Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR TP 96033

The national simulation facility NSF: the application of the real-time simulation support tool PROSIM

W. Brouwer, A.A. ten Dam and P. Schrap

DOCUMENT CONTROL SHEET

	ORIGINATOR'S REF.		SECURITY CLASS.
	TP 96033 U		**Unclassified**

ORIGINATOR
National Aerospace Laboratory NLR, Amsterdam, The Netherlands

TITLE
The National Simulation Facility NSF: The application of the real-time simulation support tool PROSIM

PRESENTED AT
CEAS Symposium on Simulation Technology "Making it Real", 30 October - 1 November 1995, Delft, The Netherlands

AUTHORS	DATE	pp	ref
W. Brouwer, A.A. ten Dam, P. Schrap	**960119**	**23**	**7**

DESCRIPTORS

Architecture (computers)	**Real time operation**
F-16 aircraft	**Scheduling**
Flight simulators	**Software development tools**
Flight simulation	**Tasks**
Graphical user interface	

ABSTRACT
This paper describes the design of the Programme and Real-time Operations SIMulation support tool PROSIM and its application to the National Simulation Facility NSF, a research and development facility at the Netherlands National Aerospace Laboratory NLR for realistic man-in-the-loop (i.e. aircraft pilot) simulations, possibly with aircraft hardware-in-the loop.
In particular this paper describes the use of PROSIM at the realisation of a full-mission F-16 MLU flight simulation on NSF. PROSIM is a generic simulation tool that can be used for design, verification, test and training of a wide range of applications, such as aircraft and ground vehicles. In order to optimise working conditions and to satisfy specific requirements that must hold in each stage of design, development, testing and operation, PROSIM consists of two distinct, but related tools: the Simulation Development Software (SDS) and the Real-Time simulator Software (RTS). The SDS is a generic software environment used to develop and test simulation models and hardware drivers, and prepare data files that are used during real-time simulations runs. The RTS performs timing control of the simulation tasks (simulation models and hardware drivers), and user control of the complete simulation run. PROSIM contains a Graphical User Interface for ease of access and use.
PROSIM is operational at NLR's Flight Divison since June 1994. Furthermore, PROSIM is used at NLR's Space Division and at NLR's Informatics Division. The wide range of applications shows that PROSIM is a generic simulation tool.

Abstract

This paper describes the design of the Programme and Real-time Operations SIMulation support tool PROSIM and its application to the National Simulation Facility NSF, a research and development facility at the Netherlands National Aerospace Laboratory NLR for realistic man-in-the-loop (i.e. aircraft pilot) simulations, possibly with aircraft hardware-in-the-loop.
In particular this paper describes the use of PROSIM at the realisation of a full-mission F-16 MLU flight simulation on NSF. PROSIM is a generic simulation tool that can be used for design, verification, test and training of a wide range of applications, such as aircraft and ground vehicles. In order to optimise working conditions and to satisfy specific requirements that must hold in each stage of design, development, testing and operation, PROSIM consists of two distinct, but related tools: the Simulation Development Software (SDS) and the Real-Time simulator Software (RTS). The SDS is a generic software environment used to develop and test simulation models and hardware drivers, and prepare data files that are used during real-time simulation. The RTS is a generic software environment for execution and control of (non) real-time simulation runs. The RTS performs timing control of the simulation tasks (simulation models and hardware drivers), and user control of the complete simulation run. PROSIM contains a Graphical User Interface for ease of access and use.

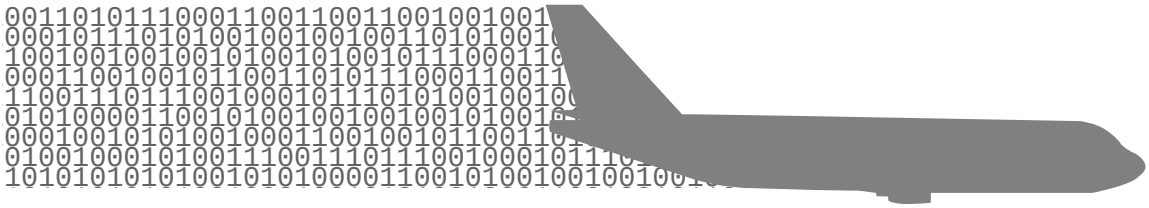
PROSIM is operational at NLR's Flight Division since June 1994. Furthermore, PROSIM is used at NLR's Space Division and at NLR's Informatics Division. The wide range of applications shows that PROSIM is a generic simulation tool.

Symbols and abbreviations

AOCS	Attitude and Orbit Control Systems
DFAD	Digital Feature Analysis Data
DMA	Defense Mapping Agency
DTED	Digital Terrain Elevation Data
FSIS	Flight Simulator Interface System
GUI	Graphical User Interface
ICP	Integrated Control Panel
ITEMS	Interactive Tactical Environment Management System
MLU	Mid Life Update
MFD	Multi Function Display
NLR	Netherlands National Aerospace Laboratory NLR
PROSIM	Programme and Real-time Operations SIMulation support tool
RTD	Real-Time Development
RTE	Real-Time Experiment
RTS	Real-Time simulator Software
SCRAMNet	Shared Common Random Access Memory Network
SDS	Simulation Development Software
TVE	Test and Verification Equipment

Contents

This page is intentionally left blank



Making it REAL

THE NATIONAL SIMULATION FACILITY NSF: THE APPLICATION OF THE REAL-TIME SIMULATION SUPPORT TOOL PROSIM

BY

W. BROUWER, A.A. TEN DAM AND P. SCHRAP

NATIONAL AEROSPACE LABORATORY NLR
AMSTERDAM, THE NETHERLANDS

CEAS SYMPOSIUM ON
SIMULATION TECHNOLOGY

October 30, 31 and November 1, 1995

1 Introduction

In this paper we describe the design and use of PROSIM during the development and operation of NSF, a generic simulation tool developed at NLR that can be used for design, development, verification, test and training. PROSIM offers a user friendly environment for research and development of a wide range of aspects of realistic man-in-the-loop simulations.

PROSIM originated as part of the National Simulation Facility NSF at the Netherlands National Aerospace Laboratory NLR. The prime purpose of the NSF is to provide a research and development facility for realistic man-in-the-loop (i.e. aircraft pilot) simulations, possibly with aircraft hardware-in-the-loop. The tasks users of NSF have to perform range from performing feasibility studies, possibly including development of new simulation models, to realistic man-in-the-loop simulations, possibly including several control stations and an experiment controller. It was identified that a generic software tool, i.e. PROSIM, is essential for successful realisation and operation of NSF, as well as easy incorporation of already envisaged future extensions.

Due to the stringent safety requirements imposed on simulations with a human-in-the-loop and/or hardware-in-the-loop, e.g. an aircraft pilot in case of NSF, the simulation models must be tested thoroughly before these are used in the actual simulator. In order to optimise working conditions and to satisfy specific requirements that must hold in each stage of design, development, verification and testing, PROSIM consists of two separate tools: the Simulation Development Software (SDS), and the Real-Time simulator Software (RTS). The SDS is a user-friendly simulation environment to develop and test simulation models and hardware drivers, and prepare data files. The RTS is a generic software environment for execution and control of (non) real-time simulation runs.
At NSF PROSIM is being used at the realization of a reliable and high-fidelity F-16 Mid-Life Update full mission simulator. This simulator utilizes ‘state of the art’ commercially available h/w and s/w where the development engineer has numerous tools at his disposal to accomplish efficiently and cost-effectively the function of software development, verification and validation.

