

AD-A048 164

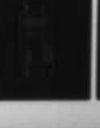
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2  
A DISCRETE-EVENT DIGITAL SIMULATION MODEL OF THE F-16 FIRE CONT--ETC(U)  
DEC 77 L R HANSON

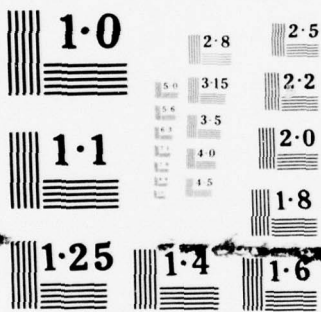
UNCLASSIFIED

AFIT/GE/MA/77-1

NL

1 of 2  
ADA  
048164





NATIONAL BUREAU OF STANDARDS  
 MICROCOPY RESOLUTION TEST CHART

1

⑨ Master's Thesis

⑧  
A DISCRETE-EVENT DIGITAL SIMULATION  
MODEL OF THE F-16 FIRE CONTROL  
COMPUTER OPERATIONAL FLIGHT  
PROGRAM USING SIMSCRIPT II.5

⑭  
AFIT/GE/MA/77-1

THESIS ⑩  
Larry R. Hanson  
Captain USAF

⑪  
Rec 47

⑫  
107p.

Approved for public release; distribution unlimited

DISTRIBUTION STATEMENT A  
Approved for public release;  
Distribution Unlimited

DL2225

DDC  
RECEIVED  
JAN 9 1978  
Bart  
yB

AFIT/GE/MA/77-1

A DISCRETE-EVENT DIGITAL SIMULATION  
MODEL OF THE F-16 FIRE CONTROL  
COMPUTER OPERATIONAL FLIGHT  
PROGRAM USING SIMSCRIPT II.5

THESIS

Presented to the Faculty of the School of Engineering  
of the Air Force Institute of Technology  
Air University  
in Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science

by

Larry R. Hanson, B.S.

Captain USAF

Graduate Electrical Engineering

December 1977

Approved for public release; distribution unlimited

PREFACE

This report results from an effort to provide AFAL/AAF-3, Wright-Patterson AFB, Ohio with a simulation model of the F-16 Fire Control Computer Operational Flight Program (OFFP). The purpose of the model is to enhance the system analysis capabilities being used by the Advanced Systems Group at AFAL to evaluate the OFFP. I hope that this additional tool will prove useful to AFAL and to any others who are associated with the system.

I wish to express my appreciation to the members of the Advanced Systems Group for their assistance and guidance during this project. They provided personal system knowledge through numerous interviews and made available the most current documentation of the system.

My thanks are extended to Major Kenneth B. Melendez of the AFIT Mathematics Department for his suggestions and leadership as my faculty thesis advisor, and to Dr. Gary Lamont and Dr. Charles Richards for their service as members of the thesis committee. My special thanks to Mrs. Beverly Stevens of Dayton, Ohio for typing the final report.

Finally, I wish to thank my wife and children for their love and patience throughout this endeavor.

Larry R. Hanson

ACCESSION for	
NTIS	White Section <input checked="" type="checkbox"/>
DDC	Buff Section <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	<input type="checkbox"/>
BY _____	
DISTRIBUTION/AVAILABILITY CODES	
Dist.	AVAIL. and/or SPECIAL
A	

## TABLE OF CONTENTS

	<u>Page</u>
Preface . . . . .	ii
List of Figures . . . . .	v
List of Tables . . . . .	vi
Abstract . . . . .	vii
I. Introduction . . . . .	1
Background . . . . .	1
Problem Statement, Approach, and Goals . . . . .	3
Scope . . . . .	4
II. The Art of Modeling . . . . .	5
Model Classifications . . . . .	6
Event Scheduling Approach . . . . .	8
III. Essentials of SIMSCRIPT II.5 . . . . .	11
IV. F-16 Fire Control Computer Operational Flight Program . . . . .	15
Associated Terminology . . . . .	15
Characteristic Differences of Avionic Software . . . . .	16
AFAL Software Test Stand . . . . .	17
Fire Control Computer (FCC) Function Descriptions . . . . .	19
Fire Control Computer OFP Mode Descriptions . . . . .	20
The OFP Structure . . . . .	23
Data Movement and Bus Control . . . . .	27
Summary . . . . .	33
V. Design of the Simulator . . . . .	35
Temporary Entities . . . . .	36
Events . . . . .	39
Subroutines . . . . .	42
Special Features . . . . .	42
VI. Results and Conclusions . . . . .	44
Verification Results . . . . .	44
Conclusions . . . . .	46
Suggested Improvements . . . . .	47

TABLE OF CONTENTS (Cont'd)

	<u>Page</u>
Bibliography . . . . .	49
Appendix A: Computer Source Listing . . . . .	51
Appendix B: OFP Segment Definitions . . . . .	73
Appendix C: List of Abbreviations . . . . .	95
Vita . . . . .	97

LIST OF TABLES

<u>Table</u>		<u>Page</u>
I	OFP Computer Program Component Description . . . . .	24
II	SAFE Transmissions . . . . .	31
III	Data Block Descriptions . . . . .	32
IV	Simulation Results-Data Comparison . . . . .	45
V	Simulation Results-I/O Task Comparison . . . . .	45
VI	Executive Segment Definitions . . . . .	74
VII	System Control Segment Definitions . . . . .	75
VIII	Bus Control Segment Definitions . . . . .	80
IX	Initialization/Error Handling Segment Definitions . . . . .	79
X	Navigation Support Segment Definitions . . . . .	80
XI	Fixtaking Segment Definitions . . . . .	81
XII	Cruise Energy Management Segment Definitions . . . . .	84
XIII	Combat Energy Management Segment Definitions . . . . .	85
XIV	Air-to-Air Gunnery Segment Definitions . . . . .	86
XV	Air-to-Air Missile Segment Definitions . . . . .	87
XVI	Air-to-Ground Segment Definitions . . . . .	88
XVII	Stores Data Select Segment Definitions . . . . .	92
XVIII	Data Entry/Display Segment Definition . . . . .	93
XIX	Self Test Segment Definition . . . . .	94



LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1	AFAL F-16 Software Test Stand-Simplified . . . . .	2
2	Model Classification . . . . .	6
3	Task Scheduling-Arrival Event . . . . .	9
4	Task Scheduling-Departure Event . . . . .	9
5	Avionic Software Terminology . . . . .	16
6	AFAL F-16 Software Test Stand-Detailed . . . . .	18
7	Rate Group Execution Sequence . . . . .	28
8	I/O and Computation Sequence . . . . .	29
9	Data Word Transmission Formats . . . . .	31
10	OFP Task Interaction Flowchart . . . . .	34
11	Task "Arrival" and "Departure" Event Combination . . . . .	37
12	Event CLOCK Flowchart . . . . .	40

ABSTRACT

In support of a request from the Air Force Avionics Laboratory, a model of the F-16 Fire Control Computer (FCC) Operational Flight Program (OFP) was developed. The initial specification required that this model allow for possible changes to the rate of accomplishment of various OFP mode-dependent tasks. The reconfiguration of the input and output tasks and the processing reserve were of particular interest.

In order to determine the most useful approach, the various computer system modeling relationships are first reviewed. Based on the author's background, the real world system and the modeling goals, a discrete event queue level simulation using SIMSCRIPT II.5 is selected as the desired approach.

The basic features of the F-16 FCC and the OFP are discussed and a description of the task movement in the system is provided. This description is used to detail the various rate groups and their member tasks. The model is verified by comparison against data derived from an actual system run and a statistical analysis provided by the OFP manufacturer.

The verification process showed that all the original design objectives were met, although several areas of possible improvement to the model are indicated and discussed.

A DISCRETE-EVENT DIGITAL SIMULATION  
MODEL OF THE F-16 FIRE CONTROL  
COMPUTER OPERATIONAL FLIGHT  
PROGRAM USING SIMSCRIPT II.5

I. INTRODUCTION

The science of "systems analysis" has been developed to aid engineers and managers to understand and evaluate the consequences of changes to large, complex systems. Many efforts exist in the literature where techniques of systems analysis have been applied to computer systems. These techniques or tools are applied to hardware, software, and to entire computer systems. These efforts compose a significant portion of what is known as computer performance evaluation (CPE). This paper discusses the technique and application of one tool, simulation, to investigate the task timing relationships in the F-16 fire control computer (FCC) operational flight program (OFP). The FCC is the controlling device of the F-16 Fire Control System (FCS).

