to add other similar subsystems; the ID is changed automatically by the MMS Model Builder as per the incrementation scheme.  The replicated subsystem models can then be edited.

## AP and ACSL Code Template

Most modules have an Auto Parameterization (AP) form and Auto Parameterization code, while all modules have an ACSL code template.  The user has to provide geometrical data, characteristics, and operating data in the input part of the AP form.  The AP code, if it exists, calculates the values of constant parameters and initial conditions that are inserted in the output part of the AP form and the ACSL code template.  The ACSL code template may include the macro invocation for the appropriate macro.

It is a good practice to expand the ACSL template (EXPAND/TEMPLATE button in the "ACSL Image" option in "AP/ACSL Coding" window) to check the code that will be included in the model source file.

## Prolog and Epilog

The model source file generated by the MMS Model Builder has three parts: prolog, contents of the Derivative block of the model, and epilog.  The prolog includes the front part of the model structure, while the epilog includes the rear part.  The contents of the Derivative block are generated from the graphical model.  The prolog and epilog can be selected from the standard versions provided in the MMS Model Builder ("Simulation \ Code Generation Settings \ Prolog/Epilog Code" menu).  The user can edit these.

The prolog includes a place to include user-defined macros, INITIAL block, front part of the DYNAMIC block, and the front part of the DERIVATIVE block.  If the user wants to invoke his macros in the model, such macros should be included in the prolog.  Either the macros themselves, or an INCLUDE statement providing the path of the macro file may be included in the prolog.

The epilog includes END statement for the Derivative section, a place to insert Discrete sections, an END statement for the Dynamic section, a Terminal section, END of model definition, and subroutines.

Modification of the prolog and epilog allows the user to include macros and FORTRAN subroutines in the MMS model. This gives the user a virtually unlimited ability to customize the resulting MMS model. Fortran subroutines (either source, object, or libararies) can also be included in the Project Settings.

When the model is built, the Program statement and call to Utils1 macro (for unit conversion) are automatically included in the model .csl file by the Model Builder, unless the "Full Model" option has been unchecked.

## Efficiency Tips

It is very important to get all the information for the system to be modeled at the outset and verify the correctness and consistency of data (i.e., mass and heat balance, component capacity and characteristics). If there are any inconsistencies, these should be eliminated by consensus with the customer. Comments regarding the source of data and assumptions should be added in the Comments column in the module AP form and Notes box in AP\ACSL Coding window. It is helpful to maintain a log of model evolution history in the prolog or epilog section for documentation purposes. In addition, the following suggestions will make the model development more efficient:

- **If the modeled system has similar subsystems,** one of these should be modeled and debugged, and then replicated using the copy/paste menu.

- **Parameters common to the system** are best entered once and then brought into adjoining modules by using the forward or backward parameter passing method and the "Get from Connect" button in the data entry window of a module, and by using the "Pass In" or "Pass Out" parameter passing Mode in the model configuration graphic.

## How to Specify Boundary Conditions

It is important to specify the boundary conditions correctly. Boundary conditions can greatly affect the behavior of the simulation. Sometimes it is appropriate to represent the boundary as an ideal pressure source. Most frequently, the boundary should be represented as an "equivalent" flow resistance in combination with an "equivalent" elevation change and an "equivalent" ideal pressure source. The "equivalent" values are chosen so that the conditions at the boundary will have the same pressure flow characteristics of the system to be modeled. More infrequently, it may be necessary to represent the boundary as an ideal flow source. One way to accomplish this is to use an ideal pressure source and valve with a PI controller to regulate the flow to the desired value.

For example, consider a model which includes only an economizer, once-through furnace, primary and secondary superheaters, attemperator, HP turbine, primary and secondary reheaters. The SL port at the exhaust of the reheaters should be connected to a pipe and a RIGHT pressure boundary. The pressure boundary would be at the condenser pressure. The flow resistance of the pipe represents the pressure-flow characteristics of the components between the outlet of the reheaters and the condenser.

The WE port at the inlet of the economizer should also be connected to a pipe and a LEFT pressure boundary. The pressure boundary represents the deaerator pressure. The flow resistance of the pipe represents the flow resistance of components between the economizer inlet and the deaerator.

## Data Collection for Simulation

This information provides a general overview about what is required for collecting information for the module AP input forms.

**System Characterization**

Steps in the process of developing a simulation include identifying the objectives of the simulation and choosing the scope of the simulation. After the scope of the simulation is established, the next step is to gather data required in the component data input form. Some general principles to identify the kinds of data required for different kinds of components are presented here.

The system to be simulated is specified by specifying the following:

- **Boundaries**
- **Flow paths**
- **Components**
- **Operating points**

Boundaries are sources and sinks of the following:

- **Pressure**
- **Temperature**
- **Torque**
- **Voltage**
- **Analog or digital control signal**

Flow paths define the transfer of the following:

- **Fluids,** such as water, oil, and gas
- **Heat** between components and to/from boundaries
- **Mechanical energy,** including conversion between components and to/from boundaries
- **Electrical energy,** including conversion to heat or mechanical power
- **Control signals** between control elements and with other components via transducers and actuators.

## Component Characterization

The classes of components are listed below with examples:

- **Flow components**
- **Flow conductance**
- **Forward and backward**
- **Variation with valve position**
- **Flow inertia**
- **Volume**
- **Elevation change**

Heat transfer components

- **Mass**
- **Specific heat capacity**
- **Heat transfer coefficients**
- **Heat transfer area**
- **Thermal conductivity**

Mechanical components

- **Moment of inertia**
- **Efficiency**
- **Where does thermal loss go?**
- **Performance curves versus speed**
- **Performance curves versus operating conditions**

Control components

- **Inputs and outputs, digital and analog**
- **Final control elements**
  - Stroke time
  - Time constant

    - Failure modes
- **Function blocks**
    - PI controller
    - Nonlinearities
        - Hysteresis
        - Deadband
    - Transfer function
    - Function generator
- **Control loop tuning**
    - Proportional gain
    - Integral gain
    - Derivative gain
    - Setpoints
    - High limits
    - Low limits
    - Anti-windup characteristics

Electrical components

- **Generator -** Moment of inertia, saturation characteristics, V curves, stator resistance and synchronous reactance, fixed and variable losses, MVAR versus MW capability curves, field voltage versus field current characteristics
- **Transformer –** Nameplate data - rated primary/secondary voltage, MVA rating, impedance
- **Electrical reactors –** Nameplate data – impedance (resistance and reactance)

Power sources

- **Boiler – Fuel characteristics**
- **Gas turbine – performance curves**

## Operating Point Data

System-wide steady state operating point data at the inlets and outlets of components may be available from plant data acquisition systems or from heat balance calculations or other sources.

Operating point data may be used to infer many component characteristics, which may be used to account for changes in equipment condition or to otherwise supplement component data from other sources.  Examples of useful operating point data include the distribution of the following phenomena:

- **Pressure**
- **Flow**
- **Specific enthalpy and temperature**
- **Power**
- **Tank levels**
- **Valve positions**
- **Pump or turbine speeds**

## Model Building

Once the prolog/epilog have been defined, the model has been graphically configured, and all data entered, the model can be built by clicking the "Build Model" button in the MMS Model Builder.  The model building process includes creating the model source code in ACSL, translataing it into the model fortran file by using any user-defined macros included in the prolog and the MMS Macro Library, compiling the model fortran file and any subroutines included in the model or project, and linking the object files, user-specified libraries, and MMS fortran library into the model executable file (.prx).  A default command file is automatically generated the first time a model is saved.

## Model Debugging

During the various phases of model building (i.e., translate, compile, link, run), it is possible and normal to encounter errors.  Some errors are very easy to correct, while some are quite involved.

Translation errors related to syntax and used-but-not-defined or multiply-defined variables are usually easy to correct.  Errors related to unsortability (i.e., unsortable statement block or algebraic loop) are usually more difficult to correct.  To correct such errors, the chain of model statements constituting the unsortable block needs to be broken or decoupled somewhere.  There may be several places for breaking; the choice depends on the model logic.  The decoupling is achieved by one of several methods, e.g., inserting a first-order lag for the chosen variable, hiding the variable in a procedural block (i.e., by omitting from the argument list of the block).  Sometimes this error happens if unrelated variables are evaluated in the same if-then block.  In such a case, the break up of the if-then block into two or more blocks corrects the error.

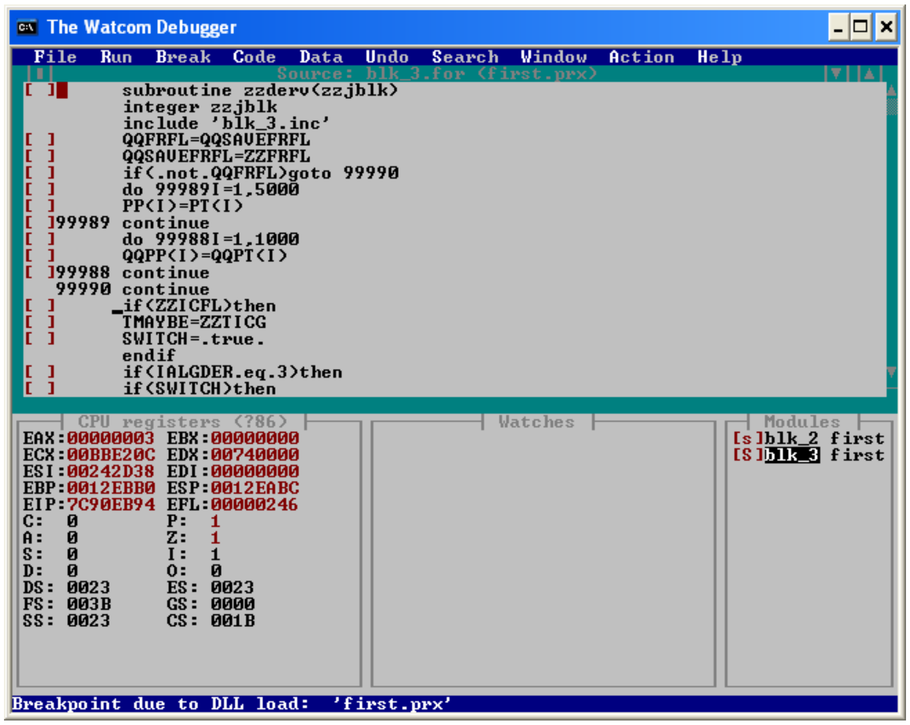
If a module includes a TABLE definition and other modules call the table, the ACSL coding of the other modules may get inserted in the .csl file before the TABLE definition module, and may cause a translation error.  This can be corrected by selecting the "Mode\Set Module Order" menu item and double-clicking on the TABLE definition module, so its ACSL code will be forced to be inserted before the other modules.