The remainder of this paper is organized as follows. In section 2 the design approach for PROSIM is described, with emphasis on the stages in software development and on genericity. Section 3 is devoted to the description of NSF that is using PROSIM since june 1994. This section provides an overview of the system complexity that PROSIM supports. Section 4 and 5 focus on the SDS and RTS with emphasis on the NSF application. The experience with NSF motivates the necessity of a generic simulation software tool. Finally in section 6 the conclusions are stated.

2 Prosim: a generic simulation tool

The design of PROSIM (Ref. 1) is based on a concept that is used at NLR for the definition, development and usage of simulators. This concept is based on NLR's experiences during several years of building and using simulators, such as the flight simulator at NLR's Flight Division (Ref. 2). The key element of the concept is the notion of genericity (Ref. 3). The use of this concept results in a clear separation of the application specific part and the simulation tool, and in a clear task and responsibility allocation of co-workers within a simulator development project, leading to a faster and cheaper development cycle.

Currently three application projects at NLR make use of PROSIM:
- F-16 MLU full-mission flight simulation (NSF),
- Test and Verification Equipment (TVE),
- Design and development of multibody systems (ISMuS).

The Test and Verification Equipment (TVE) which is currently being developed under ESA contract (Ref. 4), will serve as a generic test and verification tool to support design, integration and test activities of Attitude and Orbit Control Systems (AOCS) systems. During different phases of the AOCS design, development and test life cycle, different aspects of spacecraft attitude control are studied. Therefore, different levels of detail of the simulations are required. Relevant for the present paper is the Test Software (TSW) of the TVE project. The TSW must be structured, modular and flexible as it used to the meet various requirements of the different phases. PROSIM offers the functionality required for a large part of the TSW. Use of PROSIM reduces the costs of the design, testing and verification process of TSW, and thus TVE itself. The existence, and reuse of PROSIM also reduces the work of the future TVE programmer to the specification of models of the spacecraft dynamics and functionality of units.
The ISMuS project at NLR's Informatics Division concerns a computer based environment for Computer Aided Control Engineering. Tasks performed in this field are: modelling of dynamical systems, design of controllers for dynamical systems, and system studies. Here PROSIM is used as simulation tool for the analysis of all of these aspects. The F-16 MLU full-mission flight simulation (NSF) application will be detailed in the next sections.

One design driver of PROSIM is that the freedom for users to choose his simulation models is essential. It is for example valid to have two simulation models of the same component, or one model implemented in software whereas the other one is implemented in hardware. Ease of switching between simulation models, integration of submodels to one application model, and integration of software with hardware to build simulators of ever increasing complexity, is important for the acceptance of a simulation tool by the user. In one stage the details can be important to support the assessment of the ultimately obtainable accuracy. In another stage only the global functional behaviour may be of interest in the context of the co-operation of

components. Another design driver of PROSIM is the ability to deal with the presence and the absence of flight hardware and software. As parts of the equipment can be absent or present the statement “the flight software must not be aware that it is operating on the ground instead of in space or in the air” is as valid as the statement “a simulation model should not need to know whether it gets data from another simulation model, or a real piece of equipment”.

PROSIM supports the following stages in the usage of a simulator (Ref. 5):
-Simulation model selection, or development of new simulation models along the software development guidelines. Note: a simulation model may contain several sub-models.
-Simulator executable composition. Integration of the simulation model with the tool software to create a simulator executable.
-Simulation preparation. Preparation of a simulation by specifying the input data.
-Simulation run execution. Execution of an actual simulation run: real-time with possibly a human- (or hardware)-in-the-loop or in non-real-time (online or batch execution).
In the present paper emphasis is on the simulation tool used at NSF.

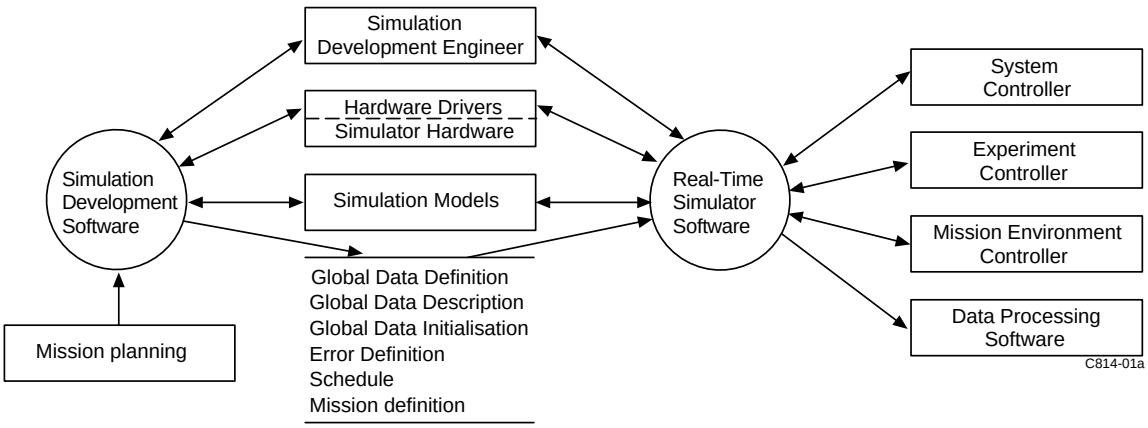


Fig. 1 Top level diagram depicting the relation between the basic tools of PROSIM: RTS and SDS

At the realisation of a simulation two major activities are distinguished: the actual real-time simulation in various (hardware and software) configurations, and the development of the simulation models used in real-time simulation. Moreover, these activities are to be performed simultaneously. Consequently, in order to optimise working conditions and to satisfy specific requirements that must hold in each stage of design, development and testing, PROSIM consists of two separate, but related tools (see figure 1):
- the Simulation Development Software (SDS), and
- the Real-Time simulator Software (RTS).

The SDS is used to develop and test simulation models and prepare data files that are used during real-time simulation by the RTS. Stringent safety requirements imposed on simulations with a human-in-the-loop and/or hardware-in-the-loop require that simulation models must be tested thoroughly. For this, the SDS offers a user friendly environment. The use of SDS at NSF is further detailed in section 4.

The RTS is a generic software environment for execution and control of (non) real-time simulation runs, using a Graphical User Interface. The RTS is used already in an early stage of the development of simulation models, as it is very important to test the scheduling of simulation models as early as possible in the development process. The use of RTS at NSF is further detailed in section 5.

PROSIM has been designed and documented according to the NSF Quality Standards. PROSIM is operational, and in use at the Flight Division, since June 1994. The present version of PROSIM supports FORTRAN 77 simulation models. Future versions of PROSIM will also allow for simulation models coded in C.

3 The Netherlands National Simulation Facility NSF

3.1 Introduction

The National Simulation Facility (NSF) has been developed such that a very wide range of applications will be possible in the near future. Applications such as the simulation of a manned helicopter, civil and military aircraft as well as road vehicles and water vehicles are foreseen. NSF includes many unique features which greatly improve reliability, maintainability and growth capability. The first application of NSF is F-16 MLU simulation in a full mission environment. Figure 2 shows the interfacing between the (large) systems necessary to accomplish this realistic simulation.