Background

The Air Force Avionics Laboratory (AFAL) has been directed to conduct an independent assessment of the F-16 fire control computer OFP. The testing and evaluation of the OFP requires the use of a software test stand.

The test stand simulates the various flight conditions of the F-16 and all aircraft subsystems, except the FCS, that are necessary to verify the effectiveness of the OFP. An actual FCC executing the OFP is attached to the aircraft simulation to complete the software test stand as shown in Figure 1. A more detailed description of the test stand is given in Chapter IV.

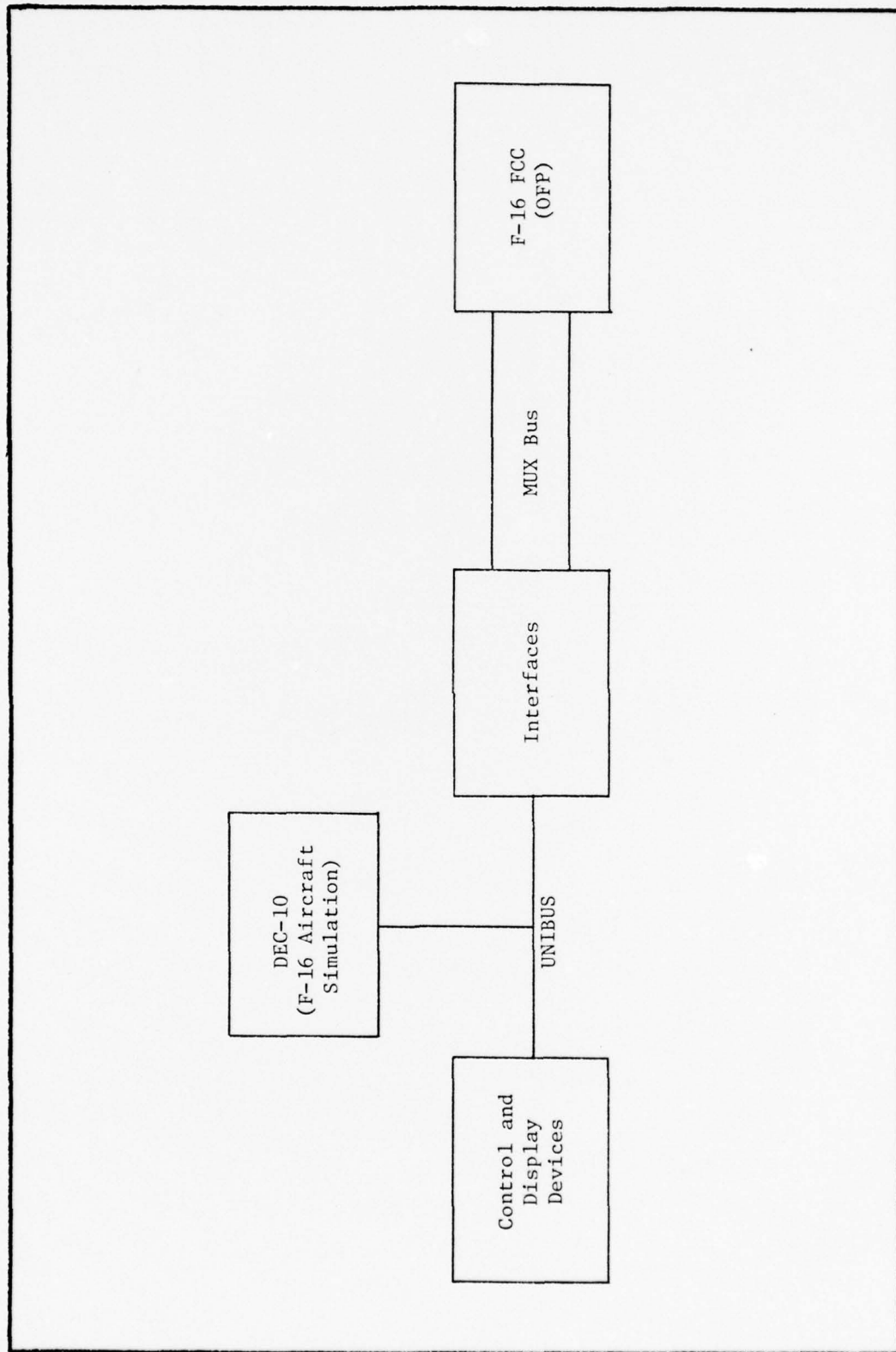


Figure 1. AFAL F-16 Software Test Stand - Simplified

It is presently envisioned that another test stand will be built at Hill AFB, Utah to support Air Force Logistics Command (AFLC) functions.

During the early evaluation of the OFP, AFAL determined that certain system and flight conditions caused the OFP to enter a system failure state called a "timeout." In an effort to anticipate when a timeout would occur, an extensive CPE effort is underway at AFAL and at General Dynamics, the OFP manufacturer.

#### Problem Statement, Approach, and Goals

The AFAL Advanced Systems Group (AFAL/AAF-3) requested that a model of the test stand be developed. The model would become a tool to be used by CPE personnel to study possible improvements to the OFP and to the test stand configuration. The processing reserve and the input/output bus transfer reserve in the OFP were of particular interest.

The approach to solve the problem was to accomplish the following steps sequentially:

1. Define the goals of the model.
2. Learn a simulation language.
3. Identify the task parameters of the OFP and of the test stand devices.
4. Develop the model.
5. Obtain the data to verify the operation of the model.
6. Operate the model under various changes in parameters.

The following goals have been established through consultation with AFAL/AAF-3 to define the model characteristics needed to study the OFP; it must allow for changes in the OFP task scheduling sequence; it must allow for program execution to be held up while awaiting termination of an executing OFP task in the FCC main processor and the FCC bus controller;

it must be sufficiently documented for ease in use and modification; it must provide sufficiently accurate statistics to form a basis for making management decisions; it must reflect FCC mode of operation; and it must allow for the FCC background task interrupt and task restart capability. AFAL/AAF-3 was also interested in the evaluation of different Software Test Stand configurations. This was not feasible with the model level needed to achieve the goals established for OFP investigation. The development of an additional model was not possible in the time constraints of this work.

#### Scope

The depth of the investigation was determined from the established goals to require a queue level model of the OFP. A simulation model was chosen to implement the model because of the need to (1) maintain accurate operating conditions, (2) explore task assignment alternatives, and (3) control the passage of time. All data input to the simulation model was deterministic data for simplification.

This report discusses a brief summary of various methods in model construction (Chapter II) and includes summary information necessary for a basic understanding of SIMSCRIPT II.5 (Chapter III). A detailed description of the OFP as it is currently documented is contained in Chapter IV. The actual design of the simulation model and how it differs from the OFP is discussed in Chapter V. The model verification results and conclusions of this work and suggested improvements are contained in Chapter VI. A listing of the simulator model is found in Appendix A and OFP Segment Definitions are found in Appendix B. A list of abbreviations is found in Appendix C.

## II. THE ART OF MODELING

Computer systems have been studied on many different levels. Each level characterizes the system by particular relationships, parameters, and events. These particular characteristics can be, and generally are, different as the level of detail is changed. Any analysis of a complex system attempts to study the effects of a modified, "simple" system composed of only the significant variables and relationships. This reduction of the total system to a limited system is defined here to be the modeling task. Through the use of a model, the system is reduced to an entity that is an abstraction and yet still understandable. Model building constantly poses a problem for the model builder of balancing the need for system detail with the need to make the model that will provide a solution to the problem. Where system detail is excluded from the model, one must "fill in" with assumptions about that detail. These assumptions must be tested against the actual system operation whenever possible.

A model is intended to aid system analysis. However, there are some risks involved in this technique. One of the risks is that the investment of time and effort may not produce a useful benefit. This can occur when the investigator relies too much on the modeling method and fails to apply his own initiative to develop the model. Model building must be regarded as an art and not as a science.

Another common problem would develop if the investigator were to attach an excessive degree of significance to the output of the model. A model can never predict beyond the range of its assumptions, and a model is only as useful as the image it presents of the real world system.

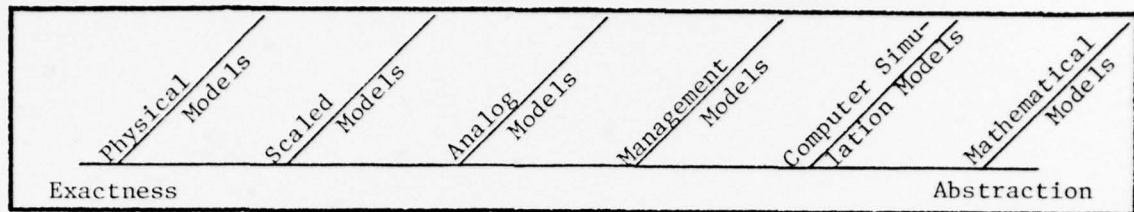


Figure 2. Model Classification (Ref 21:8)

Because of the cost and talent involved, model building may not produce a valid image of the real world system. It must be remembered that the model produced will not be a unique model and will always contain a degree of imprecision.