Compile errors are usually found in custom coding inserted in the model in a CODE block, prolog, epilog, and FORTRAN subroutines. Typical errors are syntax, conflicting type declaration, lack of declaration, omission of labels, formats, and ends of if-then blocks.

Models having a very large number of variable names in the common block may take too long to compile. Splitting the .for file in several parts results in a reasonable compile time. Refer to the PC Split option in "Settings \ Project Settings" menu.

Link errors are related to unspecified FORTRAN, object, and library files in the ACSL project.

Run-time errors can arise from divide by zero, floating point errors, illegal operations (e.g., log of negative number), array addressing outside declared space in DIMENSION statement. These are generally quite difficult to diagnose, but relatively easy to fix. The debugger is a powerful tool to diagnose errors. Less formal methods include adding temporary write statements in the model FORTRAN file to determine the ballpark in which the error occurs, displaying values of variables calculated at various places in the FORTRAN file. In the latter case, variables calculated just before the error location will have valid values, while those following it will have values like 5.55d33 or 555555333.


The Watcom Debugger
File Run Break Code Data Undo Search Window Action Help
Source: blk_3.for (first.prx)
CPU registers (?86)
Watches
Modules

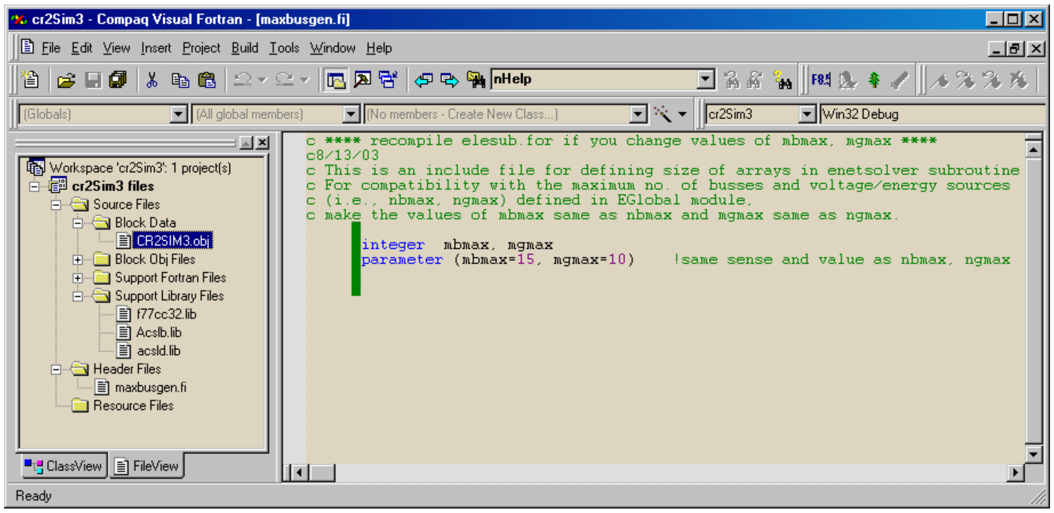
```
[ ]          subroutine zzderv(zzjblk)
             integer zzjblk
             include 'blk_3.inc'
[ ]          QQFRFL=QQSAVEFRFL
[ ]          QQSAVEFRFL=ZZFRFL
[ ]          if(.not.QQFRFL)goto 99990
[ ]          do 99989I=1,5000
[ ]          PP(I)=PT(I)
[ ]99989     continue
[ ]          do 99988I=1,1000
[ ]          QQPP(I)=QQPT(I)
[ ]99988     continue
    99990    continue
[ ]          if(ZZICFL)then
[ ]          TMAYBE=ZZTICG
[ ]          SWITCH=.true.
             endif
[ ]          if(IALGDER.eq.3)then
[ ]          if(SWITCH)then
```

```
EAX:00000003 EBX:00000000
ECX:00BBE20C EDX:00740000
ESI:00242D38 EDI:00000000
EBP:0012EBB0 ESP:0012EABC
EIP:7C90EB94 EFL:00000246
C:  0        P:  1
A:  0        Z:  1
S:  0        I:  1
D:  0        O:  0
DS: 0023     ES: 0023
FS: 003B     GS: 0000
SS: 0023     CS: 001B
```

```
[s ]blk_2 first
[S ]blk_3 first
```

```
Breakpoint due to DLL load:  'first.prx'
```

**Figure 12:** The Watcom Debugger



```
c **** recompile elesub.for if you change values of mbmax, mgmax ****
c8/13/03
c This is an include file for defining size of arrays in enetsolver subroutine
c For compatibility with the maximum no. of busses and voltage/energy sources
c (i.e., nbmax, ngmax) defined in EGlobal module,
c make the values of mbmax same as nbmax and mgmax same as ngmax.

      integer  mbmax, mgmax
      parameter (mbmax=15, mgmax=10)     !same sense and value as nbmax, ngmax
```

**Figure 13: The Compaq Debugger**

## Executing Model

When you click the "Execute Model" button, the model will run in the ACSL runtime window.  The model command file (.cmd ) will be read and executed.  More commands can be typed in the command-line window at the bottom.  A collection of previous commands is available in the drop-down window for later use.  The ACSL runtime menu Setup/Output may be used to select the model variables of interest for seeing or plotting.  After the output list is selected, it can be captured in the command file by typing q (the quit command), editing the .log file, and adding the commands in the .cmd file.  Similarly other commands typed during model execution can be added to the .cmd file.

Once the model runs without any numerical problems, it is customary to run the model to steady state and perform eigenvalue analysis (ANALYZ /EIGEN).  The eigenvalues are complex numbers with real and imaginary parts.  They can be considered as inverse of time constants and provide verification of model stability.  If all eigenvalues have non-positive real parts, the model is stable.  If eigenvalues have positive-real part, either the process itself is unstable, controller gains are too high, or coding of the custom coded equations may have errors.

# 4. How to Build and Run an MMS Model

Building an MMS model is an easy task because of the user-friendly features of MMS.  The following detailed steps serve as a guide during model development

## 1. Determine model scope and collect data

With the help of drawings and discussions with the plant personnel, determine the components to be included in the model and the steady state and dynamic performance to be studied, the nature of the study – operator training, control configuration checkout, analysis, etc.

Gather plant output rating, component nameplate and vendor-supplied data, heat balance data, model verification data, etc.

## 2. Open model

Open the Model Builder.  A new model page will appear.

As a reference for the following model building steps, online help on the Model Builder is available by clicking on "Help\Index" or "Help\Contents" in the main menu.  Online help on ACSL is available by clicking on "Help\ACSL" in the main menu.

The paths and directories for modules, dll's, and help files are preset during MMS installation. However, if you have difficulty while selecting modules for your model, auto-parameterizing, or seeing online module help, verify the path settings, and edit if necessary, by selecting "Settings\Directories/Paths" from main menu.

## 3. Select MMS Prolog/Epilog pair (you can do this anytime)

Go to "Simulation\Code Generation Settings\Prolog/Epilog Code" from the main menu.  Click "Predefined" and select the appropriate pair.  The "MMS Only" pair can be used for models that are not required to run in real time and therefore don't require RTC modules.  The "MMS – RTC" pair is used for models that include RTC modules and are to be run in the ACSL environment.  The "MMS – RTC w/ Simlulator Link" pair is used for models that include RTC modules and are to be run in the simulator or ACSL environment.

The Prolog/Epilog Code contains the ACSL model sections (e.g., Initial, Dynamic, etc.) in the MMS-specific manner.  Add "include" statements to include user-defined macros, discrete blocks, subroutines, etc., at specified locations; discrete blocks are used to simulate sampled data systems such as the functional logic in DCS controllers.  Edit the code to add comments regarding model development (like a log), change any predefined settings, etc.

## 4. Save model

(Repeat this several times during model development.)

Save the model (.mms extension is used).  A default command file will automatically be saved the first time the model is saved.

## 5. Define Project Settings (you can do this anytime)

Click on "Settings\Project Settings" and specify settings and options specific to the model.  Specify model specific .for, .lib, and .obj files to be included in the model.  If the model is large, you can select "Use PC Split" option to accelerate the compile process by splitting the translated fortran model into smaller parts that compile significantly faster.  If other PCs or cpu's are available on the LAN, you can further reduce the compile time by using them for compiling.  To use the multiple compile process, select the "Use Multiple Compile Processes" box in the "Settings\Project Settings\Project Options" menu, map the PC running the model to a drive on the local computer, change to that model directory, and type the ***compile*** command.

You have also to use the PC Split option to run the model in real time using the simulation tools.

If you want to force the linker to use a modified subroutine (.for or .obj) instead of the same subroutine in the library, specify /force in the "linker switches" box.

## 6. Select modules, connect them

Click on the "Add Module Mode" button and the last module category (e.g., RTC\Fossil) will appear. Navigate to the appropriate category, select the desired module, and click at the location where you want this module on the model page. The module will appear there. Repeat until the desired modules are on the model page.

Special modules like CODE and UJUNC (universal junction) are often used in the model. CODE is used to insert user-defined code in the model and can be connected to any module. UJUNC is used to connect two or more RTC modules where a pressure calculation must be made to solve the pressure-flow network.

Where necessary, select a module and press F1 key to see module online Help to understand the details of module functioning.

Connect the modules. Use Connection line attributes (color, thickness, etc.) to distinguish the types (e.g., water – blue, steam – red, control signals – dashed purple). If necessary, refer to the list of ports and allowed connections in this document.

Use Add\Rectangle, etc., to add text notes for model development log and special markings.

## 7. Enter module data and AP

For each module in the model, double click the icon to see the Data Input window for entering data. Customize the Description and Identifier (called ID). The default units are US; select "SI Units" box if you want SI units. The entire model must use the same units. SI units can also be selected by using main menu "Settings\SI Units".

Enter input data, adding supporting information (e.g., reference or assumed) in the comments column. In some modules (e.g., PIPE), you have to select PPmethod (parameter passing within the module) and the data input requirements change accordingly (method TEST1 is seldom used). For example, if you choose "Forward", you have to input the inlet values and delta-P, delta-h. Similarly you may have to select the APmethod. The most common choice is UseBCs. "Mixed" is used in special situations and allows user to specify method of calculating each output parameter. Method TEST1 is seldom used.