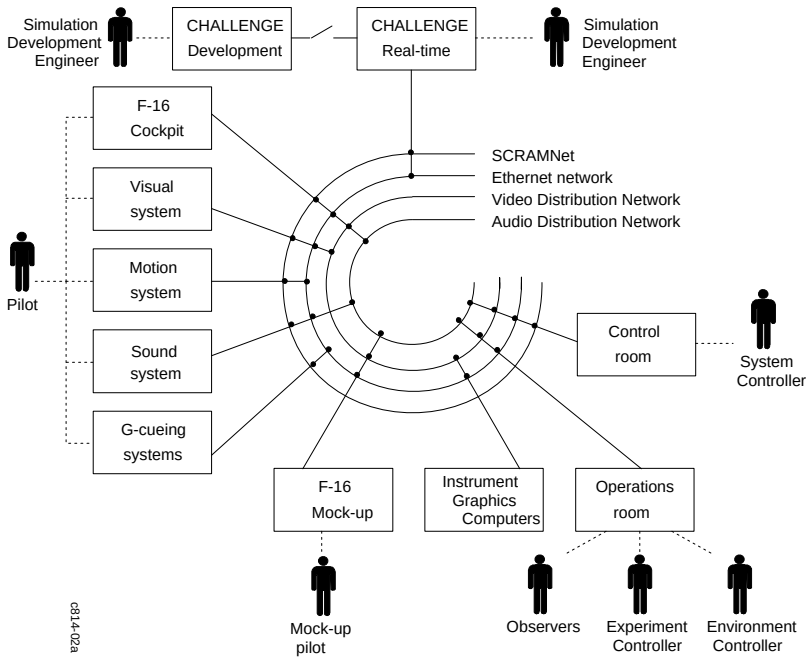


Fig. 2 NSF architecture

PROSIM, hosted on the Silicon Graphics Challenge computer, is the 'heart' of the simulation and dedicated to perform the control of the simulation and scheduling of several tasks necessary for real-time full mission simulation. The tasks can be divided into two different types. Tasks can be hardware drivers that interface with the various systems connected to the host computer, or simulation models performing the computations needed for the application. These simulation models and hardware drivers require specific scheduling and interfacing facilities. The hardware drivers are used to interface with hardware systems such as: Control desk, Cockpit, Motion system, G-cueing system, Visual system, Sound system, Graphics computers, and Mock-up (Refs. 6,7).

3.2 NSF users

The complexity of NSF is emphasized by the many different users that are involved. There are at least six users, each having different but similar interfaces. These users can be characterized as follows.

Simulation development engineer: The simulation development engineer uses PROSIM for the creation, debugging, verification, validation and testing of simulation software required to perform one or more simulation runs. The simulation development engineer must have detailed knowledge of most of the systems utilized in simulation.

Pilot: In the cockpit the pilot has interfaces providing all the cues to perform the tasks demanded by the project. These cues are provided by the motion system, the g-cueing system, the sound system, the visual system and the cockpit controls and instruments (including the avionics).

System controller: The system controller is responsible for a safe operation of the simulator. The control desk provides him all displays and controls necessary to perform this task. If required the system controller can overrule the commands of the experiment controller.

Experiment controller: The experiment controller controls and monitors the correct execution of his experiment. His interface exists primarily of displays and controls available at a control station in either the control room or the operations room. The displays that are monitored give a graphical overview of the simulation run, and include all displays and outside view of the pilot. The experiment controller can control the progress of simulation run as far as authorized by the system controller.

Environment controller: The environment controller is responsible for the execution of a predefined scenario, as far as the environment outside the aircraft is concerned. The environment controller has control over elements of the environment as threats and targets, but also atmospheric elements and navigation aids.

Observers: The observers are passively observing the progress and execution of the simulation experiment. They have the opportunity to view almost all graphical displays the environment controller and experiment controller have at their disposal.

In all states of the simulation, the development engineer and the system-, environment- and experiment controller can monitor all variables used by the simulation models and hardware drivers. Above all the development engineer is also privileged to change data, as this facility has been proven extremely important to expedite testing the simulator.

3.3 NSF simulator hardware

The simulator systems used to perform a F-16 full mission simulation are three computing systems, a control desk, a cockpit, a motion system, a g-cueing system, a visual system, a audio system, several graphics computers and a mock-up. The cockpit and the mock-up are F-16 MLU specific hardware where all other systems are applicable to simulate other aircraft as well. Next these systems will be described briefly. More detailed information can be found in Ref. 7.

Computing systems. The NSF's computing capability consists of three Silicon Graphics systems - two dedicated real-time systems and one software development system. All the Silicon Graphics Challenge L have four R4400 MIPS processors, with a total computing capacity of 180 MWhetstones, and can be expanded to eight processors. For interfacing these computers have an Ethernet adapter and a SCRAMNet interface. The advanced fibre-optic-based Flight Simulator Interface System (FSIS) connects computing systems and simulator hardware systems via SCRAMNet in a ring-like fashion.

Control desk. Situated in the control room, the control desk is used to control and monitor the simulation. Also the control desk can function as a control station for the environment controller as well as the experiment controller. The control desk gives a status overview of the simulation systems and provides all information necessary to assure that the system controller can monitor and control the simulation. Facilities available are full duplication of cockpit instruments and outside view, recording facilities as video recorders, multi-channel recorders, colour displays and colour plotters. Via a video distribution network all video signals used during a flight simulation run can be distributed, providing copies of video signals wherever needed.

Cockpit. The F-16 cockpit is a section of a real F-16 aircraft with several adaptations related to the simulation environment it has to operate in which differs from the in-flight situation.

The cockpit provides the pilot an interface with the aircraft systems, the avionics systems, the outside world environment through the visual display system, the audio system, the motion system and g-cueing system. For the pilot the instruments and controls in the (simulator) cockpit are identical to the instruments and controls in the real F-16 MLU aircraft.

Motion system. The six-degrees-of-freedom high-performance motion platform system has been made by Hydraudyne and is not only for simulating fast jets (as the F-16) but also for simulating motions of rotary wing aircraft and fixed wing transport aircraft. The actuators are fully hydrostatic and have a phase lag of 45 degrees at 4 Hertz.

G-cueing system. The g-cueing system shall consist of a g-seat system, an anti-g suit system, a helmet loader and a lapbelt restrictor. Currently this system is not available until 1996.

Visual system. The visual system provides the pilot with an outside view using an Evans & Sutherland visual system. This system includes a three channel ESIG-3000 AT/GT image generator and a VistaView head-tracked projection system. This system projects an oval shaped 120^0H x 90^0V background with a high-resolution 40^0H x 30^0V area-of-interest inset, which are both head-tracked to allow nearly 360^0 field-of-regard. This outside view is projected on a 17 ft dome mounted on the six degrees-of-freedom motion platform. Apart from the several geo-specific large area databases that are available (e.g. U.S. Hunter-Liggett area), there is a capability for developing customer-specific databases and to import DMA DTED/DFAD compatible data in combination with photographs for geo-specific terrain simulation.

Audio system. The audio system enhances the realism of the simulation by generating digital audio and special effects. The CHeSS/Paradigm Emax III sound simulation system provides the pilot with aural queues including "in-cockpit sampled" F-16 aerodynamic hiss and engine

sounds, voice messages, missile launching, gunfire and radar warning receiving tones. These sounds can be generated independently and simultaneous. A development station is available for modifying or creating sound libraries.

Graphics computers. The graphics computers support the simulation by generating the displays for the multi function displays and head-up displays in the cockpit. Furthermore these computers generate displays for the operations room from where the simulation can be monitored by providing tactical overviews and a graphical presentation of the cockpit instruments (the graphical as well as mechanical instruments).

Mock-up. The F-16 mock-up provides the basic functionality of an F-16. It is equipped with four colour CRTs (three head down and one head up), touch screens and F-16 controls and utilizes also the simulator software on one of the real-time Challenge computers. Apart from being a player in the full mission simulation, this facility also offers an extremely powerful tool for testing and verifying F-16 simulation software.

3.4 NSF application software

As NSF is a research facility most of the systems are simulated by software, although in some cases emulation might be possible and satisfactory. This implicates the development and execution of a vast amount of software, preferably in real-time. As shown in figure 1 the simulation software hosted on the real-time computer consists of simulation models software and hardware drivers software. Furthermore on all other systems shown in figure 2 a large amount of software is developed and used. The simulation models characterizing the F-16 full mission simulation can be described by the following groups:

- NSF hardware systems,
- F-16 simulator systems,
- F-16 airframe,
- F-16 aircraft systems,
- F-16 avionic systems,
- F-16 weapons,
- Aircraft (mission) environment,
- General simulation models.