This chapter discusses the relationship of computer simulation models to other types of models. Of the three discrete event modeling techniques, the event scheduling approach is discussed in detail.

#### Model Classification

Models have been classified by several schemes (Ref 21:7, 23:29, 10:12). Of interest here are only simulation models. Shannon places digital computer simulation models next to mathematical models as shown in Figure 2. It should be noted that all models in Figure 2 are classes of simulation models and a computer may be used in some portion of any of these models. Computer system simulation is a methodology to solve a problem using a model of a real world computer system. This is not to be confused with computer simulation which uses models which are not necessarily images of real world computer systems.

The type of simulation model that is developed is based on its intended use, the goals of the investigation, the level of detail, and the real world system. Changes in the system state will usually be of major importance when studying the real world computer systems. These state changes



can occur continuously or at discrete moments in time depending on the defined state parameters. Thus, the model of the computer system is often built in the time domain. This allows the model to control time and, therefore, provide the tools of time expansion, compression, interruption, and restart capability. The speed of the simulating computer is independent of the model for all practical purposes.

The goals of this investigation, as well as the attributes of the system being modeled allow for changes in system state to occur at distinct moments in time. This condition is best described in a discrete event model. Here an event denotes a change in the system state. A discrete event model requires a clock to be maintained that is advanced after each change of the system state. The amount of advance corresponds to the real time required to elapse before the next state change.

There are three alternative discrete event modeling techniques described by Fishman (Ref 10). These techniques are related to various simulation programming languages. SIMSCRIPT and GASP are related to the event scheduling approach. GPSS and SIMULA are related to the process interaction approach. And CSL is related to the activity scanning approach. Fishman defines a process as "a sequence of events ordered on time," an activity as "a collection of operations that transform the state of an entity," and an event as "a change in state of an entity" (Ref 10:24).

A model type that can allow for variations of task (job) parameters as well as identify the system state (mode) is a "queuing" model. Jobs enter the system and compete against existing jobs for system resources. When the resources are not available, the job is placed in a waiting line or a queue until the resources become available. Each task must contain parameters to describe the time and resources required to complete the task. Resource management policies are often analyzed using a queuing

model (Ref 23:34). An entire system can be modeled as a network of queues and their relationships.

#### Event Scheduling Approach

Only the event scheduling approach to discrete event simulation is included here because this approach permits the determination of the system state at the OFP-event level in the most straightforward manner. As will be seen in Chapter IV, the OFP is composed of a series of task lists which are easily represented as a set (queue) of jobs to be completed. The reader is referred to Fishman for a discussion of the other approaches (Ref 10:38, 40). The use of SIMSCRIPT II.5 makes use of the event scheduling approach. This language was available on the AFIT computer facility, and GASP was not available.

A change in system state occurs each time a task enters the system and each time a task completes execution and departs the system. Associated changes in state occur when the resource or server becomes busy or is idled.

When a task enters the system, it identifies its server requirements and its service time. The server is checked for availability. If the server should be idle when the task enters, the server is placed in a busy condition. The task completion time is identified, and the task completion is scheduled. If the server should be busy when the task enters, the task is placed in a queue. This arrival event is shown in a flowchart in Figure 3.

When a task has received the required service, it departs the system. As shown in Figure 4, this task completion frees the server, which is now available for another task. Therefore, the queue is checked to determine if a task is awaiting service. If the queue is empty, the server is placed in an idle condition. If there is a waiting task in the queue, the server

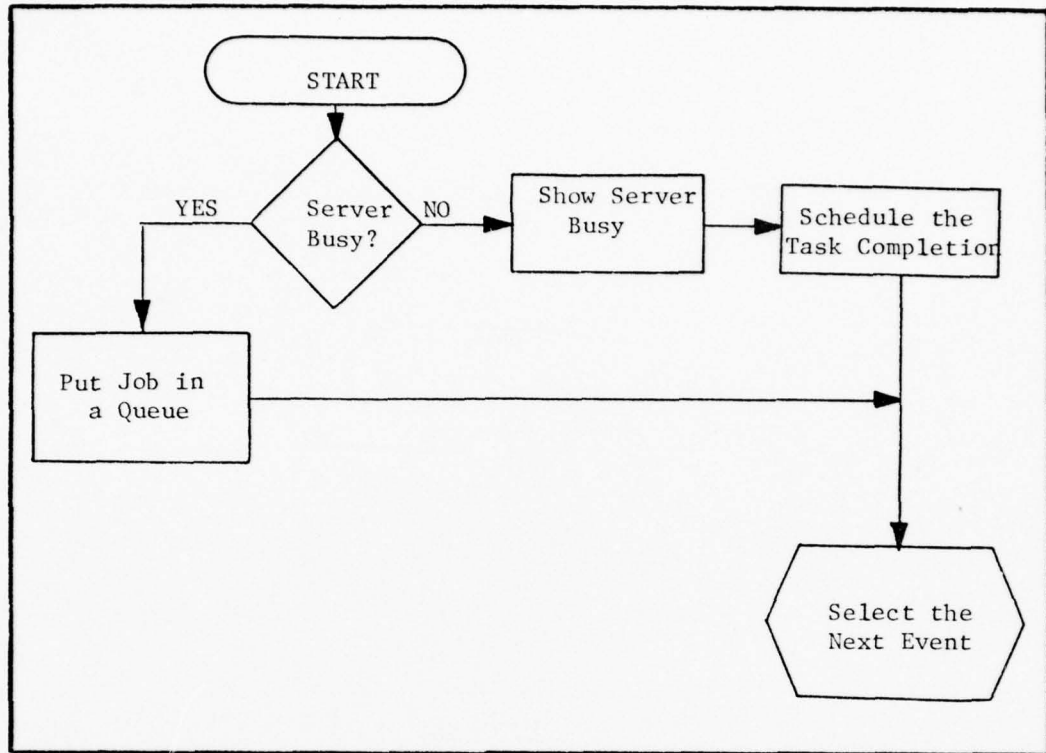


Figure 3. Task Scheduling-Arrival Event

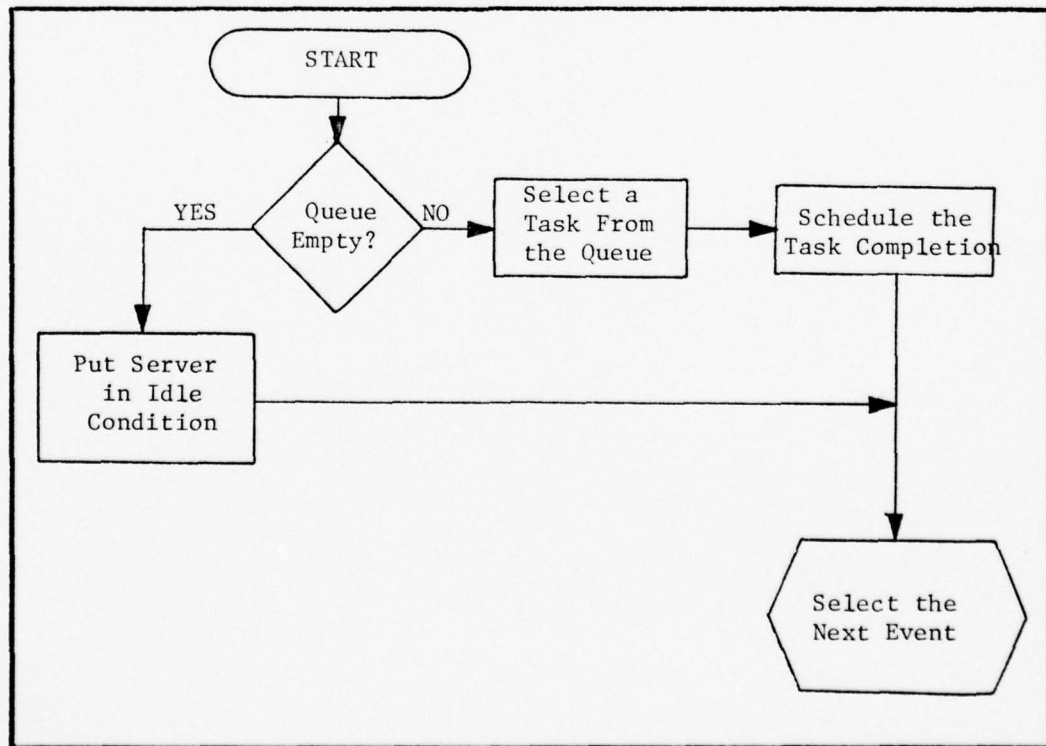


Figure 4. Task Scheduling-Departure Event

starts servicing or executing the new task, the new task completion time is identified, and the task completion is scheduled.