For consistency of flow, pressure, enthalpy, etc., across interconnected ports of adjacent modules and to avoid retyping same values twice, you can use "Pass (in)" or "Pass (out)" buttons. Thus if you select "Pass (in)" button and place the cursor at the In port of a module, the values of flow, etc., will be passed from the port at the other end of connection.

After completing data input, click the "Auto Parameterize" button; the output data as calculated by the module AP code will be automatically entered in the AP data form.

Click the AP/ACSL Coding button to see the ACSL template. Click the Expand button to see the ACSL coding; this will be included in the model source code during csl file generation. Verify the coding for completeness (e.g., if one of the ports that must be connected is not connected, the macro call in the coding will not be complete). Also examine the Expansion Notes (warnings).

Click the Notes option at the bottom and type any comments regarding the module attributes (assumptions, data source, etc.).

## 8. Build model

After completing the model configuration, save the model and click the "Build model" button. The model source file (.csl) will be generated, translated, compiled, linked to produce an executable model file (.prx). Before linking, any fortran subroutines specified in the project settings will also be compiled. If there are any translation or compilation errors, resolve them (ACSL and Fortran Reference manuals will be helpful) and rebuild model.

After the model builds successfully, examine model.wrn or model.out file to see if there are any "Used But Not Defined" variables. If there are, define these variables in the model and rebuild or define them in the command file.

## 9. Edit command file

A default command file (.cmd) is automatically generated when you first save the model. This file includes a proced called s_master, that is required when running the model with Sim_Tools. Edit this file to add commands like:

```
set NRWITG=.t.    !no rewind of plot data file

out t, variable1, variable2

prepare t, variable1, variable2
```

The variable names can be found from the expanded ACSL code for various modules or can be determined from the stream name (e.g., P_streamname for pressure) found by double-clicking on a connection line and clicking the "Change" button. ACSL Reference manual will be helpful.

## 10. Execute model

**In ACSL Runtime environment**

Click the "Execute Model" button. The model will run in the ACSL run-time window and execute the commands in the command file. Commands like:

```
start, contin, s tstop=100., plot
```

can be typed in the command line or composed from the online ACSL run-time menu. Commands for a standard execution sequence can be added from the .log file in the command file in a "proced/end" pair for future execution by simply invoking the proced name.

**In SimTools environment**

If you want to run your model with dynamic interaction as in a training simulator, you have to use the "MMS – RTC w/ Simlulator Link" prolog/epilog pair and click the "Set Standard SimTools Options" button in "Project Options" window before building the model. The resulting model .prx file can also be executed in ACSL Runtime environment.

To run in SimTools environment, execute S_Master and use "File\Select Runtime Model" menu to select the .prj file of the model to be executed. Click on the "Load/Start" button; the ACSL Runtime window will appear and the s_master proced in the command file will be executed. You will see the model time increment at the bottom of the S_Master. You can execute other components of Sim_Tools, i.e., nView (to see and set selected model variables), G_Master (to see trends), Action

Center (Instructor station), MMI Manager (to interact from MMI screens).  For more on Simulation Tools, see Chapter 7.

# 5. How to Build and Run an ACSL Example Model with MMS

The ACSL Beginners Guide (Reference 7) includes examples which may be compiled and run using the Model Builder.  The procedure for this is similar to the procedure for building and running an MMS model, except for the following:

- **Select the "MMS Only" Prolog/Epilog pair.**
- **Include the example model source code explicitly or by an "include" statement at the end of the Prolog code or at the beginning of the Epilog code, or by configuring the model with a CODE block containing the example model source code in its ACSL template.**
- **Exclude the Program statement since the Model Builder automatically includes the *Program* and *Util1* statements in the .csl file.**
- **Edit system variable names (e.g., tstop, cint, quit) used in the prolog to avoid conflict with similar names used in the example model source code.**
- **Edit model variable names (e.g., x, dx) used in the example model source code if they conflict with similar names used in MMS pressure-flow solver (as indicated by "multiply defined" translation error).**
- **After the model is first saved, add example model commands to the command file.**
- **Execute the model only in the ACSL run-time environment.**

# 6. How to Create an MMS Module with MMS CompGen

This chapter summarizes the procedure for creating an MMS module, using MMS CompGen.

## 1. Prepare outline of module documentation

Prepare an outline of the module icon, connection ports and types (refer to the list of ports and allowed connections in this document), module dynamics equations, parameter calculation equations, and help document.  Use an MMS module as an example.

## 2. Execute MMS CompGen and open the module page

CompGen.exe is located in the MMS\Bin folder.  A shortcut to CompGen should exist in the Windows Start Menu, MMS group .  CompGen defaults to a new blank module page.

## 3. Familiarize

Familiarize yourself with the Help, Settings, and other main menu items and buttons.  The Help\Index menu provides online reference for Compgen and Settings\Directories/Paths shows the path of various Compgen-related files.

The user-created module, DLL, and Help files must be located in the C:\MyMMS\Components directory.

## 4. Create module icon

Click on the Draw Rectangle Mode button if you want the icon to have a rectangular boundary.  You may also click on the Rounded Rectangle or Ellipse mode buttons. Draw the module symbol using the various "Add" mode buttons and choice of color, etc., and "Edit" menu.

Click "Draw Port Mode" button and draw port rectangle at desired locations to add ports for flow stream connections, control signal connections, electrical power connections, etc.

Click on "Add/Edit ID rect" and "Add/Edit Notes rect" buttons to add rectangles M and N for module name and function, respectively.  Select and drag the rectangles to desired location.  Select "Edit\Notes for Rectangle" menu and type module function.  Click on the "Edit Module Parameters" button and type module Default ID.  Click OK and see the module ID appear on the icon.

## 5. Save module

Save the module.  Repeat this several times during module development.  Also save the icon picture for documentation purpose as a .emf file by selecting "File\Save Metafile" menu.

## 6. Define Port ID and Type

Double click on the port to assign port type (drop down box) and name.  The list of port types can be seen by selecting "Edit\Connect Types" menu.

## 7. Define Data Input form entries

Click on the "Edit Module Parameters" button and type Description and an eight or less character long Macro Name (which is also the AP dLL name).  Use "Add", "Edit", and other buttons to create variable definitions.  Typically the variables that serve as Inputs to AP calculations are defined first, followed by the Output variables that are calculated by the AP code dLL.  These variables are typically subdivided into several categories (e.g., Physical Parameters, Operating Parameters) for user convenience (use the Data Input form of a standard module as guide).  Intermediate calculated values can be included in the form for user information.  The variable names are used in the AP/ACSL coding (.cpp and .csl files).  Thus if the macro name or Data Input form entries are later changed or new entries are added, the AP code will have to be regenerated and the ACSL code will have to be regenerated or edited.  Also if the output names used in the ACSL image are consistent with the MMS naming convention (e.g., Kxxxx for constant parameters, Ixxxx for initial conditions), the editing in ACSL Coding is minimized.

## 8. Build AP routine

Click on the "Edit AP CSL coding and Notes" button in Compgen and click in the "Auto-Parm" circle.  Click "New" button to generate the C++ file module.cpp.  Insert the AP code for the parameter calculation equations between the lines:

```
//******** BEGINNING OF AUTO-PARAMETERIZATION (AP) CODING ********

//**************************** END OF AP CODING ****************************
```

The variables in the Data Input form are automatically declared in the Prolog and Epilog Include files. These files can be seen by clicking in the "Prolog Incld" and "Epilog Incld" circles. The AP code should calculate the Output variable values from the Input variable values.

Click the "Build" button to compile the AP routine and build the .dll file. The messages about the build process can be seen by clicking in the "Build Results" circle.

### 9. Create ACSL Image

As in the **Build AP routine** paragraph, click in the "ACSL Image" circle and click "New" button to create module.csl file. Edit the macro call (or add ACSL coding), initial conditions and constant parameter definitions in the ACSL image and insert comments (see ACSL Coding in Compgen online help). Click the Expand button to verify if the string substitutions (e.g., <&ID>) and assertions (e.g., <V?Leg=0>) expand correctly.

### 10. Add notes in Notes menu of AP/ACSL Coding window

As in the **Build AP routine** paragraph, click in the "Notes" circle and add notes regarding module development, assumptions, etc.

### 11. Test the module by making a test model

Save the module and exit from Compgen. Create a test model in Model Builder using this module and boundary conditions, etc., to test the module. Iterate between the Compgen and Model Builder until the module is debugged.

### 12. Create Help file

Create a help document for the module and convert it to .chm format using Robohelp html. Add it in the c:\MMS\Help directory for online access.

# 7. Simulation Tools

The nHance Technologies simulation tools (also called SimTools) consist of a number of different applications. The major tool set includes:

- **Simulation Master (S_Master)**
- **nView**
- **Graph Master (G_Master)**
- **Action Center**
- **MMI Manager**

The tools provide continuous "on-the-fly" interaction with the MMS model. Their interrelationship is shown in Figure 14.

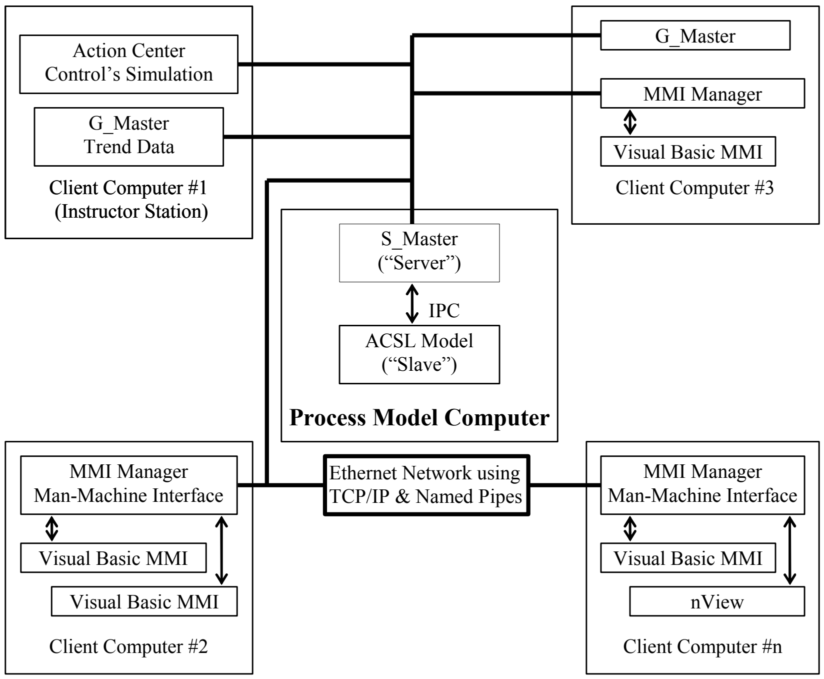


**Figure 14:** Overview of SimTools

Of the tools mentioned above, the S_Master application could be considered as the "server." The rest of the tools can be considered as "clients." Figure 14 graphically depicts how the tools fit together. Note that for a given simulation, only a single instance of S_Master and a model are allowed. In addition, only one copy of the Action Center can be connected to S_Master. Any number of the other tools can be connected to S_Master at any given time. To add another instance of nView, type "nView 2" from a DOS window. Figure 14 indicates that several applications can be simultaneously connected to S_Master over a network, or on the same computer, depending on the user's preference or requirements.