An overview of the main components involved at each group that are all programmed in the software language FORTRAN can be found in appendix A.

Besides newly developed simulation software, software also originates from the existing simulation facility and from other institutes outside the NLR. The simulation models of the airframe, some of the aircraft systems, part of the aircraft environment and the general simulation models has been rehosted from the Concurrent MicroFive, the host computer that is replaced by the Challenge at the Flight Simulation Department of NLR in the near future. These simulation models are proven and familiar. This software had to be adapted according the programming rules and interface requirements applied to the Challenge computer system.

The software for the avionic systems has been provided by Lockheed Flight Simulation Department. This software had been running on Harris computer systems implicating that several features unique for these systems had to be converted to equivalences on the Challenge computer system. As the avionics components can be exchanged by their hardware counterparts these simulation models require a strict scheduling of 50 Hz. Together the simulation software running on the real-time host computer is using approximately 450 COMMON blocks encompassing 7000 variables and using 1.2 MBytes of memory.

4 Software development in NSF

The Simulation Development Software (SDS) is an environment for the efficient development and modification of simulation software and data files. These simulation software and data files are required for performing a simulation run. Simulation software refers to a set of simulation models and hardware drivers. The simulation models are used to simulate the characteristics of a particular vehicle (the F-16 MLU for NSF), while the hardware drivers are used to drive the connected hardware systems (motion, visual, F-16 MLU cockpit, sound, g-cueing, mock-up and control room). Data files refer to a set of files used for configuration of the Real-Time simulator Software (RTS) and initialisation of the Global Simulator Data (GSD) memory.

The primary user of the Simulation Development Software (SDS) is the simulation development engineer. This engineer interacts with the SDS using a Graphical User Interface. The most important component of the SDS is Build Simulator. Except for the Mission Definition file, all files in NSF are edited by using a general purpose editor (vi). The output of the SDS is defined as the complete set of simulation software and data files required by the Real-Time simulator Software (RTS), also identified as the simulator configuration. All the simulation software and data files are specified in a human-readable (ASCII) form and must be compliant to the predefined PROSIM format and syntax. This PROSIM format is described in detail in the Interface Control Document of the RTS and the SDS. The most important simulation software and data files are described below.

Simulation Software: The Simulation Software is defined as a set of simulation models and hardware drivers to be used by the RTS. These simulation models and hardware drivers as described in paragraph 3.4 comply to the PROSIM format with respect to the definition of the used constants and variables from the GSD memory, the definition of simulation tasks used by the scheduler and the syntax of the used programming language. At this moment NSF only uses the FORTRAN programming language. In the near future PROSIM will also support the use of the C programming language.

Global Data Definition: One or more Global Data Definition files define the exact contents and structure of the GSD memory. This GSD memory is the shared memory segment used for the communication between all simulation models, hardware drivers and different components of the RTS. For NSF it was decided that all variables and constants used in the simulation software are defined in the global data definition files. This gives the development engineer a powerful tool for debugging the simulation software. In general the Global Data Definition file contains the following types of information.

- Segment, Group and Block names.
- Symbolic names of constants and variables, including array dimensions and number of characters.
- Type of constants and variables.
- Physical unit of constants and variables.
- Default, Constant, Minimum and Maximum values.
- Description of the Segment, Group, Block, Constant or Variable.
- Additional comment(s).

Within NSF one segment is used. The groups are related to the simulation software organization as defined in 3.4., where for each simulation software group the first character of the group name is a fixed character as defined by the programming rules of the Flight Simulator Department at NLR. This gives each simulation software group the opportunity to use more global data definition groups and thereby providing more independency of the software within a simulation software group. Furthermore each global data definition group has a set of global blocks were within NSF by means of defining the global block names an additional level is introduced. This implicates that each simulation software group is divided into several subgroups where each subgroup is having its own set of GSD blocks.

Schedule: The Schedule file defines the scheduling requirements for the simulation tasks to be scheduled by the Scheduler of the RTS. A description of the scheduling requirements is given in section 5.

Error Definition: The Error Definition file contains the definition of all errors, which may be used during a simulation run. These errors include errors from the generic RTS components, as well as the simulation models and hardware drivers. For each error specified in the Error Definition file the name, severity (notice, warning, error or interactive error), category, description, and question are given.

Global Data Initialisation: The Global Data Initialisation files are used to assign values to the parameters and variables in the GSD memory. The Global Data Initialisation files are defined as a hierarchical tree of files, containing references to lower level files or data for initialisation of parameters or variables in the GSD memory. Figure 9 shows the highest level of data files in NSF. The data on the lowest level of the Global Data Initialisation files contains: Symbolic name of the parameter or variable (in the GSD memory), Physical unit, Value for parameter or variable, and Optional comment. Using the Global Data Initialisation files the GSD variables arranged as lists, arrays and other data structures can be initialised in a convenient way.

Mission Definition: The Mission Definition file contains the definition of all events, stimuli and data recording scripts, which must be executed by the generic RTS components during the performance of a simulation run. This Mission Definition file provides an extremely useful tool to separate the mission dependent actions from the simulation software functionality during the experiment execution as well as the debugging phase.

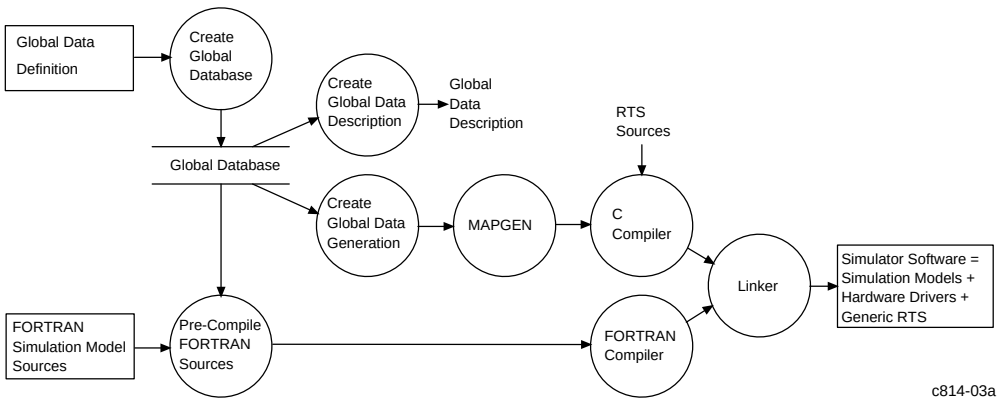


Fig. 3 Components of the Build Simulator Software.

Build Simulator Software: The Build Simulator Software component contains a set of programs that support the management of interfaces of the simulation software of PROSIM. The Build Simulator Software component supports the creation of a data dictionary to control interface management of global constant and variable definitions. This data dictionary, the Global Database, contains the definition of every variable with global significance available in the GSD memory. For these global constants and variables special attributes can be specified to describe the type, allocation, default, range and usage description. Also, the Build Simulator Software component provides tools for the generation of correct declarations on access to the GSD memory in source files. Finally, the Build Simulator Software component provides tools for retrieving, tracing and listing the use of global constants and variables.