The last instruction in the arrival event and the departure event is "Select the Next Event." This instruction returns control to the simulation "dispatcher" that makes the discrete event simulation work. The dispatcher is an executive scheduler and is often called a control or timing routine. As an event is scheduled, an entry is made in a special list to identify the event and when it is to begin execution. The dispatcher searches that list to find the next event start time, the simulated time is advanced to this next start time and the event is "started."

A queue usually has a rule for selecting the next task to be placed in service. This rule is called a queue discipline. The discipline can select the task to be removed from the queue based on a task parameter (such as service time or priority) or on an order of task arrival (such as first-in-first-out or last-in-first-out).

Several procedural features can be added to the arrival and departure events. An event may need to schedule itself. This would allow the model to be self-generating and thus a simulation could continue without requiring external data.

It is often possible to combine the arrival and the departure events. This occurs if the queue is never in an empty condition, i.e., there is no possibility that the server will become idle. This condition would result when several queues feed a single server and all are not empty at once. The addition of a large repeated background task would also provide this option. This feature is used in the model and is discussed in Chapter V.

To aid in efficient use of computer memory, records of completed tasks are often discarded (destroyed). This feature may not be desirable, however, due to some purpose of the model.

### III. ESSENTIALS OF SIMSCRIPT II.5

This chapter provides a brief review of the SIMSCRIPT II.5 language characteristics. It is intended only as a general review and the reader is referred to the literature (Ref 22) for the language details.

A simulation language provides the basic mechanism for creating a discrete event simulator. Functional flow charts are first used to define a model of a real world system. The pictorial model is then coded by means of a simulation language to form the simulator. Simulation languages such as GASP, GPSS, and SIMSCRIPT have special built-in characteristics for dealing with discrete events and for gathering certain statistics.

SIMSCRIPT is a general purpose programming language, which is compiled directly into an assemble-language code by its compiler. It provides flexibility for statistical outputs, mathematical manipulations, and programmer expression in simulator design. The simulated system is composed of "entities," "attributes," and "sets." An entity denotes an object of the system such as a task, a variable, a channel, a memory block, a server, etc. An entity is described by its attributes. The status of the system is determined by the current values of these attributes. Entities can be grouped into sets. A set can arrange the entities on an order basis of first-in-first-out (FIFO), last-in-first-out (LIFO), or ranked on the value of some attribute. Sets are used as queues, etc.

Entities interact with specific conditional activities called "events." An event is like a subroutine, and is composed of SIMSCRIPT statements. An event always occurs at a specific time and changes the state of the system by changing the attribute values of the interacting entities. A process which has a finite duration must usually be modeled and programmed

as at least two events; one which initiates the process and one that terminates the process. Typical events would be job arrival, job schedule, request memory, request processor, end computation, begin input, begin output, end input, end output, release memory, etc.

The SIMSCRIPT compiler automatically creates an executive scheduler, a simulation control or timing routine, that advances the simulated time to the time of the next scheduled event and calls the events in proper sequence. The SCHEDULE statement allows the modeler to define when an event is to occur in the future. An event can schedule itself or another event.

The input to the simulator is a description of the real world hardware and software configuration, and a description of the workload. The workload is represented by jobs or tasks that enter the simulator and place demands for system resources or services. All initializations must be stated in program statements in SIMSCRIPT II.5.

The output of the simulator can be a snapshot of the system status at some point in the simulation run. The programmer has the capability to measure various activities and output a basic statistical summary of what occurred. The attribute values are monitored and summary variables are updated to provide a simple means to produce the statistics.

Some activities are stochastic and the language includes facilities which ease the burden in the generation of random variates that fill a desired probability distribution. There are ten random number generators and eight distributions available.

SIMSCRIPT II.5 contains a program section called the PREAMBLE. Program statements in the PREAMBLE are nonexecutable and provide the compiler with model definitions and relationships. The ownership/membership or

structure of entities, attributes, and sets are defined. The existence of events, routines, variables (name, type, and dimension), and arrays are stated. The summary variables, attributes, and types of statistics desired are described. The PREAMBLE can be omitted, in which case certain default conditions are used.

Another program block is the MAIN section. This is the main routine of the simulator. It is in MAIN (for the model developed in this work) where the system initialization occurs. Variables are set to their starting values. Sets are built. Entities are either CREATED or READ in from data cards. Entities are FILED in sets. Space for dimensioned or subscripted variables is RESERVED. Events are SCHEDULED. The simulation is started with a START SIMULATION statement.

The simulation will stop when a variable reaches a stated value or there are no more scheduled events. Usually the simulation is stopped after a certain period of simulated time has elapsed, i.e., TIME.V equals some value, or when a queue overflows, i.e., N.set exceeds some value, or when a certain statistic is achieved, i.e., a summary variable equals some value.

The language is very efficient but does require some programming skill. It can be used to simulate a complex system or to do general programming. Its special features make it a significant time-saver when compared to the nonsimulation higher order languages. For example, one SIMSCRIPT statement often replaces an entire FORTRAN subroutine. A drawback may be the requirement for the programmer to learn how to express algorithms by the language statements. It is often difficult to debug a large simulator due to the automatic generation of data structures and set relationships. The author chose to use SIMSCRIPT II.5 for the simulator in this study due to his

limited background in other simulation languages and due to the availability of SIMSCRIPT II.5 on the AFIT computer system.



#### IV. F-16 FIRE CONTROL COMPUTER OPERATIONAL FLIGHT PROGRAM

This chapter develops the concept of avionic software. It discusses an evaluation tool, the software test stand. A description of the Fire Control System is included that describes the system requirements or functions, system mode operations, and OFP design construction. Emphasis is made to the movement of data in the FCS. The reader is encouraged to pay particular attention to the description of the OFP modes of operation and the OFP segment discussion as the model simulates all OFP modes at the segment level.

##### Associated Terminology

"Avionic Software" is a term that is frequently used to identify vastly different concepts. To understand the relationship of an operational flight program (OFP) to the term of avionic software, the following definitions are suggested by Trainor (Ref 25:998):

1. Avionic Mission Software (MS): Software which executes in the flight computer and performs direct mission related functions. An analogous term is "embedded" software.
2. Avionic Support Software (SS): Software which is required to aid in the design and production of avionic systems (both hardware and mission software).
3. Operational Flight Program (OFP): The OFP contains those functions which execute in the flight computer in (or near) real-time and perform the primary aircraft mission functions. (Examples for the F-16 are the following functions: Executive control, navigation, guidance, stores management, fixtaking, in-flight system test, bus control, fuel management, combat energy management, data management, and targeting.)
4. Operational Test Program (OTP): A comprehensive program for flight-line diagnosis and maintenance. It usually is in the form of a flight-line loadable software package that executes in the actual flight computers while these computers are still

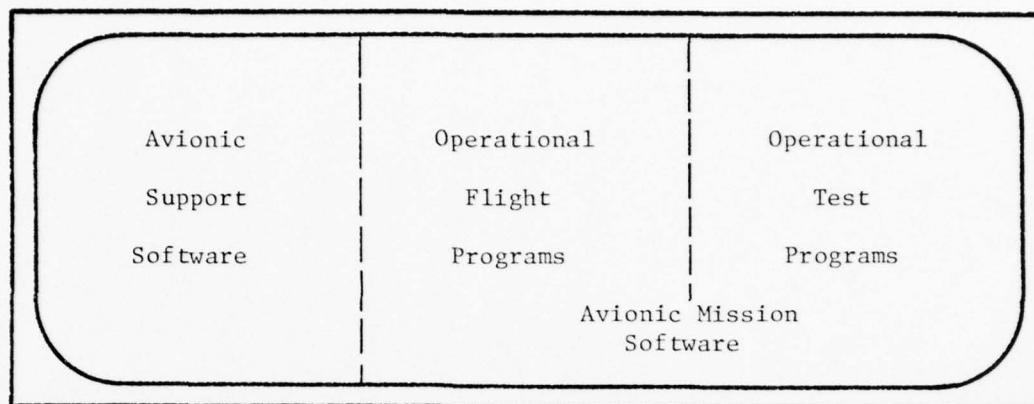


Figure 5. Avionic Software Terminology

resident in the aircraft. The program will identify system faults and will, therefore, certify the computer air worthiness. (Examples of F-16 functions include: computer health tests, input/output hardware tests, sensor operation tests, and fault/unit isolation/identification.)