## Simulation Master (S_Master)

Simulation Master or S_Master is the main component in the simulator environment. S_Master's main function is to connect to the model and service client requests. The client requests include model commands, such as start/stop, save/restore state, etc. S_Master also serves other requests, such as getting or setting individual variables, or lists of variables.

In addition to serving client requests, S_Master can pace the ACSL model to a specified real-time scale factor and has the capability of linking with external processes. The links to an external process are used in the event that the process model is linked with an external model, e.g., an MMS process model linked with an external control system model or the actual control hardware. In this case, S_Master is capable of telling both the MMS model and the external control system to take time steps, freeze, save state, etc., thus ensuring both models stay in synchronization with each other.

S_Master saves simulator data images with the following file extensions .ic, .ssf, .tbt, and .spf (details in Data Images section). In addition, S_Master keeps log files that are automatically saved with the .slg file extension.

## Graph Master (G_Master)

Graph Master or G_Master is a real-time trending application. G_Master communicates a list of variables to be trended to S_Master. S_Master, in return, responds with the current value of these variables at each S_Master defined communication interval. G_Master displays the data in a number of different formats, performs simple statistical calculations of the data, and has the capability to export the data for additional external processing.

G_Master saves data files with the .gmg file extension. G_Master can export its data in delimited ASCII text files with the .tmd file extension.

## Action Center

The Action Center contains the traditional power plant simulator "Instructor Station" features. It has the capability to give ACSL variables contextual names (e.g., flow set point instead of kwsp), and manipulate these variables in a user-friendly manner by typing values or using a slider control. In addition, the Action Center contains the capability to communicate model commands, such as run/freeze, save/restore states, etc. to S_Master. The Action Center uses OLE automation to link into G_Master so that Action Variables can be trended with the click of a button. OLE automation is also used to allow Visual Basic to link into the Action Center to provide for a Visual Basic scripting language.

## MMI Manager

The MMI manager interfaces with 32-bit versions of various development applications such as Visual Basic 5.0, Visual Basic 6.0, Visual C++, etc. With the MMI Manager and its custom controls, the user is able to easily create Hard Panel emulations, or Man Machine Interface screens. The MMI Manager gives the Visual Basic programmer an easy way to get and set model variables from a remote process. The MMI Manager is designed to enable connecting multiple Visual Basic Screens to the same MMI Manager and contains provisions for the communication of information between the screens.

## nView

nView is a simple but powerful interface with the model through MMI Manager and S_Master. At start, it includes two tabs (i.e., pages) in the "10Test" format (i.e., 10 variables/page): "nView" for selected model variables and "Process Model" for generic model variables like t. To see or set a variable, simply enter the ACSL variable name. The value will appear. If the variable can be set (i.e., is not calculated in the model code) and you want to change its value, type the desired value and click the "Set" button. You can also see the pages in "DataView" format by deselecting the "10Test" from the View menu. In this view, you can see more than 10 variables. To add another tab, right-click on a tab and click the "Add System Tab" button. The nView data file is saved as a .nvw file.

## Definitions

The following items are frequently used when discussing Simulation Tools:

### Action Variables

An Action Variable is an object whose value is determined by one or more process model variables. The name of an action variable is usually more descriptive than the variable name used in the process model. Because an action variable is an object, additional properties can be associated with it, such as scale & offset, upper/lower limits, engineering units, etc. In addition, action variables can be assigned to an arbitrary system (e.g., Process model, Control model) and secondary system (e.g., Pressures, Setpoints) and sorted based upon these systems.  Simply put, action variables are Windows objects that serve as wrappers around model variables.

### Actions
Actions are objects that set action variables to predefined values.  Typically, an instructor will change external conditions, initiate a malfunction or a training scenario, or perform a "local" operator action from the instructor station.  The Action Center classifies all of these operations as actions, as they all perform essentially the same thing;  namely manipulate one or more action variables.  Upon initiation, actions are automatically placed in either the pending, or the active state.

### Active Actions
An action is considered active if it is in the middle of performing an action that requires more than one time step to complete.  An example of an active action is the ramp action, which will be active over the duration of its ramp.

### Pending Actions
An action is considered pending if it has been initiated, but its triggered condition has not been met.

### Triggered Conditions
Triggered conditions can be associated with all actions.  Triggered conditions are expressions built with action variables and logical operators that are evaluated to determine the appropriate time for activation of an action.

### Data Images
The term data image is used to describe all simulator run-time data.  Thus, the different types of data images are:  Initial Condition (*.ic) files, Saved Snapshot (*.ssf) files, Timed Backtrack (*.tbt) files,  and Parameter (*.spf) files.  All of the data image types are binary files and are discussed below.

### Saved Snapshot File (.ssf)
The saved snapshot file is a file stored on disk that contains the state of the simulation at a given point in time.  This image contains all simulator variables, including states and parameters.

### Timed Backtrack File (.tbt)
A timed backtrack file is a saved snapshot file that has been automatically saved by the simulator. The number of backtrack files saved, and frequency in between file saves are user configurable. Timed backtrack files differ from saved snapshot files simply by the way in which they were saved. Timed backtrack files are saved automatically by the simulator, saved snapshot files have to be saved by a user.

### Parameter Files (.spf)
Parameter files are a subset of simulator data.  The intent is that parameter files contain only simulator variables that do not change from one time-step to the next.  An example of a parameter is an actuator time constant.  Sometimes, however, it is difficult to define whether a variable is a parameter or not.  Because of this difficulty, the distinction between parameter and non-parameter variables has been left up to the model developer.

### Default Parameter File
The default parameter file is a special parameter file that is always loaded with an initial condition.  If the simulator detects that the default parameter file does not exist, it will automatically create one. The name of the default parameter file is user configurable.

### IC’s or Initial Condition Files (.ic)
An IC file differs from a saved snapshot file in that after the IC is loaded, the default parameter set is restored.  Thus, an IC can be thought of as a saved snapshot file, less parameters.

## Security

The instructor station software (i.e., Action Center and G_Master) supports three levels of security. These three levels are:  Manager, Instructor, and Student.  The user’s security level is determined by membership, or lack thereof, in certain Windows NT local groups.  If the current user is a member of the group "Manager," he is considered to be a simulation manager.  If the user is a member of the local group "Instructor," he is considered to be a simulation instructor.  If the user is not a member of the other two groups, or is a member of the local group "Student," he is considered to be a simulation student, and will have very restricted rights.  See the Windows NT documentation for information on user accounts and local groups, which are defined in Control Panel \ Administrative Tools \ Computer Management \ Local Users and Groups.

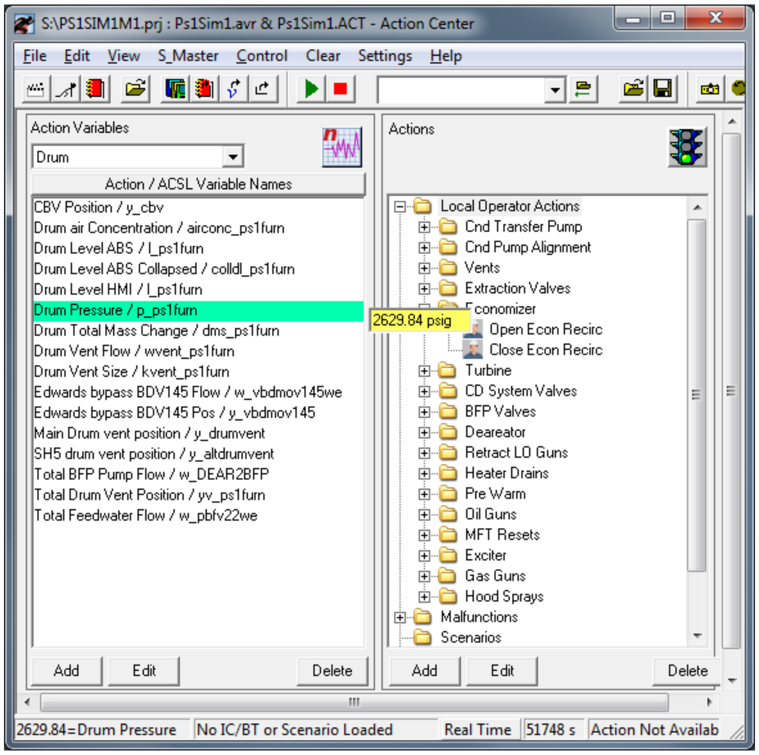
## Using the Action Center

The Action Center is an instructor station software application used for traditional model control, activation of local operator actions, setting external conditions, and initiating malfunctions.  The Action Center saves action variable files with the .avr file extension and action files with the .act file extension.  In addition, the Action Center is capable of saving an ACSII log file with the .log file extension.

### Action Center Start Up
When the Action Center is initially started, it attempts to connect to S_Master.  If this connection is complete, the user will see the model control toolbar, as presented in "Action Center, Menus and Toolbars" section later.  If the model control toolbar does not appear, the user should verify that S_Master has been started, and that a model is running.  If the model control toolbar is still not visible, the user should choose the S_Master/Connect menu item from the Action Center.

Once connected to the model, the user should load up a set of action variables and actions.  To do this, simply select the File/Open menu item.  The open dialog box defaults to action variables files, so only a list of action variables will be presented.  Select the desired action variable file and choose open.  After the action variables have been loaded, select the File/Open menu item again.  This time, change the file type to action files, and choose the desired action file.