The central part of the Build Simulator Software component is identified as the Global Database. This Global Database results from the large number of items to be exchanged between the different simulation models and hardware drivers via the GSD memory. This Global Database gives an overview and description of all global constants and variables, together with the related attributes. With respect to the independency of the definition of constants and variables the structure of the Global Database distinguishes two levels. The first level is defined by the use of Global Blocks. The second level is defined by the use of Groups of Global Blocks, where definitions of variables and constants within one Group are completely independent of the definitions in an other group. This allows the development engineer to use equal variable names in different groups. A Group contains a set of Global Blocks related to the same simulation model or hardware driver. During the development of NSF it was identified that at

least 6 levels are necessary to organize the simulation software that currently encompasses 450 Global Blocks. The 6 levels are realized by chosing a specific naming convention for the Groups and Global Blocks.

The communication between the different simulation models and hardware drivers is implemented by direct access in the GSD memory. This way of implementation gives NSF the opportunity to get access to variables even when the simulator is running real-time. Moreover it provides the development engineer and the system, environment and experiment controller a debugging facility in all states of the simulator. Only the development engineer is also privileged to change global data. This last facility has proven to be extremely important to expedite testing the simulator.

Access to the GSD memory, defined by the Global Blocks, is implemented as access to “COMMON” blocks. These “COMMON” blocks are compatible with writing software in FORTRAN and are linked to the globally accessible shared memory segments. PROSIM uses an active way of inserting declarative statements. This active declaration has the following advantages:

- Once a Global Block has been declared available for one module, it can be verified if it has been referenced inside the module.
- Once a Global Block has been declared available for a module, the declarative statements, required and necessary, can be generated precisely.
- When a global variable of a Global Block is used and the Global Block is not declared, it is possible to generate a warning or error message.

The development of a GSD memory and a set of simulation models and hardware drivers can be divided into four stages. The first stage of development is concerned with the declarations of the Global Groups, Blocks, Variables and Constants available in the GSD memory. This definition is given in one or more coherent Global Data Definition files. This stage is completed with the creation of the Global Database. The second stage of development is concerned with the creation of source code for the simulation models and hardware drivers. This stage is supported by tools for the creation of cross-reference maps, source tracing on globally defined variable names, creation of alphabetically ordered lists, etc. The third stage of development is concerned with pre-processing and compilation of the simulation model and hardware driver source files. This stage takes care of checking the consistency between the definition of the globally defined variables and constants in the Global Data Definition file and the use thereof in the simulation model and hardware driver source files. The result of this stage is object files and compilation listings of the simulation models and hardware drivers. The fourth stage of development is concerned with the generation of two additional files. These files are needed for compilation, linking and running of the complete simulator. First, the Global Data Generation

file must be created, which is used for the generation of the Global Data Access function. This function must be compiled and linked together with all other sources and objects to form the complete simulator software. Second, the Global Data Description file must be created, which is used during simulation by the RTS for all access to the GSD memory. Access to the GSD memory is performed by the RTS for actions such as data recording, event monitoring, etc.

In summary, the Build Simulator Software components contain programs for processing of the Global Data Definition files and the source files of simulation models and hardware drivers. Support of these actions has led to the definition of a number of components, of which the main components and their interactions are shown in figure 3.

5 Real-time simulation in NSF

The Real-Time simulator Software (RTS) is defined as a generic software environment for the execution and control of real-time and non real-time simulation runs. A generic software environment implies that it is not necessary to modify the RTS for different types of simulated vehicles (e.g. fixed wing aircraft, helicopter or spacecraft) or connected hardware systems (e.g. Motion, Visual, Sound). The RTS is designed in a modular structure, with minimal dependency between simulation models and hardware drivers, in order to guarantee re-configurability of the simulation facility, the possibility of research on simulation systems and the use of different simulation models and hardware systems for the performance of a simulation run.

The RTS can be used in the Real-Time Experiment (RTE) or Real-Time Development (RTD) mode. As the names of these modes indicate, the RTE mode is used during real-time experiments, possibly with a human-in-the-loop. The RTD mode is used for development and testing of simulator software and data files by the Simulation Development Engineer. One of the main differences between the RTD and RTE mode is that in the RTD mode the RTS can be used in a multi-user version sharing the computer resources with other users. In the RTE mode the RTS is used as a single-user version having all resources available. Only in the RTE mode the stringent real-time performance requirements must be satisfied.
During the software development in NSF four different hardware configurations of increasing complexity are employed. The combination of modularity of the simulation software and the modes in which the RTS can be used has achieved that, apart from a few data files, for all hardware configurations the same simulator software configuration can be used thereby extremely expediting the development process. The four hardware configurations are:

a. The development engineer using the RTS hosted on the development or real-time Challenge computer. In this configuration the RTS can be used in RTD or RTE mode. The graphical user interface is by means of an X-terminal.
b. As a. with additional communication over Ethernet with one or more graphical computer systems, displaying the MFD's, HUD, Out of The Window view, and if applicable project specific equipment.
c. The F-16 mock-up facility used by the Development engineer and pilot that provides the facility to test the several avionic systems and flight control functions in an integrated and real-time environment.
d. The NSF configuration as described in section 3 and shown in figure 2.

Figure 4 shows the configurations b. and c. with the systems, users and interfaces that are utilized.

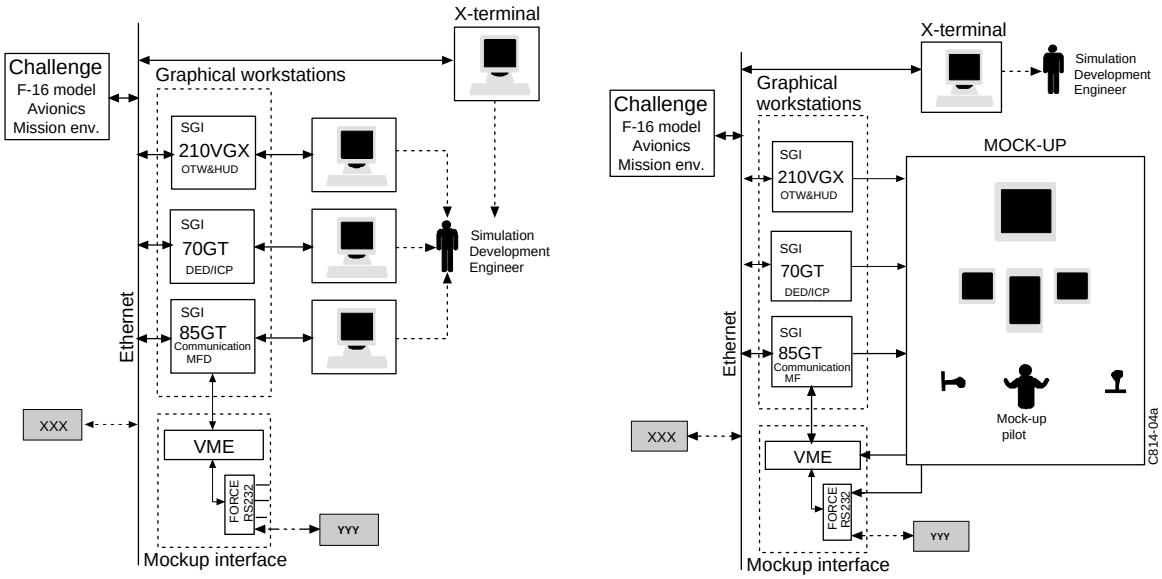


Fig.4 Two configurations used during the development of simular software.