The relevant terminology is depicted in Figure 5 and shows the relationships between terms.

#### Characteristic Differences of Avionic Mission Software

Avionic mission software differs from the usual Automatic Data Processing (ADP) software in several respects. Avionic mission software is used to close several multiple frequency control loops. The problems to be solved are represented in terms of radar, navigation, and weapon delivery functions, i.e., algorithms (equations and logic) with timing, and throughput constraints. This could be compared with an on-line realtime control of some manufacturing process. In addition, the software must present all pertinent information to the aircrew displays and react in realtime to insure aircrew safety. Further, the total software package must provide absolute control over weapon release so that nuclear weapons might be placed under its control.

The greatest number of inputs to the avionic computer come via analog-to-digital converters, from sensors, and from aircrew/aircraft control actions. These inputs are taken without regard to the status of the device that caused the input. Hardware failures must be detected by the software in order to provide a continuous, reliable operation. The computer is absolutely in-line and its functions cannot be bypassed, postponed, nor replaced.

An EXECUTIVE is a major segment of an OFP. It must be written such that as it interacts with other functional modules, it satisfies the critical areas of timing and control. The software is developed as a permanent configuration and is changed, normally, only as a result of system modification.

The necessity for correctness in the OFP logic has led to the use of a "software test stand" to perform a verification and validation of the critical relationships.

#### AFAL Software Test Stand

This facility provides the CPE personnel of AFAL with the real-time tool needed to evaluate the OFP. Figure 6 shows the test stand configuration (Ref 26). It contains a DEC-10 host computer, a fire control computer which has been instrumented with two hardware interfaces (SCADU, URT), the analog-to-digital interfaces required for the cockpit controls (stick, rudder, throttle), a Fire Control/Navigation Panel (FCNP), a bus monitor unit (BMU), an operator keyboard, and the display devices. The avionic support software is executed in real-time on the host computer and the mini-computers (PDP-11's). The mini-computers control the displays and the interfaces. The support software contains avionic subsystem simulation models, data analysis routines, and test operator control capabilities in modular

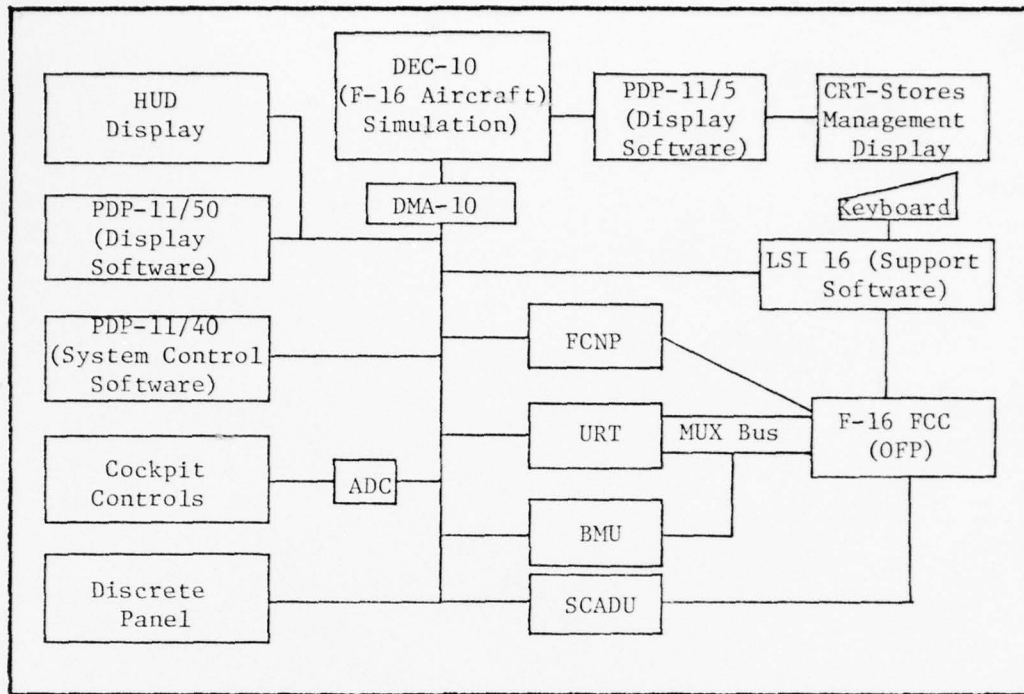


Figure 6. AFAL F-16 Software Test Stand - Detailed (Ref 26)

segments. Additionally, external environment modules provide a parametric description of the terrain, weather, targets, atmosphere, and the earth gravitational effects for use in generating test scenarios.

The Discrete Panel represents all the required cockpit switches and additional switches that allow the operator to select the simulation mode of operation. The cockpit controls allow the operator to "fly" the simulated aircraft. The BMU provides a means for a processor to monitor and record bus messages, identification tags, and related timing information. The Universal Remote Terminal (URT) simulates the operation of from one to 32 remote terminals with full subsystem avionics communication capabilities. The aircraft subsystems are simulated on the DEC-10. The Heads Up Display (HUD) gives a simulated "out-the-window" display of flight conditions. The FCNP provides pilot inputs/outputs to the total system.

The SCADU provides access to the FCC internal bus. The DMA-10 is a direct memory access unit to the DEC-10. The LSI 16 is an interfacing computer hardware unit. The Keyboard allows the operator to communicate with the DEC-10, the PDP-11's, and the FCC. The FCC is an actual Magic 362F avionic computer manufactured by DELCO Electronics.

The AFAL Software Test Stand is used to verify that the OFP actually satisfies its design functions. Program conflicts are identified through the testing of the OFP in a simulated operational environment.

#### Fire Control Computer OFP Function Descriptions

The following OFP functions are described in the "Computer Program Development Specification for the Fire Control Computer Operational Flight Program" (Ref 11):

The navigation functions of the OFP are steering and cruise energy management. The steering computations are performed in another aircraft subsystem, the Inertial Navigation Unit (INU). The OFP interfaces the INU steering problem solution to the steering display devices. The cruise energy management tasks present to the pilot the flight path guidance needed to maximize the aircraft range, endurance, or the fuel remaining at the destination.

The fixtaking functions provide the capability to update the estimate of the aircraft present position, to store the coordinates of thirteen mission planning positions for in-flight reference, and to update the estimate of the aircraft true altitude. Fixtaking tasks are not confined to the FIX mode of the OFP.

The air-to-air combat functions include the displays and controls required to fire the M61 gun and to deliver Sidewinder missiles. Combat energy management tasks provide the pilot with cues to develop and execute effective and efficient aircraft maneuvers during most air-to-air modes.

The air-to-ground attack functions include the capability to attack either preplanned or in-flight-designated surface targets. The capability includes computation for automatic and manual ballistics, course-to-release point, time-to-go, automatic weapon release, ripple-bombing, laser target identification, visual and blind deliveries, and radar freeze.

The malfunction analysis functions require the ability to collect and store FCS faults, to initiate subsystem operational self-tests, and to display the results of the subsystem self-tests. The initialization, hardware error, addressing error and power down requirements are satisfied.

The stores data function is to interface the stores management subsystem. Weapon parameters must be available to insure accurate air-to-air combat, air-to-ground attack, and energy management.

The heads up display (HUD) functions provide large amounts of data to the pilot in the form of symbols, scales, and windows. This allows the pilot to see certain critical information while he is looking out the aircraft windscreen.

Data entry/display functions include processing logic to control the FCNP, to input data into the FCC memory, to display the data associated with the FCC mode and options, and to determine a steering point.

Mission planning functions include the ability to store and access destination coordinate data, mark point coordinate data, and offset aimpoint data. This assists the pilot to increase weapon delivery effectiveness.

The support utility functions are the common utility routines shared by various segments of the OFP. There are 11 mathematical functions required by the OFP. The ability to mask and unmask bus interrupts and to provide common exit procedures from OFP segments are also required by the OFP.

Bus control functions include control of the serial data transmission over the multiplex (MUX) buses between aircraft subsystems, control of the analog-to-digital converters (ADC), reading the "sample and hold" hardware for discrete input from seven points (such as the throttle power setting and the landing gear), detection of the operational reliability of the hardware involved in the above activities, and formatting data for/after transmission on the MUX buses.