## Action Center Views


S:\PS1SIM1M1.prj : Ps1Sim1.avr & Ps1Sim1.ACT - Action Center
File Edit View S_Master Control Clear Settings Help
Action Variables
Drum
Actions
Add
Edit
Delete
Add
Edit
Delete

Action / ACSL Variable Names

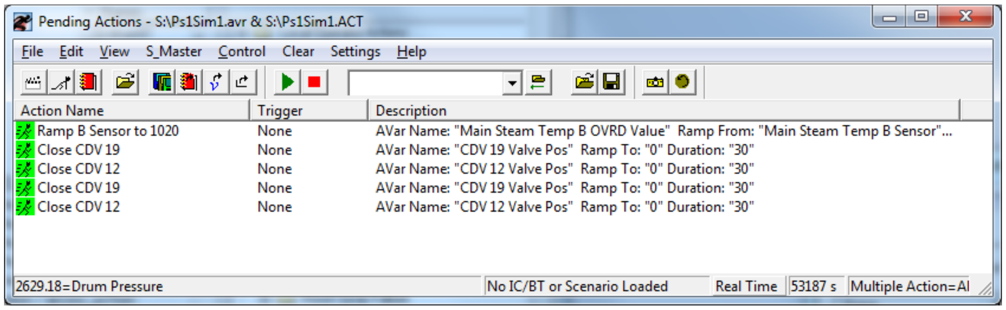
CBV Position / y_cbv
Drum air Concentration / airconc_ps1furn
Drum Level ABS / l_ps1furn
Drum Level ABS Collapsed / colldl_ps1furn
Drum Level HMI / l_ps1furn
Drum Pressure / p_ps1furn
Drum Total Mass Change / dms_ps1furn
Drum Vent Flow / wvent_ps1furn
Drum Vent Size / kvent_ps1furn
Edwards bypass BDV145 Flow / w_vbdmov145we
Edwards bypass BDV145 Pos / y_vbdmov145
Main Drum vent position / y_drumvent
SH5 drum vent position / y_altdrumvent
Total BFP Pump Flow / w_DEAR2BFP
Total Drum Vent Position / yv_ps1furn
Total Feedwater Flow / w_pbfv22we

Local Operator Actions
Cnd Transfer Pump
Cnd Pump Alignment
Vents
Extraction Valves
Economizer
Open Econ Recirc
Close Econ Recirc
Turbine
CD System Valves
BFP Valves
Deareator
Retract LO Guns
Heater Drains
Pre Warm
Oil Guns
MFT Resets
Exciter
Gas Guns
Hood Sprays
Malfunctions
Scenarios

2629.84 psig

2629.84=Drum Pressure | No IC/BT or Scenario Loaded | Real Time | 51748 s | Action Not Availab

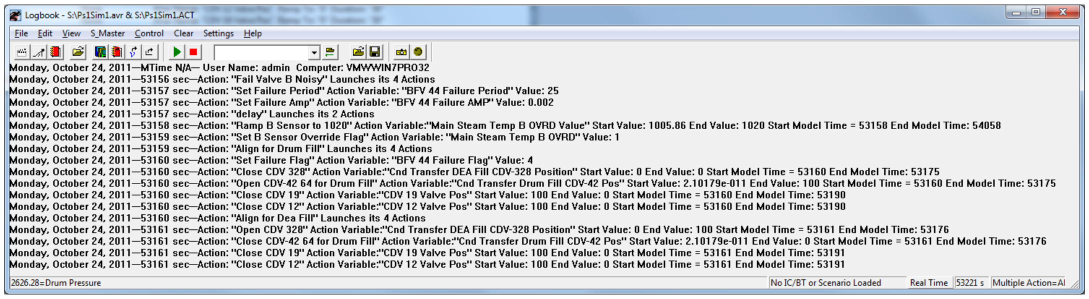
**Figure 15:** Action View


Pending Actions - S:\Ps1Sim1.avr & S:\Ps1Sim1.ACT
File Edit View S_Master Control Clear Settings Help

| Action Name | Trigger | Description |
|---|---|---|
| Ramp B Sensor to 1020 | None | AVar Name: "Main Steam Temp B OVRD Value" Ramp From: "Main Steam Temp B Sensor"... |
| Close CDV 19 | None | AVar Name: "CDV 19 Valve Pos" Ramp To: "0" Duration: "30" |
| Close CDV 12 | None | AVar Name: "CDV 12 Valve Pos" Ramp To: "0" Duration: "30" |
| Close CDV 19 | None | AVar Name: "CDV 19 Valve Pos" Ramp To: "0" Duration: "30" |
| Close CDV 12 | None | AVar Name: "CDV 12 Valve Pos" Ramp To: "0" Duration: "30" |

2629.18=Drum Pressure | No IC/BT or Scenario Loaded | Real Time | 53187 s | Multiple Action=Al

**Figure 16:** Pending View


Logbook - S:\Ps1Sim1.avr & S:\Ps1Sim1.ACT
File Edit View S_Master Control Clear Settings Help

```
Monday, October 24, 2011—MTime N/A— User Name: admin  Computer: VMWWIN7PRO32
Monday, October 24, 2011—53156 sec—Action: "Fail Valve B Noisy" Launches its 4 Actions
Monday, October 24, 2011—53157 sec—Action: "Set Failure Period" Action Variable: "BFV 44 Failure Period" Value: 25
Monday, October 24, 2011—53157 sec—Action: "Set Failure Amp" Action Variable: "BFV 44 Failure AMP" Value: 0.002
Monday, October 24, 2011—53157 sec—Action: "delay" Launches its 2 Actions
Monday, October 24, 2011—53158 sec—Action: "Ramp B Sensor to 1020" Action Variable:"Main Steam Temp B OVRD Value" Start Value: 1005.86 End Value: 1020 Start Model Time = 53158 End Model Time: 54058
Monday, October 24, 2011—53159 sec—Action: "Set B Sensor Override Flag" Action Variable: "Main Steam Temp B OVRD" Value: 1
Monday, October 24, 2011—53159 sec—Action: "Align for Drum Fill" Launches its 4 Actions
Monday, October 24, 2011—53160 sec—Action: "Set Failure Flag" Action Variable: "BFV 44 Failure Flag" Value: 4
Monday, October 24, 2011—53160 sec—Action: "Close CDV 328" Action Variable:"Cnd Transfer DEA Fill CDV-328 Position" Start Value: 0 End Value: 0 Start Model Time = 53160 End Model Time: 53175
Monday, October 24, 2011—53160 sec—Action: "Open CDV-42 64 for Drum Fill" Action Variable:"Cnd Transfer Drum Fill CDV-42 Pos" Start Value: 2.10179e-011 End Value: 100 Start Model Time = 53160 End Model Time: 53175
Monday, October 24, 2011—53160 sec—Action: "Close CDV 19" Action Variable:"CDV 19 Valve Pos" Start Value: 100 End Value: 0 Start Model Time = 53160 End Model Time: 53190
Monday, October 24, 2011—53160 sec—Action: "Close CDV 12" Action Variable:"CDV 12 Valve Pos" Start Value: 100 End Value: 0 Start Model Time = 53160 End Model Time: 53190
Monday, October 24, 2011—53160 sec—Action: "Align for Dea Fill" Launches its 4 Actions
Monday, October 24, 2011—53161 sec—Action: "Open CDV 328" Action Variable:"Cnd Transfer DEA Fill CDV-328 Position" Start Value: 0 End Value: 100 Start Model Time = 53161 End Model Time: 53176
Monday, October 24, 2011—53161 sec—Action: "Close CDV-42 64 for Drum Fill" Action Variable:"Cnd Transfer Drum Fill CDV-42 Pos" Start Value: 2.10179e-011 End Value: 0 Start Model Time = 53161 End Model Time: 53176
Monday, October 24, 2011—53161 sec—Action: "Close CDV 19" Action Variable:"CDV 19 Valve Pos" Start Value: 100 End Value: 0 Start Model Time = 53161 End Model Time: 53191
Monday, October 24, 2011—53161 sec—Action: "Close CDV 12" Action Variable:"CDV 12 Valve Pos" Start Value: 100 End Value: 0 Start Model Time = 53161 End Model Time: 53191
```

2626.28=Drum Pressure | No IC/BT or Scenario Loaded | Real Time | 53221 s | Multiple Action=Al

**Figure 17:** Log View

The Action Center supports three different views (Figure 15-17). A view can be thought of as what you are looking at in the Action Center Window. The main Action Center view, which is visible at startup, is the action view. The pending view is used to list all Actions that are active, or pending.

The log view is used to view a log of instructor station actions that have taken place since starting up the Action Center. The three different views are selected from the "View" menu item, or by choosing one of the first three icons on the main toolbar (Figure 18).

## Action Center Buttons, Menus and Toolbars

Some of the most commonly used Action Center menu commands are available as icons on various toolbars. Wherever possible, the use of the Action Center will be described via the icons on the toolbars. All Action Center toolbars are removable, and can, therefore, have different locations depending upon the individual users' preference. The two most frequently used toolbars in the Action Center, presented below, are the Main toolbar and the Model Control toolbar. The main toolbar is where different views are selected and Action Center data files are loaded and saved. The model control toolbar allows for starting/stopping the model, the quick selection and loading of IC's, and provides access to the open/save data image dialog boxes.

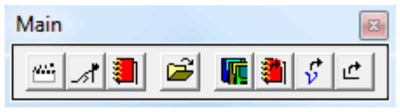


**Figure 18:** Main Toolbar

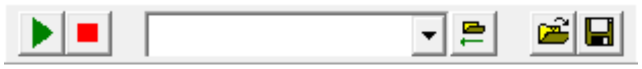


**Figure 19:** Model Control Toolbar

## Model Control

Figure 18 above presents the model control toolbar. The first two icons on this toolbar are used for starting and stopping the model. Thus, the green triangle icon is used to start the model, and the red square icon is used to stop the model. In addition to using the model control toolbar icons to start/stop the model, the Action Center Control menu can be used, or the user may use the keyboard by selecting the F3 key to start the model, and the F4 key to stop the model.

## Saving, Restoring and Deleting Data Images

Data images can be saved and restored from the Action Center directly. Deletion of data images, however, must be performed via File Manager or Windows NT Explorer, as described below.

- **Saving**
  The easiest way to save a data image is to use the save icon on the Model Control toolbar. This icon is the disk-like icon that is the last icon on the model control toolbar. The "Save Data Image File" dialog box will be presented once this button is pressed. The proper data image type should be selected, and the file name entered in the appropriate field. Once all information has been entered, the user should press the "Save" button to save the data image.

- **Restoring**
  The easiest way to load a data image is to use the open icon on the Model Control toolbar. This icon is the standard file open (open folder) icon that is the second from the right on the model control toolbar. The "Load Data Image File" dialog box will be presented once this button is pressed. The proper data image type should be selected, at which point, a list of only those data images will be presented. The user should select the data image to be loaded, and press the "Open" button to load the image.