Major features of the RTS are:

- All user interactions with the RTS, e.g. command for state transition, occurrence of events, data recording, and monitoring, etc., are performed through a Graphical User Interface.
- All simulation models and hardware drivers communicate with each other through direct access of the GSD memory, which is defined as a shared memory segment containing global variables. The GSD memory can also be accessed by the generic RTS components for interaction with the ongoing simulation run.
- The RTS is operated using a predefined number of states and state transitions (figure 5).
- Simulation tasks are scheduled in the different states and state transitions of the RTS according to the scheduling requirements as given in the schedule file. Scheduling requirements define the frequency, sequence or dependencies, processor selection, offset and duration of the simulation tasks.
- Global variables available in the GSD memory can be recorded periodically with a specified frequency. Recording of global variables is possible in the simulator states Initial Condition, Real-time Operate and Frozen.
- For development and evaluation purposes, it is possible to stimulate the simulation models and hardware drivers with pre-defined data. Possible stimuli are step, ramp, doublet, sin, cosine and time-series.
- Events can be based on a pre-defined condition in the GSD memory or on a user command.
- Error messages, commands, marks, timing info and comments are logged in a journal file.

- Configuration and initialisation of the RTS and GSD memory is performed by data files, which are created by the development engineer using the SDS and are selected using a dedicated selection mechanism.

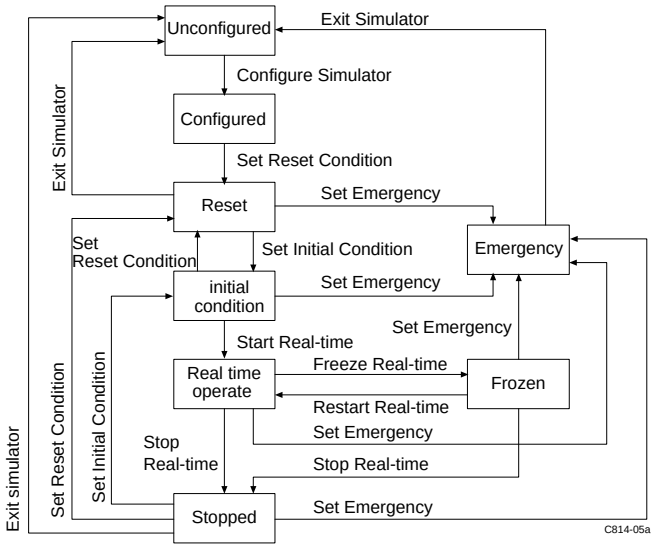


Fig.5 State transition diagram of RTS

The primary input for the Real-Time simulator Software is defined by the set of simulator software and data files. All user interactions of the simulation development engineer, system controller, experiment controller and environment controller are performed through Graphical User Interfaces.

The output of the RTS is separated in two types of files. The first type is a log file containing messages that are logged during the performance of a simulation run. All logged messages are accompanied by the simulation and up-time of the simulator. Possible messages are:

- Errors, warnings and messages that occurred in the simulation software or generic RTS components.
- Print statements specified in Mission Definition Language (MDL) scripts.
- Comments or markers entered by the user.

The second type is the run data files containing global variables that are periodically recorded. These run data files are used by the data processing software.

The implementation of the Real-Time simulator Software is based on a client/server architecture, together with the strict separation of the execution of real-time or synchronous activities and non real-time or a-synchronous activities. The MissionTool (Client) communicates with the Eventmanager (Server) through the exchange of events. As can be seen in figure 1, it is

possible to connect multiple MissionTool users to one simulator. This is achieved by connecting multiple MissionTools to a simulator that is represented by one Eventmanager.

The main components of the RTS that are relevant for the implementation of NSF are described in the remainder of this section.

Scheduler
The scheduler is defined for time-management, or scheduling, of the execution of a collection of simulation tasks in real-time according to certain scheduling requirements as given in the schedule file.

Due to several elements characterizing the simulations, some characteristics of the scheduler are extremely important. One of these elements is that the experiments are man-in-the-loop simulation. Therefor some simulation models have to be scheduled in a strict order and with a specific frequency. Such simulation models are for example the avionics simulation models and the simulation models computing the airframe motion. Also the possibility to exchange simulation models for the hardware counterpart (having a fixed scheduling frequency) requires the necessity to specify the frequency a model has to be scheduled with.
Another element characterizing the simulation is that most of the experiments are focused on handling qualities and pilot workload. Time delays between inputs due to pilot actions and outputs (cueing) to the pilot have to be deterministic and within certain limits. The scheduler must provide such scheduling facilities that it can be guaranteed that each time frame the simulation models are scheduled at a certain frequency and in the same order. In this light it is also necessary to define a certain priority to simulation models so that ‘time-critical’ simulation models can run real-time at the cost of less time-critical simulation models scheduled with a ‘preference’ frequency. For this less time-critical simulation models an ultimate lower-limit-frequency will have to be specified to prevent simulation models not to be scheduled at all at high processor load.

PROSIM defines a simulation task as an instance of the generic view on anything that needs to participate in the simulator’s real-time activities. The simulation tasks are executed by the scheduler as one entity. The scheduler does not distinguish between simulation tasks implementing simulation models, hardware drivers or other components of the RTS. Simulation tasks can be grouped into task groups, which are defined as sets of logically coherent simulation tasks. A task group always exports one “initialize” and one “terminate” operation, which respectively establish or eliminate (invalidate) the environment that is required by each individual simulation task within the task group.

```
BASIC_FREQUENCY          50
RT_PROCESSORS            2
          .
          .
          .
STATE INITIAL_CONDITION
 STRONGLY_PERIODIC       VISCOM            PERIOD 1   PROCESSOR 1
 STRONGLY_PERIODIC       IDBCOM            PERIOD 1   PROCESSOR 1
 STRONGLY_PERIODIC       EventManager      PERIOD 1   PROCESSOR 1
 STRONGLY_PERIODIC       iccio             PERIOD 1              AFTER IDBCOM
 CYCLIC                  MOTCOM_NEUTRAL                PROCESSOR 1
 CYCLIC                  SetData
 CYCLIC                  IDBCOM_CONTROL PROCESSOR 1              AFTER SetData
 CYCLIC                  confgsm                                 AFTER SetData
 CYCLIC                  iccomp                        PROCESSOR 1   AFTER confgsm
 CYCLIC                  MOTCOM_CONTROL                PROCESSOR 1   AFTER MOTCOM_NEUTRAL, iccomp
 CYCLIC                  sysinit                       PROCESSOR 1   AFTER iccomp
 CYCLIC                  avionxic                      PROCESSOR 1   AFTER
                                                       IDBCOM_CONTROL,MOTCOM_CONTROL
 CYCLIC                  ITSCOM_CONTROL
 CYCLIC                  menvic                                  AFTER ITSCOM_CONTROL, avionxic
 PERIODIC                ICVISUAL          PERIOD 1              AFTER VISCOM, cdtouch, avionxop
 PERIODIC                ITSCOM            PERIOD 2
 PERIODIC                IDBFLASH          PERIOD 1   PROCESSOR 1
 PERIODIC                cnavmod           PERIOD 1              AFTER iccio
 PERIODIC                celecsys          PERIOD 1              AFTER iccio
 PERIODIC                cflinstr          PERIOD 1              AFTER iccio, avionxop
 PERIODIC                avionxop          PERIOD 1              AFTER iccio
 PERIODIC                ufcop             PERIOD 1              AFTER avionxop
 PERIODIC                eth2gfx           PERIOD 1              AFTER ufcop
 PERIODIC                cdtouch           PERIOD 1              AFTER IDBCOM
          .
          .
          .
 PERIODIC                cded              PERIOD 1   AFTER ufcop
 PERIODIC                mfd_buttons       PERIOD 1   AFTER IDBCOM
          .
          .
          .
```

Fig.6 Example of NSF schedule file during Initial Condition state.