#### Fire Control Computer OFP Mode Descriptions

The OFP operates in one of fifteen mutually exclusive modes. A description of these modes is provided in the following paragraphs:

1. "Air-to-Air" Modes
  - a. Dogfight (1)
  - b. A/A Missiles (2)
  - c. Snapshot (3)
  - d. LCOS (4)

2. "FCC Air-to-Ground" Modes

- a. CCIP (5)
- b. Dive Toss (6)
- c. Beacon (7)
- d. CCRP (8)
- e. Strafe (9)
- f. LADD (10)

3. "Air-to-Ground" Modes

- a. CCIP (5)
- b. Dive Toss (6)
- c. Beacon (7)
- d. CCRP (8)
- e. Strafe (9)
- f. LADD (10)
- g. EO Weapons (11)

4. Non-Weapon Delivery Modes

- a. ALT CAL (12)
- b. FIX (13)
- c. TEST (14)
- d. Baseline (15)

The numbers in parentheses are used in the model to define the current OFP mode of operation.

The Dogfight mode computes and displays snapshoot gunnery and dynamic missile launch zone solutions. Combat energy management is not provided in this air-to-air mode.

The A/A Missiles mode computes the dynamic launch zone information (maximum and minimum missile launch ranges, aiming and aircraft lead pursuit steering commands) necessary for the use of the Sidewinder missiles against maneuvering and non-maneuvering targets. Target dynamics and missile performance determine the solution. An override feature provides automatic target acquisition. Dynamic wing twist is also calculated to provide corrections required to accurately point the missile seeker.

The Snapshoot mode displays a simulated bullet stream impact point. It provides the pilot with optimal transient firing opportunities.

The LCOS mode provides the gunnery solution to the Lead-Computing Optical Sighting (LCOS) problem. This is based on data from aircraft sensors (primarily the angular body rates of the F-16), target range rate and range, and the aiming reticle on the heads-up-display (HUD). The pilot must smoothly maneuver his aircraft to place the target within the aiming reticle. The OFP positions the aiming reticle on the HUD to indicate projectile firing path.

The CCIP mode is used to deliver air-to-ground weapons under visual conditions in either level or dive deliveries. The OFP displays on the HUD the Continuously Computed Impact Point (CCIP).

The Dive Toss mode is used to visually designate targets, to provide steering to the weapon release point, to generate automatic release of the weapon, and to compute the pull-up flight path cues on the HUD.

The Beacon mode is used to attack targets that are located by their relative position in reference to a ground-based radar beacon. The pilot proceeds as if in CCRP mode after the target is identified.

The Continuously Computed Release Point (CCRP) mode utilizes a basic coordinate bombing mechanization and the radar ground map features to provide an adverse weather, blind-attack delivery.

The strafe mode is employed to attack ground targets with the M61 gun. Weapon impact point is displayed to the pilot.

The Low Altitude Drogue Delivery (LADD) mode is provided for blind delivery of nuclear weapons. Except for the pull-up and the vertical steering information, this mode is the same as the CCRP mode.

The electro-optical (EO) weapon mode is employed to deliver television guided weapons on ground targets in either level or dive release conditions. This mode requires no processing in the implementing OFP component. EO symbology is displayed on the HUD.



The ALT CAL mode uses fire control radar (FCR) data, a computed D-value, and terrain elevation to update the estimate of the airplane altitude. The D-value is a measure of the difference between the sensed air density and the "standard day" density altitude. The accuracy of fixtaking and weapon delivery functions depend on the accuracy of the aircraft altitude estimate.

The FIX mode permits the execution of the cruise energy management tasks and associated tasks to update the aircraft position estimate. Critical fixtaking tasks are executed in the appropriate weapon delivery modes.

The Test mode is used to retrieve a maintenance fault table which consists of all faults reported during the FCC self-test. It is also used to initiate subsystem built-in-tests that interrupt the normal operation of the subsystem.

The Baseline mode has the lowest mission priority. It could be considered a "standby" mode of operation. In this mode the OFP only does the self-test tasks and essential navigation and control functions.

#### The OFP Structure

The construction of the OFP was implemented through a top down design. The highest level contains computer program components (CPC) which represent the building blocks used to implment the OFP functional requirements described on page 19. Fifteen components have been identified and are listed in Table I (Ref 11). Component interfacing is accomplished by the executive component and the sharing of data items. CPCs are partitioned to form the next design level.

The OFP is controlled by the initialization and error handling component, the system control component, the executive component, and the bus

Table I. OFP Computer Program Component Description

Component Number	Name	Symbolic Code Tag	Number of Segments
1.X	Executive	EX	4
2.X	System Control	SC	10
3.X	Bus Control	BC	18
4.X	Initialization & Error Handling	IE	5
5.X	Navigation Support	NS	5
6.X	Fixtaking	FX	17
7.X	Cruise Energy Management	CR	4
8.X	Combat Energy Management	CO	4
9.X	Air-to-Air Gunnery	GN	2
10.X	Air-to-Air Missiles	AM	4
11.X	Air-to-Ground Attack	AG	13
12.X	Stores Data Select	SS	2
13.X	Data Entry/Display	DE	1
14.X	Self-test	ST	1
15.X	Support Utility	SU	14

where X is reserved for the segment number

\*Segment Definitions are found in Appendix B.

control component. The initialization and error handling component assumes program control on power up and following the detection of certain hardware and software errors. The system control component controls the mode dependent task execution. The executive provides primary interface between components. It schedules the execution of time dependent tasks and services interrupts. It interleaves lengthy, mission-directed tasks with short, periodic service tasks. The bus control component controls the MUX buses and the input/output data flow.

The navigation support component, cruise energy management component, combat energy management component, and the air-to-ground attack component implement the associated FCC functions and provide data for the HUD. The stores data select component, self-test component, and the support utility component implement the respective OFP functions. The fixtaking component implements the fixtaking functions and provides the mission planning function of range and course identification. The air-to-air gunnery component provides the implementation of the air-to-air combat functions as they relate to the solution of the lead-computing optical sight algorithms. The air-to-air combat functions that involve the computation of the dynamic launch zone for effective air-to-air missile launch is implemented in the air-to-air missiles component. The data entry/display component satisfies the data entry/display functions, the mission planning functions related to data storage and data search, and the self-test function of test result display.

The second design level is called the segment level. Segments represent the executable portion of the OFP. A segment is the smallest partition of code that can be scheduled or entered through an interrupt. Functionally related segments are divisions within a component. Segments of the same component are scheduled within the OFP at a frequency that will

insure complete OFP mission effectiveness. A segment is, therefore, an OFP "task" or "job."

There are three general classes of interrupts: periodic, external, and error interrupts. In normal operation, error interrupts never occur. A 20 millisecond timer provides the periodic interrupts. FCC task scheduling is based on this interrupt. Each 20 millisecond period is called a minor frame. A data bus interrupt occurs after each data transfer period. A software interrupt is signaled to the executive dispatcher after each executive controlled task is completed. External interrupts occur after an infrequent data transfer fault and at the completion of an analog-to-digital data conversion. A bus data transfer fault is serviced by the bus controller.

The OFP logic relies on the mode dependent scheduling of tasks and on the structural design of the OFP. The structural design consists of periodic "time slice" tasks followed by "main loop" tasks. Main loop tasks can be considered to be "background" tasks. Time slice tasks are executed immediately following the timer or clock interrupt. This time-dependent scheduling is accomplished by the executive component.

The time slice tasks are divided in "rate groups." A rate group is a list of tasks which are grouped by frequency of execution. The rate groups are named by their frequency of execution, i.e., 50 Hz, 25 Hz, 12.5 Hz, 6.25 Hz, 3.125 Hz, 1.5625 Hz, and the 1.5625 Hz offset rate groups. Short, periodic service tasks are scheduled in the rate groups by a system control segment at a rate to satisfy the subsystem refreshment requirements. In the model "TS" is used to indicate 50 Hz tasks, "RG" is used to identify the other rate group tasks, and "ML" is used to refer to main loop tasks.

Each time slice interval contains time for the execution of two rate groups. This permits a balanced computational load as shown in Figure 7. Each "X" in Figure 7 represents the list of tasks that are all executed each time that rate group is scheduled. The interval shown in Figure 7 repeats itself every .64 seconds and is called a major frame. A major frame is thus divided into 32 minor frames. Note that there are two 1.5625 Hz rate groups, each of which executes once in a major frame.