- **Deleting**
  Data images are deleted through file manager, or Windows Explorer. In either case, once the utility is opened, drive S: should be selected. After drive S: is selected, the user may want to sort the list of available files by date. After the particular data image file has been located, all files of that same name, but with the various different file extensions should be selected. The delete key should be pressed to delete the images. Various Windows NT security levels can be enforced to allow/deny the deletion of these data images. See the Windows NT user manual for more information on the use of File Manager, Windows Explorer, and security.

- **Initiating Actions**
  Actions are sorted by file folders similar to the way in which files are stored on the Windows NT disk drive. The first step in initiating an action is to locate the action by navigating through the file folder structures. Once the action has been located, it should be highlighted by single clicking on the action icon. Once highlighted, the action can be initiated by pressing the Green-light icon, as seen at the beginning of this paragraph. The Green-light icon is located in the upper right hand corner of the action section on the Action Center. These steps are sometimes referred to as selecting the action and pressing "go."

- **Trending Action Variables**
  Action variables can be easily added to G_Master for data collection and trending. Before this feature can be used, G_Master must have been previously started. To add an action variable to G_Master, simply highlight the action variable by clicking on it, and pressing the G_Master icon, as seen at the beginning of this paragraph. The G_Master icon is located in the upper right hand corner of the action variable section on the Action Center. These steps are sometimes referred to as selecting the action variable and pressing "link." If the link was successful, the first G_Master document will start collecting data for the new variable.

## Configuring S_Master from the Action Center

S_Master, not the Action Center, is responsible for saving timed backtrack files, and pacing the simulator to run at the user selected time interval. The Action Center can, however, configure S_Master to perform these events via the 'S_Master/Config Setup' menu item. When this menu item is selected, the "S_Master Configuration" dialog box is presented. It is from this dialog box that the user should configure the following events.

- **Back Track Frequency & Number of Files**
  The backtrack feature can be turned on or off via the check box presented in the 'S_Master/Config Setup' dialog box. In addition, the user can select the frequency at which the timed backtracks are to take place, and the number of files to store.

- **Real Time Scale Factor**
  The real time scale factor adjustment is where the user can speed up or slow down the simulation. A factor of one indicates that the simulator calculations are performed at real-time, or, in other words, one second of simulation time is equal to one second of clock time. A factor greater than one is faster than real time, and a factor less than one is slower than real time. For example, a factor of 2 is twice as fast as real time, or one second of simulation time will be performed in ½ second of clock time. A factor of 0.5 is ½ as fast as real time, or one second of simulation time will be performed in 2 seconds of clock time. In all cases, the selection of the real time scale factor is only a desired simulation speed as the limits at which the simulator can be executed are governed by the available hardware, size of the model, etc.

**Security**
Simulation managers have the ability to add/edit actions and action variables and can modify the setup configuration of the Action Center.  Simulation Managers also have the ability to add/edit users and user permissions.

Simulation instructors have the ability to perform all of the traditional instructor station capabilities, such as start/stop the model, initiate malfunctions, etc.  Instructors do not, however, have the ability to configure the Action Center, add/edit user permissions, or modify actions or action variables.

Simulation students' use of the instructor station is limited to basic model control and activation of local operator actions.

## Using G_Master

**G_Master startup**
A blank document is created automatically when G_Master is started.  This blank document is essentially, a trend chart that contains no variables to be trended.  You can choose to add variables to trend to this graph, or you may decide to close this graph, and open a pre-defined file.  The Windows standard file/open menu items or toolbar icons can be used to open/save G_Master data files.

**Adding Trends to G_Master**
Trends are added to G_Master via the "Edit/Add Trend" menu item, or the "Add Trend Icon," as presented at the beginning of this paragraph.  Once the add trend option has been selected, the "Add Variable" dialog box will be presented.  This dialog box contains fields for the ACSL variable name, a Description, and a Scale & Offset factor.  If you are familiar with ACSL (the simulation language used in the process model) and MMS (the application used to build the process model), you can fill out this form by hand.  If, however, you are not familiar with ACSL & MMS, you can obtain the list of predefined action variables from the Action Center.  In order to obtain a list of action variables, the Action Center must be running on the same computer, and have a set of Action Variables loaded. Once this condition is met, you can press the "Action Center" button on G_Master's "Add Variable Dialog" box after "Add Trend" is selected.  This button will present you with a list of action variables to choose from for trending.

**Removing Trends From G_Master**
Trends are removed from G_Master via the "Edit/Remove Trend" menu item, or the "Remove Trend Icon," as presented at the beginning of this paragraph.  Once the remove trend option has been selected, the "Remove from List" dialog box will be presented.  To remove trends, simply select the variables to remove from this list and press enter, or the "OK" button.

**G_Master Views, Windows & Documents**
G_Master allows for the viewing of trend data in one of three different ways.  Each of these ways is called a view.  The three views are:  Digital, Single Axis, or Multiple Axis.  The digital view allows for viewing, in numerical form, the current value of any variable being collected.  The Single Axis view allows for viewing all, or a subset of variables being collected on a common axis.  The multiple axis view allows for viewing each variable on its own axis.  The three different views are selected under the "view" menu item.

Each view of the data can be seen at the same time by using three different windows.  For example, if a digital display is currently being viewed, the "Window/New" menu item can be selected which will present a new window for that same data.  Then, a different view can be selected for that window from the "View \ Graph - Multiple Axes" menu and variables can be selected from the "View \ Set Trend Visibility" menu.  Thus two different views of the same data can be setup.  There is no limit to the number of different windows that can be setup for each document.

A G_Master document contains all the data collected for each variable added.  G_Master is a multiple document application, so that multiple G_Master documents can be opened at one time.  When you save a G_Master document into a .GMG file, you save the trend data, the configuration for each window opened, and the ranges and scales for each variable trended.  The name of the document loaded is displayed on each window, followed by a colon, and the window number.

### Setting Trend Visibility

Sometimes, it is not desirable to display all data collected on a G_Master document in a particular graph.  Thus, you must tell G_Master what variables you want to be displayed when a new window is opened.  You do this by choosing the "View/Set Trend Visibility" menu item.  When you select the "Set Trend Visibility" menu item, you will be presented with a list of variables available for trending.  You should select the variables you want to appear on the new window, and press "OK."  If a variable you want to trend is not in the list, it must be added to the document as described above.

### Changing Scales

The Scale menu item on G_Master allows you to change the time scale, or y-axis scale.  You must select from a series of pre-defined time scales, but have flexibility in choosing the y-axis scale.  If you want G_Master to automatically scale the y-axis based upon the data available, choose "Auto." If you want to enter the upper and lower limits for the y-axis, choose "Manual.".  If the current view is a multiple axis view, you will have to select the trend for which the new scale applies.

### Printing From G_Master

G_Master prints trend data via the Windows Print functionality. Thus, before G_Master can print, a Windows printer must be installed.  G_Master is capable of printing in black & white or color, to a local, or network printer, depending upon the Windows NT printer setup.  See the Windows NT documentation for installing printer drivers and printing to network printers.  Once the printer has been installed, you can print any G_Master window by selecting the window to be printed, re-sizing the window to get the desired aspect ratio, and pressing the print icon on the toolbar.  Alternatively, you can choose the "File/Print Preview" menu item to preview the printed image.  Printer settings are modified via the "File/Page Setup" menu item.  Some printers require the "Print Text as Graphics" option be selected in order to print properly.

### Security

The simulation managers and instructors have full use of G_Master.  Students, however, have limited access.  A student can execute the G_Master application, but the student is not allowed to view the information, or edit the information being collected.  Because of these limitations, G_Master will start up in a minimized state when initiated by a student.  The student will not be able maximize G_Master, or perform any other actions, except close.

# 8. Building an ACSL Macro

An ACSL macro contains lines of ACSL code that are substituted for every invocation of the macro in an ACSL program during ACSL translation from ACSL to FORTRAN; during this process, the ACSL statements are also sorted for proper sequence of execution.  In contrast, the lines of code in a subroutine must be in proper sequence of execution and the subroutine is compiled separately from the program in which it is invoked.  During substitution of macro code, the arguments in the invocation are used in place of the dummy arguments in the macro definition and in the macro code. Thus the macro allows code generation with variable names or parameter values that are defined online. For example, the superheater macro may be invoked to represent a primary superheater or a secondary superheater in a boiler model. The resulting FORTRAN code will have unique variable names (if the arguments themselves are unique) based on the invocation arguments. An ACSL macro may be invoked multiple times in a program. For error-free translation, the names generated should be unique and variables should not be defined more than once in the ACSL program.

## Macro Statements

The ACSL macro language is used to write an ACSL macro and is described in the chapter "Macro Language" of the ACSL Reference Manual **[2]**. A macro must begin with a macro definition header that usually includes one or more arguments, and must end with a MACRO END statement. The statements in the middle may be typical ACSL statements. Variable names formed from argument strings are specified as x&ID or x&ID&out where ID is the dummy argument that is substituted by the actual argument "ABC" at translation time. The resulting variable names generated during translation will be xABC or xABCout.

## Macro Arguments

The MMS Model Builder generates ACSL coding for each module by expanding the ACSL template that typically includes a macro invocation. The arguments of the macro invocation typically include the module ID and port stream names. When expanding the template arguments, the MMS Model Builder optionally allows a user-defined separator (default is an underscore _) that is prefixed to the expanded argument to make the generated names more readable (see the Model Builder menu "Simulation \ Code Generation Settings \ Variable Separator"). So the corresponding MMS template and expanded macro arguments will be <&ID> and "_ABC". The corresponding ACSL expansion names for the previous example will be x_ABC or x_ABC_out.

The ACSL macro language provides for several other useful features. For example, MACRO ASSIGN allows for a count of the number of arguments, and MACRO IF allows a check to verify if the user has supplied all necessary arguments or not. A MACRO GOTO allows for unconditional branching.

## Macro Example

An example of a macro is:

```
macro onezero(cond, Pid)

!determine if status is 1 or 0
!cond - condition string for one (integer)
! Pid - status ID string

!invocation: onezero ("pos_a*pos_b", "_XtoY")

integer m&Pid
if (&cond .ge. 1) then
  m&Pid = 1
else
  m&Pid = 0
end if

macro end
```

## References (available online)

[1] ACSL/Windows Users Guide, AEgis Technologies, Huntsville, AL.

[2] ACSL Reference Manual, AEgis Technologies, Huntsville, AL.

[3] Watcom FORTRAN User Guide (CD-ROM) (online documentation) and Language Reference (f77LR.hlp).

[4] MMS Graphical Development Tools User’s Guide (MMS Model Builder and MMS CompGen), nHance Technologies, Lynchburg, VA, 1996 (Revised 2005).