Simulation tasks can be divided into four classes, depending on their use during the execution of a simulation run. The first class includes the periodic simulation tasks. Periodic tasks are time critical tasks, which must be activated on average with the specified frequency. The second class includes the strongly periodic simulation tasks. Strongly Periodic tasks are activated at exactly the frequency specified and are used for most of the NSF hardware driver tasks. The third class includes the background simulation tasks. Background tasks are less time critical tasks, and are activated as often as possible in the time left by the (strongly) periodic simulation tasks taking a minimum execution frequency into account. The fourth class includes the cyclic simulation tasks. Cyclic tasks are used for iterative initialisation of the simulation. The cyclic tasks are the only tasks that are not allowed to be scheduled during the Real-time Operate state.

The classes of simulation tasks have default scheduling requirements, which can be changed or extended. Possible scheduling requirements for the execution of simulation tasks are:

- Frequency, the (minimum) frequency at which the simulation task must be activated. At NSF most simulation software is scheduled at 50 Hz.
- Execution processor, the processor number of the processor on which the simulation task must be executed. If the processor number is absent, it will be chosen by the scheduler.
- Preemptability, preemptable simulation tasks are interrupted in case of more urgent tasks.
- Allowed execution time, the allowed execution time specifies the amount of time required for the execution of periodic tasks.

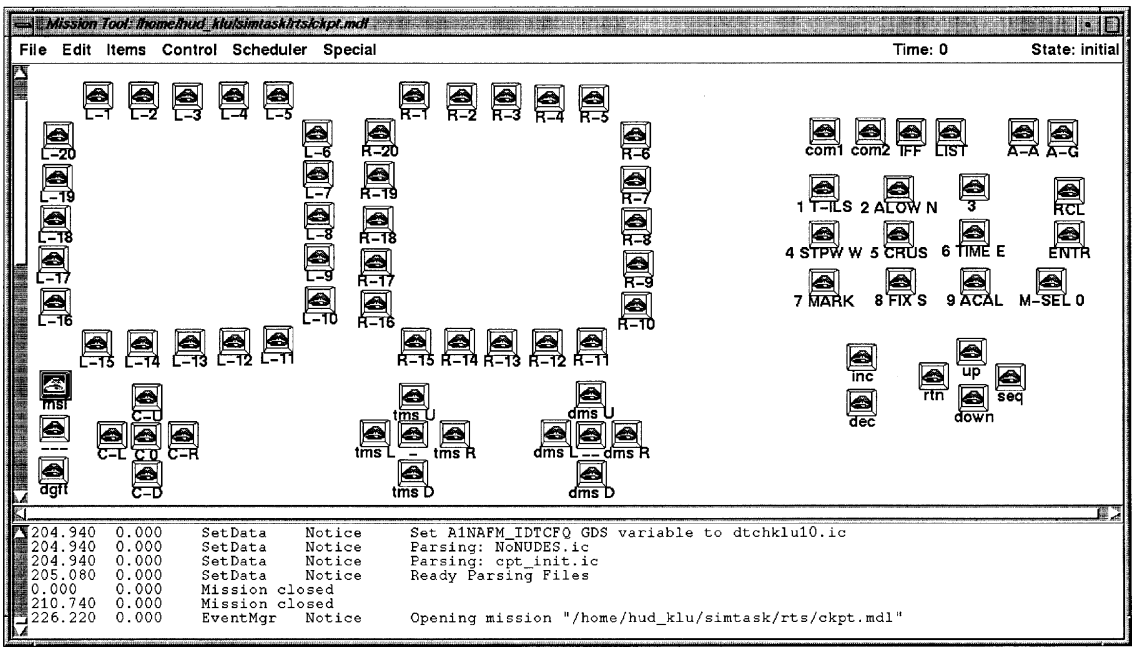


Fig. 7 Example of the Mission Tool top-level window

- Dependencies, dependencies are used for synchronisation purposes between the scheduled simulation tasks. The start of the execution of a specific task can be forced after the completion of other tasks. In particular this option is used for the simulation tasks performing I/O with other computer systems.
- Offset, the first execution of a periodic simulation task can be postponed by specifying the offset.

As specified before, the simulation can be divided in several states. For each state, a separate collection of simulation tasks with scheduling requirements can be defined. This implies that the scheduler does not need to make any distinctions between the different states. An example of the schedule file at the Initial Condition state is given in figure 6.

Error Handler

The error handler is defined as the central component where all errors occurring in the synchronous and a-synchronous parts of the RTS are collected and presented to the user. All reported errors are provided with a time stamp, specifying the simulation time and uptime of the moment of occurrence of the error.

MissionTool

The MissionTool serves as the Graphical User Interface (GUI) for the execution of (non) real-time simulations. This GUI contains functions to monitor, modify, record, stimulate the value of GSD variables, view and edit actions of the mission definition, and general control over the execution of the simulation run.

The MissionTool is configured using the mission definition specified in the Mission Definition Language (MDL). This MDL is a meta-language for simulation scripting, in which actions are described in limited free-text or 'C' type syntax. Each action contains its name, attributes and statements to be executed, and the condition on which the action must be executed. The facilities available for the definition of the actions and conditions in MDL include: flow control; read and write access to GSD variables; and Monitor, Registration and Stimulation commands. During the development and operation of NSF the configuration of the MissionTool by means of the MDL file is experienced as an efficient and effective tool. An example of an MDL file is shown in figure 7 where the bezels of the F-16 MLU MFD's and the ICP buttons are emulated when testing and debugging the F-16 MLU avionic software.

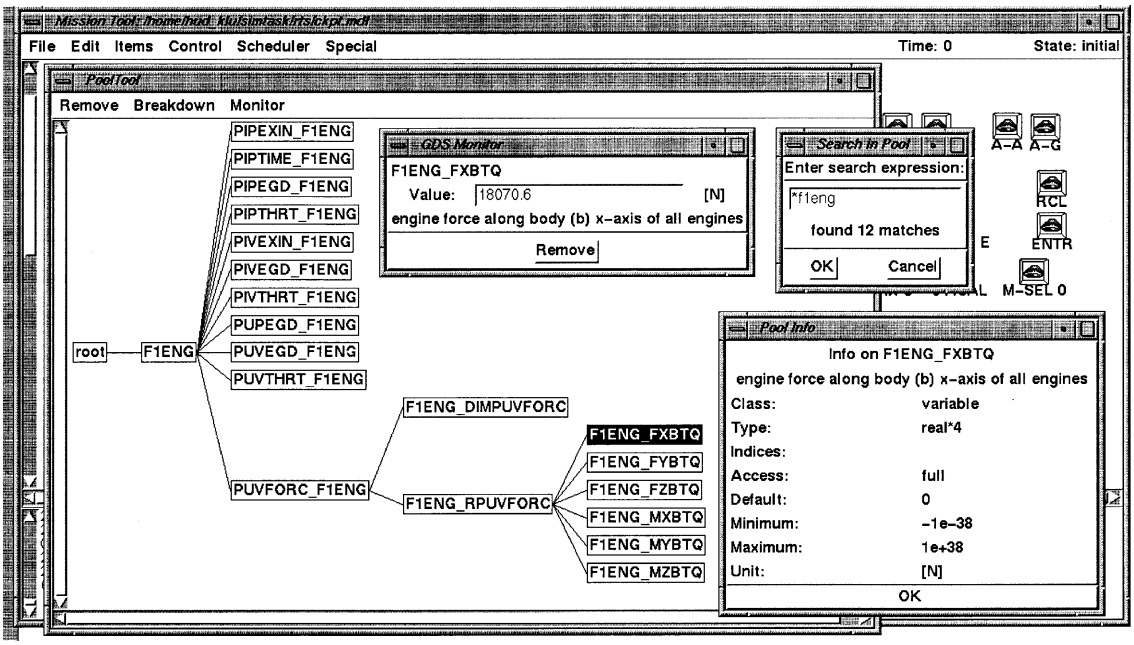


Fig. 8 Example of the PoolTool window.