Three basic main loop configurations implement the OFP weapon delivery functions: air-to-air missiles and gunnery, and air-to-ground attack. These tasks require lengthy computations. Cruise energy management does not require a fast execution rate and is assigned to a main loop task list. The main loop tasks are divided into two repeating lists of tasks which are called "tracks." Each track alternates execution for a specific mode-dependent time allotment. The executive component sums the execution time spent executing each track and schedules the required track to be executed next.

#### Data Movement and Bus Control

Data transmissions are grouped into blocks of data. A block contains all the data that is transmitted between two particular subsystems during a specific rate group. All the blocks of a rate group are divided into input groups and output groups. Each group is assigned to its specific rate group as a rate group task. Since it is desired that all computations be made with the most current data available and that the computation results be sent to the subsystems as soon as possible, the input task is scheduled as the first task in each rate group and the output task is scheduled as the last task in each rate group. If input data must be formatted, a task to accomplish the formatting is scheduled immediately after

Rate Group	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
50	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
25	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
12.5	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6.25	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
3.125	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
1.5625	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Offset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 7. Rate Group Execution Sequence (Ref 11)

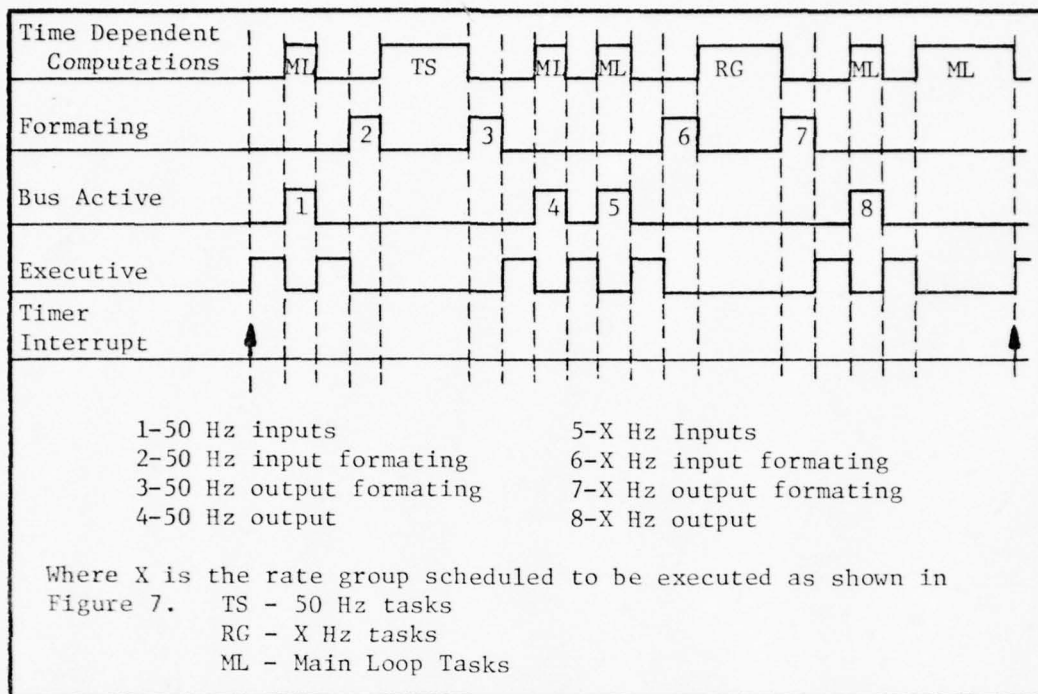


Figure 8. I/O and Computation Sequence

the input task. Output forming, if required, occurs immediately before the output task. Thus the sequence of operations within each minor frame is as shown in Figure 8.

A hardware bus controller is provided to increase central processor unit (CPU) utilization. The OFP bus component starts the bus controller at the required time and the bus controller directs the proper data movement. The CPU then begins executing main loop tasks. When the data transmission is complete, the bus controller interrupts the CPU and the rate group tasks are executed. Only during actual I/O data transmission does the hardware bus controller operate independently from the CPU.

Should the bus controller detect a transmission fault on one of the MUX buses, the other bus is used by the bus controller to retransmit the last data block. If a second fault is detected, the defective data block

is discarded and the data from the last good transmission of that data block is used as the input to the current frame time dependent computations. All data received by the FCC is double buffered in order to permit this capability.

There are three types of transmission on the MUX buses. The first type includes data blocks that are moved between the FCC and an aircraft or a FCS subsystem component. It also includes data blocks that are moved between subsystem components. The second type of data transmission appears after each subsystem-to-subsystem block transmission and is identified by the suffix SAFE. For example, "CO2 SAFE" follows transmission of the data block named "CO2." SAFE transmissions signal the receiving subsystem that it has received its last data word. It is a single data word called a command word. SAFE transmissions are detailed in Table II. The third type of data transmission appears in the 1.5625 Hz offset rate group. These transmissions are used to test the integrity of the MUX buses and the hardware bus controller. The communication formats required for different data transmissions are shown in Figure 9. A data word is sent every 20 microseconds. The control and status data words are required, as indicated, to insure that the proper devices have control of the MUX bus. The gap time duration for subsystem to (or from) FCC communications is between 2 and 5 microseconds. The gap time duration for subsystem to subsystem communications is between 4 and 10 microseconds. There is a gap time that is hidden in Figure 9 that exists between the data block transmissions. This gap time varies between 17 and 37 microseconds in duration. The average for this last gap time, however, is approximately 20 microseconds.

The data block descriptions are contained in Table III. Each data block is identified by name, word size, transmitter, receiver, transmission duration, and rate group task to which the block is assigned. These descriptions are used in Chapter V to develop the I/O tasks.



Table II. SAFE Transmissions (Ref 11)

Name	Rate Group Task	Receiver
C02 SAFE	25	HUD
C03 SAFE	25	INU
I04 SAFE	50	HUD
I05 SAFE	50	FCR
R02 SAFE	50	SMS
R03 SAFE	50	HUD
R04 SAFE	25	REO
S02 SAFE	50	HUD
S03 SAFE	1.5625	REO
S04 SAFE	50	FCR
Z01 SAFE	6.25	FCR

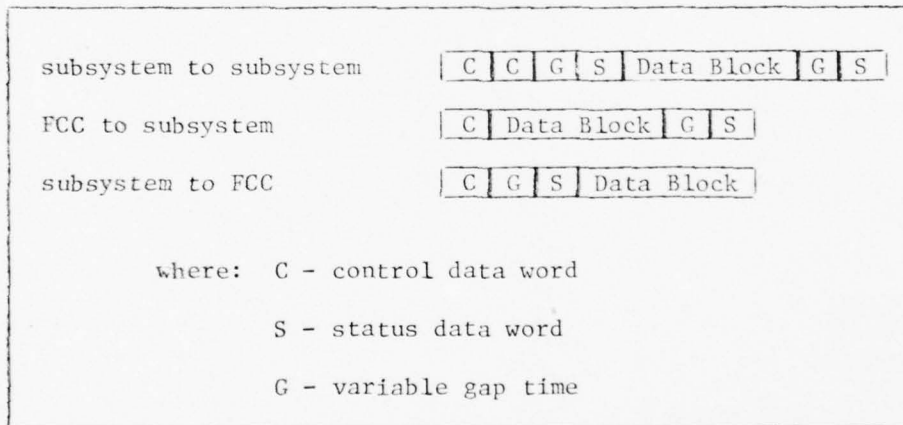


Figure 9. Data Word Transmission Formats

Table III. Data Block Descriptions

Name	Rate Group Task	Size	Transmitter	Receiver	Transmission Duration	Microsecond
E01	50	4	TISL	FCC	122-125	
F02	50	16	FCC	INU	362-365	
F03	50	24	FCC	FCR	522-525	
F05	50	4	FCC	REO	122-125	
F07	50	2	FCC	SMS	82-85	
F09	50	18	FCC	HUD	402-405	
F10	50	24	FCC	HUD	522-525	
F14	50	5	FCC	TISL	142-145	
H01	50	9	HUD	FCC	222-225	
I01	50	28	INU	FCC	602-605	
I04	50	13	INU	HUD	348-360	
I05	50	13	INU	FCR	348-360	
R01	50	21	FCR	FCC	462-465	
R02	50	3	FCR	SMS	148-160	
R03	50	5	FCR	HUD	188-200	
S02	50	4	SMS	HUD	168-180	
S03	50	4	SMS	FCR	168-180	
C01	25	10	CADC	FCC	242-245	
C02	25	9	CADC	HUD	264-270	
C03	25	2	CADC	INU	124-130	
F13	25	7	FCC	REO	182-185	
R04	25	9	FCR	REO	264-270	
F01	12.5	7	FCC	INU	182-185	
F04	12.5	8	FCC	FCNP	202-205	
I02	12.5	10	INU	FCC	242-245	
P01	12.5	4	FCNP	FCC	122-125	
F08	6.25	19	FCC	HUD	422-425	
F15	6.25	5	FCC	REO	142-145	
S01	6.25	31	SMS	FCC	662-665	
Z01	6.25	1	REO	FCR	104-110	
F12	1.5625	25	FCC	INU	542-545	
F16	1.5625	29	FCC	INU	622-625	
S03	1.5625	2	SMS	REO	124-130	
Z02	1.5625	1	REO	FCC	62-65	

### Summary

The OFP is a software program that performs primary aircraft mission functions. These functions are assigned to modules of the OFP called Computer Program Components (CPC). The CPCs are composed of segments. Segments are tasks that are executed at rates sufficient to satisfy system effectiveness requirements. Short, service segments are scheduled as tasks on a regular, periodic basis through the use of a repeating list (or queue) system. Lengthy, mission directed segments are scheduled as main loop or background tasks and execute during input/output (I/O) data transmissions and following the completion of the service tasks.