[5] MMS Reference Manual, nHance Technologies, Lynchburg, VA, 2005.
(MMS Methodology and Basis)

[6] MMS Basics Manual, nHance Technologies, Lynchburg, VA, 2005.

(Part 1 – Generic Formulation of MMS RTC Modules)

(Part 2 – Help Documentation for MMS Modules)

(Appendix A – Selection and connectivity of MMS Modules)

(Appendix B – Gas and Steam Property Routines Used in MMS)

(Appendix C – Possible Error Messages During Model Building)

[7] ACSL Beginners Guide, AEgis Technologies, Huntsville, AL.

[8] Compaq Visual Fortran online documentation (DFMain.chm)

[9] nHance Help System online documentation

# Appendix A: Example of an MMS Model

## Scope

The example model DEAEX consists of a deaerator system. The scope and configuration of the model is determined from the P&I diagrams of the plant (Figures A-1 through A-3). Figure A-1 shows the schematic of the deaerator feedwater supply from the condensate pumps to the deaerator. Figure A-2 shows the schematic of the deaerator steam supply from the IP turbine exhaust to the deaerator. Figure A-3 shows the schematic of the BFW supply line from the deaerator to the HP heater boundary condition through the BFP's. Based on the purpose of the model, it is decided to:

- **Lump the three condensate pumps into one;**
- **Represent the gland condenser, drain cooler, and the LP heaters by a heated pipe**
- **Include two valves for condensate feed into the deaerator (one for normal use and one as a standby);**
- **Include one valve for steam supply;**
- **Lump the three BFW pumps into one.**

Controllers are to be provided for deaerator level and pressure control. The hotwell outlet, IP turbine exhaust, and HP heater FW inlet are the boundary conditions.

## Model Development Preliminaries

Open the Model Builder and select the appropriate Prolog/Epilog pair (MMS-RTC for this example). Review the Project Settings.

## Model configuration

To configure the model in the MMS Model Builder, it is necessary to know which modules will be used. In this example, it is decided to use the RTC and control modules. The modules are grouped functionally in the MMS Model Builder to facilitate their selection. Figure A-4 shows the deaerator system model. The feedwater entering the deaerator comes from a hot well through a condensate pump, a pipe representing LP heaters, and two FW valves in parallel, LPFWV and LPFWBV. LPFWV is normally used for deaerator level control. LPFWBV is a bypass valve identical to LPFWV that is manually operated and is normally closed. If LPFWV should fail, LPFWBV may be manually controlled to stabilize the deaerator level.

Steam for deaeration is obtained from turbine extraction through a valve XTRV. The feedwater discharged from the deaerator is pumped by the boiler FW pump BFWP into the HP heater boundary condition.

The deaerator level is controlled by a PI controller LC using a 3-element control. The deaerator pressure is controlled by a PI controller PC. A CODE block is used to calculate the heat added by LP heaters, pump speed change dynamics, and the saturation margin at BFWP inlet.

## Data Input and Auto-Parameterization

Next the data for model parameterization is collected. A validated (and therefore consistent from mass and energy balance standpoint) heat balance diagram (Figure A-5) is required to input the operating parameters of the system (e.g., pressure, flow, specific enthalpy). Valve capacity, actuator time constant, pump characteristic, level and pressure control setpoints and other geometric information called in the relevant AP forms are also required. At this point, data are entered in the

AP form for each module, and the Auto-Parameterize (AP) button is clicked. This provides a running check on the validity of data entered and the results may be useful in entering data in other modules.

### Build Model

Next the "Build Model" button is clicked to translate, compile and link the model to create the executable model (.prx file). If necessary, the model configuration is corrected to remove any translation, compilation and link errors.

### Run Model

A starter command file is created when the model is saved for the first time. This command file is edited to add "out", "prep", and other commands and "proced"s (command macros). The model is executed to analyze various scenarios, e.g., level setpoint change, pump speed change and trip, actuator malfunction, controller tuning, etc.

Depending on the choice of the Prolog/Epilog pair and the Project Settings, the model can be run in the ACSL Runtime or the SimTools environment.

### Suggested Exercises

The example model can be built progressively as a series of exercises included in the MMS installation directory (..\MyMMS\Models\Examples\MMS Training\RTC\ ):

- **Exercise 1 – Scope model and collect data**
  - Determine the scope and content of model
  - Get heat balance data and component characteristics
  - Verify the data to ensure correct mass, energy, and momentum balance
- **Exercise 2 – Configure model**
  - Familiarize with different prolog/epilog pairs and select the appropriate pair
  - Review project settings
  - Select and place modules
  - Enter module IDs
  - Connect
  - Annotate
- **Exercise 3 – AP each module**
  - Enter input data
  - AP
  - Verify ACSL template

- **Exercise 4 – Build model**
  - Save model and click "Build Model" button. If necessary, resolve any translation, compilation and link errors until the model build is successful.
  - Examine translator messages for any "variables used but not defined" warnings and resolve these either in the model configuration or in the command file.
  - Examine .for and .inc files to familiarize with translated program structure

- **Exercise 5 – Execute model**
  - Click "Execute Model" button
  - Type command: START; S TSTOP=10.; CONTIN
  - See if model runs without numerical errors or problems and resolve these as necessary with or without a debugger.
  - Type the following command and see the debug dump: S NDBUG=1 ; CONTIN

- **Exercise 6 – Run model in ACSL Runtime environment**
  - Edit command file as necessary
  - Run to steady state (with the NDBUG command, examine derivatives of dynamic states to be sure)
  - Verify with heat balance data, remove anomalies
  - Use command ANALYZ /EIGEN to calculate eigenvalues and see if any have positive real parts (indicating instability), determine integration step size
  - Save the IC with command SAVE /F = ‘SS1.SAV’
  - Add command macros (PROCEDs) in .cmd file to restore IC (RESTOR /F = ‘SS1.SAV’), run transients and plot
  - Run transients
  - Add a rectangle with text in the model configuration and copy the .cmd file in the "Edit Text" window

- **Exercise 7 – Run model in SimTools environment**
  - Copy RTC6.MMS, RTC6.CMD file from Ex. 6 into a new directory named EX7. Rename RTC6.MMS to RTC7.MMS, rename RTC6.CMD to RTC7.CMD.
  - Open RTC7.MMS, select the "MMS – RTC w/ Simulator Link" prolog/epilog, save model
  - Go to "Project Settings \ Project Options" and click "Set Standard SimTools Options" button
  - Click "Build Model" button and make sure model runs in the ACSL Runtime environment without obvious problems.

- Start S_Master and configure it for this model
- Practice S_Master operation:
  - Definitions
  - Starting/Stopping
  - Hard Stop
  - Single Step
  - Data Images
  - Saving Data Images
  - Loading Data Images
  - Fast/Slow Time execution
- Practice G_Master Operation
  - Adding a variable to trend
  - Views
  - Single Axis
  - Multiple Axis
  - Digital Display
  - Scale
  - Time Scale
  - Y-Scale
  - Views vs Documents
  - Trend Visibility
  - Saving Documents
  - Exporting Data
- Practice Action Center Operation
  - Action Variables (Definitions)
  - Actions (Definitions)
  - S_Master functionality built into Action Center
  - Start/Stop, Save/Load Data images
  - Configuration

  - Runtime Speed
  - Timed Backtrack Configuration
  - Creating Action Variables
  - Creating Actions
  - Types of Actions (LOA, Malfunction, etc.) Single Action
  - Ramp
  - Triggered Conditions
  - Multi Action
- Linking with Action Center from G_Master
- Linking with G_Master from Action Center