The top level window of the MissionTool consists of three graphical windows. An example of a top-level window is given in figure 7. The header of this top-level window contains a line with a menubar, the simulation time and simulator state. The other two graphical windows contain the action sheet and event logger windows.

The action sheet window contains the representation of the set of actions defined in the mission definition, which can be created, activated, deactivated, executed, or edited. The lay-out of the action sheet window can be organized by positioning the available actions. The event logger window provides a log of all the activities occurred during the simulation run execution. All activities are logged simultaneously to a log file. From the menubar of the MissionTool the PoolTool window can be opened by selecting the appropriate menu option. With the PoolTool, GSD variables can be selected to be set, monitored or information obtained. This

information includes the name, a description, the class, the type, unit, access rights, and the default, minimum and maximum values of the GSD variable. The PoolTool also provides a presentation of the hierarchy of the GSD memory using a tree structure. This tree can be expanded or collapsed to view the required level of detail. The value of a variable in the GSD memory can be monitored (by all users) and/or set (only by the development engineer) using Monitor windows. Monitors can be started by an MDL command or by selection of the New Monitor option from the PoolTool menu. An example of the PoolTool window with a Monitor window is given in figure 8.

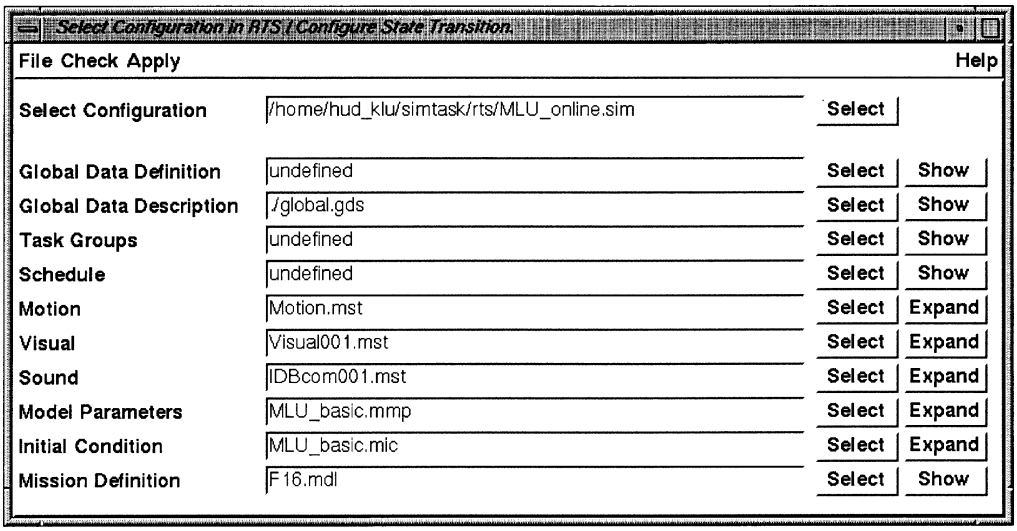


Fig. 9 Example of the Select Configuration window.

Configure Simulator

Configure Simulator contains the Select Configuration and the Set Global Simulator Data components, which perform the following actions at the state transitions to Configured, Reset and Initial Condition:

- Selection of data files from libraries created in or with the SDS.
- Configuration of the RTS for the execution of one or more simulation runs using the selected configuration files.
- Initialization of the GSD memory using the selected data files.

An example of a Select Configuration window is given in figure 9.

6 Conclusions

It has been recognized in an early stage of the development of the National Simulation Facility NSF that an effective simulation tool like PROSIM is indispensable to obtain a high fidelity simulator where NSF is aiming at.
PROSIM is a generic simulation tool developed at the Netherlands National Aerospace Laboratory NLR and can be used for design, verification, test and training of a wide range of applications. It has been tested and validated, and its use and correctness has been proven already during the development and operation of NSF, where currently PROSIM is used as simulation tool for F-16 MLU full-mission flight simulation.

PROSIM is operational, and in use at the Flight Division since June 1994. Evaluation of PROSIM shows that users experience PROSIM as a user friendly and versatile simulation tool.

7 References

1. A.A. ten Dam, P. Schrap and W. Brouwer, PROSIM: the Simulation Program of the National Simulation Facility NSF, presented at the Third Workshop on Simulators for European Space Programmes, ESTEC, Noordwijk, The Netherlands, 15-17 november 1994

2. A.P.L.A. Marsman, Flexible and High Quality Software on a Multi Processor Computer System Controlling a Research Flight Simulator, in AGARD Conference Proceedings no. 408, september 1986.

3. W. Loeve, Engineering of systems for application of scientific computing in industry, NLR Technical Publication TP 92160 L, 1992

4. F.J. Sonnenschein, M. Schoonmade and T. Zwartbol, Spacecraft Attitude and Orbit Control Systems Testing, presented at the AGARD Flight Vehicle Integration Panel Symposium on Space Systems Design and Development Testing, Cannes, France, 3-6 October 1994.

5. M.J.H. Couwenberg, A.A. ten Dam and M.T.G.A.R. Mans, The approach for the development of simulators at the National Aerospace Laboratory (NLR): two space related examples, Proceedings “2nd Workshop on Simulators for European Space Programmes”,1992

6. P. Schrap, Programme and Real-Time Operations Simulation Support Tool PROSIM: Software User’s Manual, to appear as NLR Technical Report, 1995

7. H.A.J.M. Offerman , Development of the Dutch National Simulation Facility NSF, Proceedings of AIAA 1994, Flight Simulation Technologies Conference, Scottsdale, Arizona, USA, 1994

Acknowledgements

PROSIM was designed and realised by a team from NLR’s Flight- and Informatics Division, of which the present authors are members. Part of the RTS has been made by BSO and FSS under supervision of NLR.

This page is intentionally left blank

Appendix

A Overview of simulation models

F-16 Airframe
. Aerodynamics
. Engine
. Undercarriage
. Mass and geometry

F-16 aircraft systems
. Flight instruments signals
. Flight control system
. Speedbrakes/drag chute deflection
. Voice message unit status
. Fuel system status
. Gear system status
. General system status

F-16 simulator systems
. Cockpit inputs/outputs
. ARWR
. Data Entry Display (DED)
. MFD bezels

F-16 Avionics Systems
. Advanced Identification Friend or Foe (AIFF)
. Airborne Self Protection Jammer (ASPJ)
. Central Air Data Computer (CADC)
. Combined Altitude Radar Altimeter (CARA)
. Data Transfer Equipment(DTE)
. Fire Control Computer(FCC)
. Fire Control Radar(FCR)
. Global Positioning System (GPS)
. Head-Up Display (HUD)
. Improved Data Modem (IDM)
. Instrument Landing System (ILS)
. Inertial Navigation System (INS)
. Multi Function Display Set (MFDS)
. Microwave Landing System (MLS)
. Radar Warning Receiver (RWR)
. Stores Management Set (SMS)
. Tactical Air Navigation (TACAN)
. Digital Terrain System (DTS)
. Up-Front Controls (UFC)
. Counter Measures Dispensing Set (CMDS)

F-16 Weapons
. Missile Fly-out characteristics
. Air-to-ground ballistics
. Maverick Pre-Launch Condition
. M61 gun

Aircraft Mission environment
. Threats
. Air targets
. Ground targets
. Aircraft environment
- Navigation signals
- Aircraft disturbances
- Atmosphere

General simulation models
. Flight mechanics
. Initial condition
. Stability derivatives

NSF Hardware systems
. Motion system
. Visual system
. ITEMS
. FSIS
. Control Desk
. Sound system
. G-cueing systems
. Mock-up
. Instrument graphics Computers