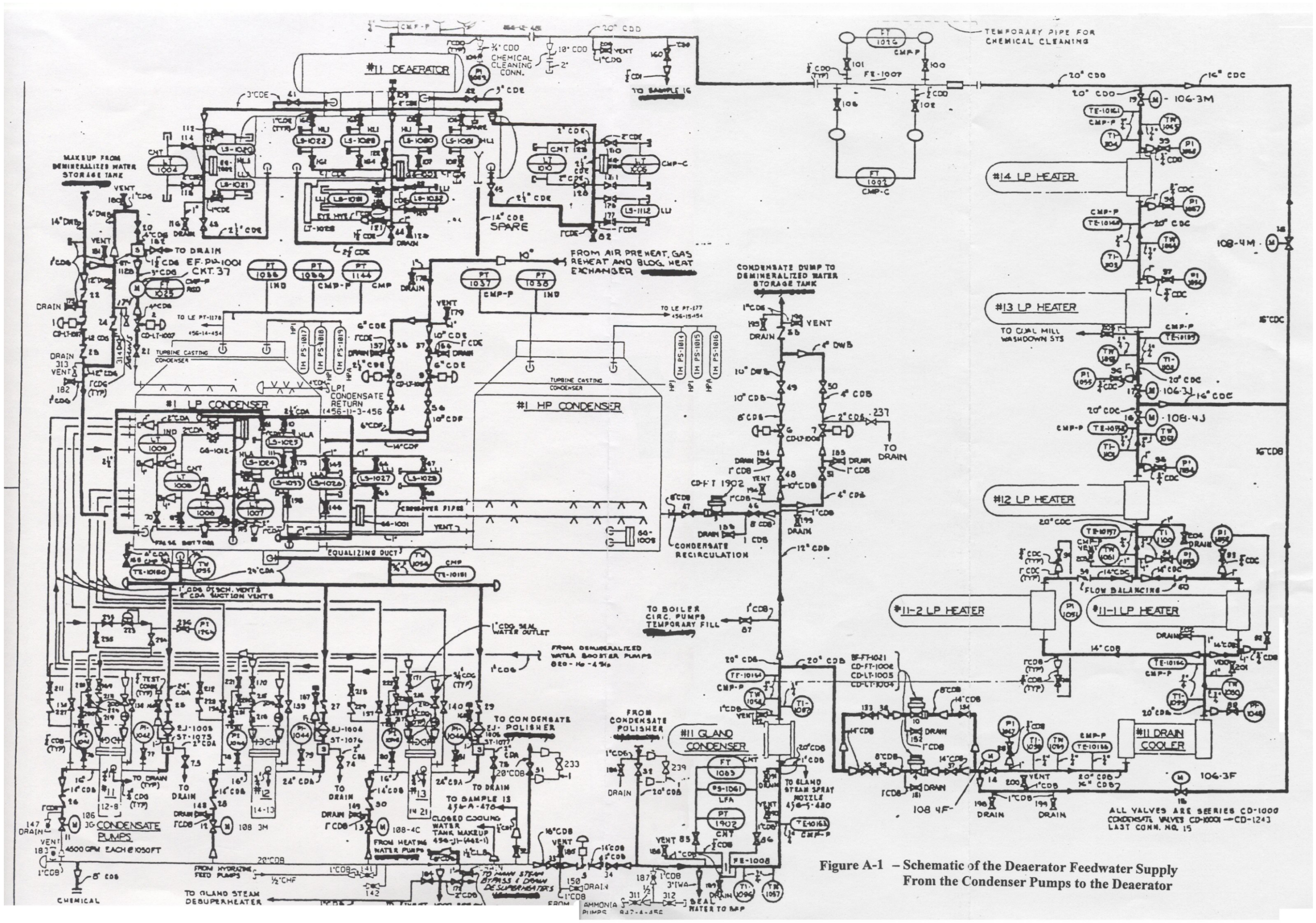

#11 DEAERATOR
CHEMICAL CLEANING CONN.
TO SAMPLE 16
TEMPORARY PIPE FOR CHEMICAL CLEANING
MAKEUP FROM DEMINERALIZED WATER STORAGE TANK
14" CDE SPARE
FROM AIR PREHEAT, GAS REHEAT AND BLDG. HEAT EXCHANGER
CONDENSATE DUMP TO DEMINERALIZED WATER STORAGE TANK
#14 LP HEATER
#13 LP HEATER
TO COAL MILL WASHDOWN STS
#12 LP HEATER
#11-2 LP HEATER
#11-1 LP HEATER
FLOW BALANCING
#1 LP CONDENSER
#1 HP CONDENSER
LPI CONDENSATE RETURN (456-11-3-456)
CROSSOVER PIPES
EQUALIZING DUCT
CONDENSATE RECIRCULATION
TO BOILER CIRC. PUMPS TEMPORARY FILL
FROM DEMINERALIZED WATER BOOSTER PUMPS
TO CONDENSATE POLISHER
FROM CONDENSATE POLISHER
#11 GLAND CONDENSER
#11 DRAIN COOLER
CONDENSATE PUMPS
4500 GPM EACH @ 1050 FT
CLOSED COOLING WATER TANK MAKEUP
FROM HEATING WATER PUMPS
TO MAIN STEAM BYPASS & DRAIN DESUPERHEATERS
TO GLAND STEAM DESUPERHEATER
FROM HYDRAZINE FEED PUMPS
SEAL WATER TO BFP
AMMONIA PUMPS
TO GLAND STEAM SPRAY NOZZLE
ALL VALVES ARE SERIES CD-1000
CONDENSATE VALVES CD-10001 — CD-1243
LAST CONN. NO. 15
Figure A-1 – Schematic of the Deaerator Feedwater Supply From the Condenser Pumps to the Deaerator

**Figure 20:** Schematic of the Deaerator Feedwater Supply

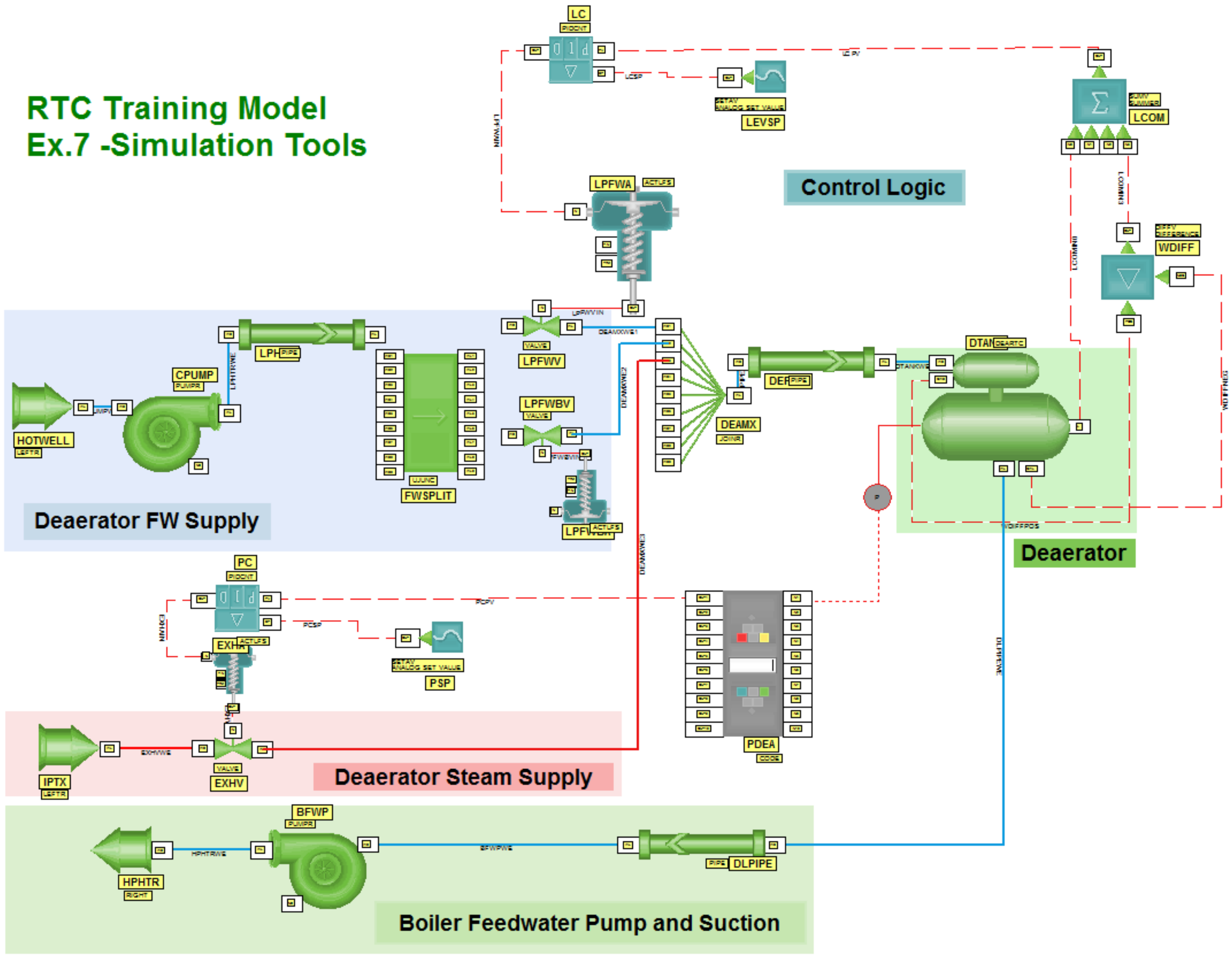


**Figure 21:** MMS Diagram of the Example Model

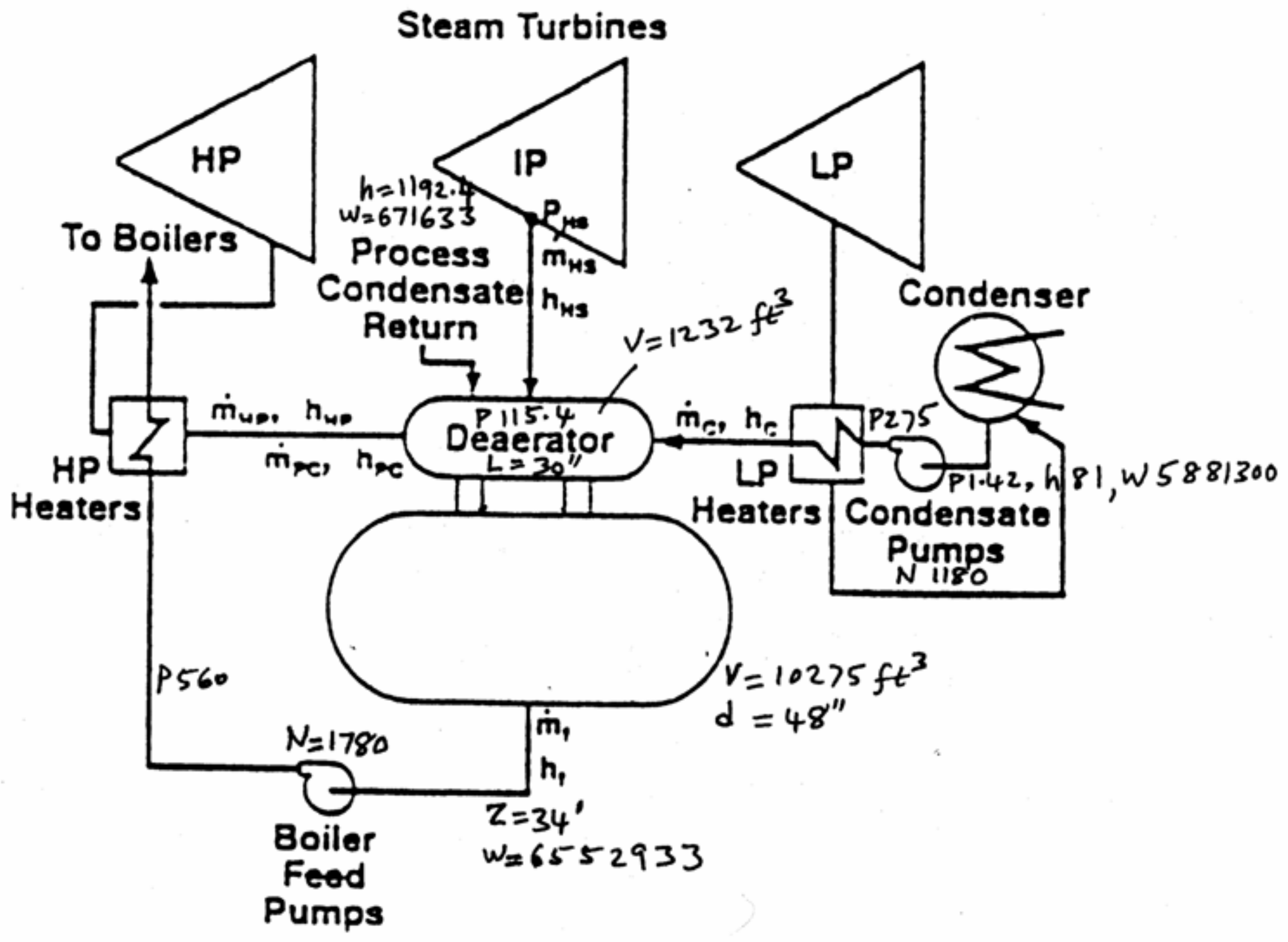


Figure A-5 – Heat Balance Diagram

**Figure 22:** Heat balance diagram

# Index