All the tasks that the OFP executes are stored in the FCC memory. An executive controls the scheduling and execution of the time dependent tasks. It responds to a timer interrupt. The duration between interrupts is 20 milliseconds and is called the minor frame. All time dependent tasks have executed at least once after a major frame. A major frame is 32 minor frames in length.

Main loop tasks and their associated service tasks are scheduled according to one of the fifteen OFP modes of operation. These mode dependent tasks are activated by the system control component and are assigned a priori to execute at a specific rate or are assigned to one of two repeating queues as background tasks. The task interaction is depicted in Figure 10. The queues in Figure 10 contain the indicated list of tasks to be executed by the OFP.

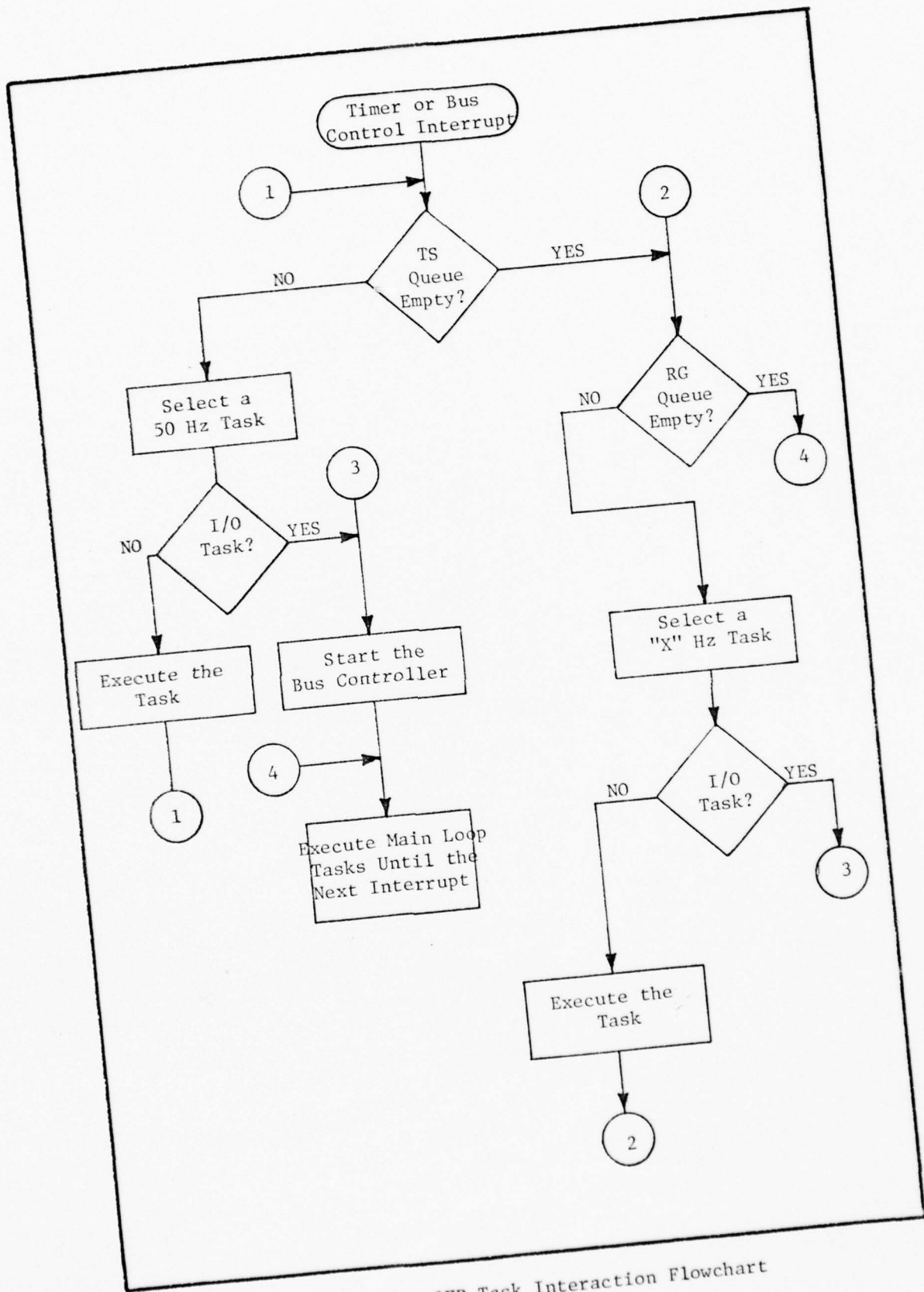


Figure 10. OFP Task Interaction Flowchart

## V. DESIGN OF THE SIMULATION MODEL

This chapter discusses the temporary entities, events, subroutines, and special features of the discrete event simulation model used to emulate the F-16 fire control computer operational flight program. The model achieves the goals listed in Chapter I by simulating the OFP at the segment or task level for all OFP modes of operation.

The approach was to develop a simplified version of the OFP that would accurately schedule the 50 Hz tasks for only one minor frame and in only one mode of operation (the DOGFIGHT mode was chosen). The model was then enhanced by adding the other rate group and main loop tasks and extending the number of minor frames that were simulated. The scheduling of the mode dependent tasks was the final feature included in the model. All segment attributes are described by deterministic values. Comments were included to provide a self-documented source program, and the event names and variable names were chosen to reflect the names used in OFP documentation. The program source listing appears in Appendix B.

An indented sentence structure was used to increase code readability. Continuation lines were indented, as were the conditional lines resulting from IF statements. Groups of comment lines were set off by the use of boxes, and all minor comments were placed after column 40 on the source program card, as needed.

As shown in Chapter IV, the OFP is a network of interacting queues. Tasks in these queues are executed on a recurring schedule. Because the two main loop tracks are repeating lists of tasks (stored in a queue), there is never a time when all the system queues are empty. This permits the combination of the task "arrival" and task "departure" events discussed

in Chapter II. Figure 11 shows the basic concept of the combination of these events.

The "Enter" point is reached on an interrupt (timer or bus controller) or at the application of system power.

#### Temporary Entities

In the simulator the 50 Hz tasks are given the name TS.JOB. The TS is to designate a 50 Hz time slice task. Tasks from other rate groups (RG) are called RG.JOBs. The main loop (ML) tasks are named ML.JOBs. Each task has the attributes of normal service time (\*\*.SERVICE) and task name (\*\*.NAME). The service time is taken from the typical execution time descriptions of the Product Spec (Ref 11) and is a fixed value throughout the simulation. The task name is an alphanumeric code. Only 10 characters are available for the name due to the simulation language and the host computer. Therefore, the task name contains the first 10 characters of the mnemonic code used in the Product Spec to identify the task with the decimal point replaced by an "X" and the slash (/) replaced by an "I."

Since TS and RG tasks may be I/O tasks, they have an attribute called TS.BLOCK and RG.BLOCK respectively. I/O tasks allow the CPU to execute main loop tasks while the bus controller completes the I/O tasks. All I/O tasks "block" or inhibit the selection of the next task from either the TS or the RG queue. This attribute is set to "I" for I/O tasks and to "O" for other tasks. This attribute is set to "2" for tasks that are scheduled only after a certain background task has executed.

The rate group tasks have an attribute that identifies its frequency of execution called QUEUE. Tasks are scheduled according to the value of QUEUE, i.e., all 6.25 Hz rate group tasks have their QUEUE value set to "6" and all of these tasks are completed when that rate group is scheduled.

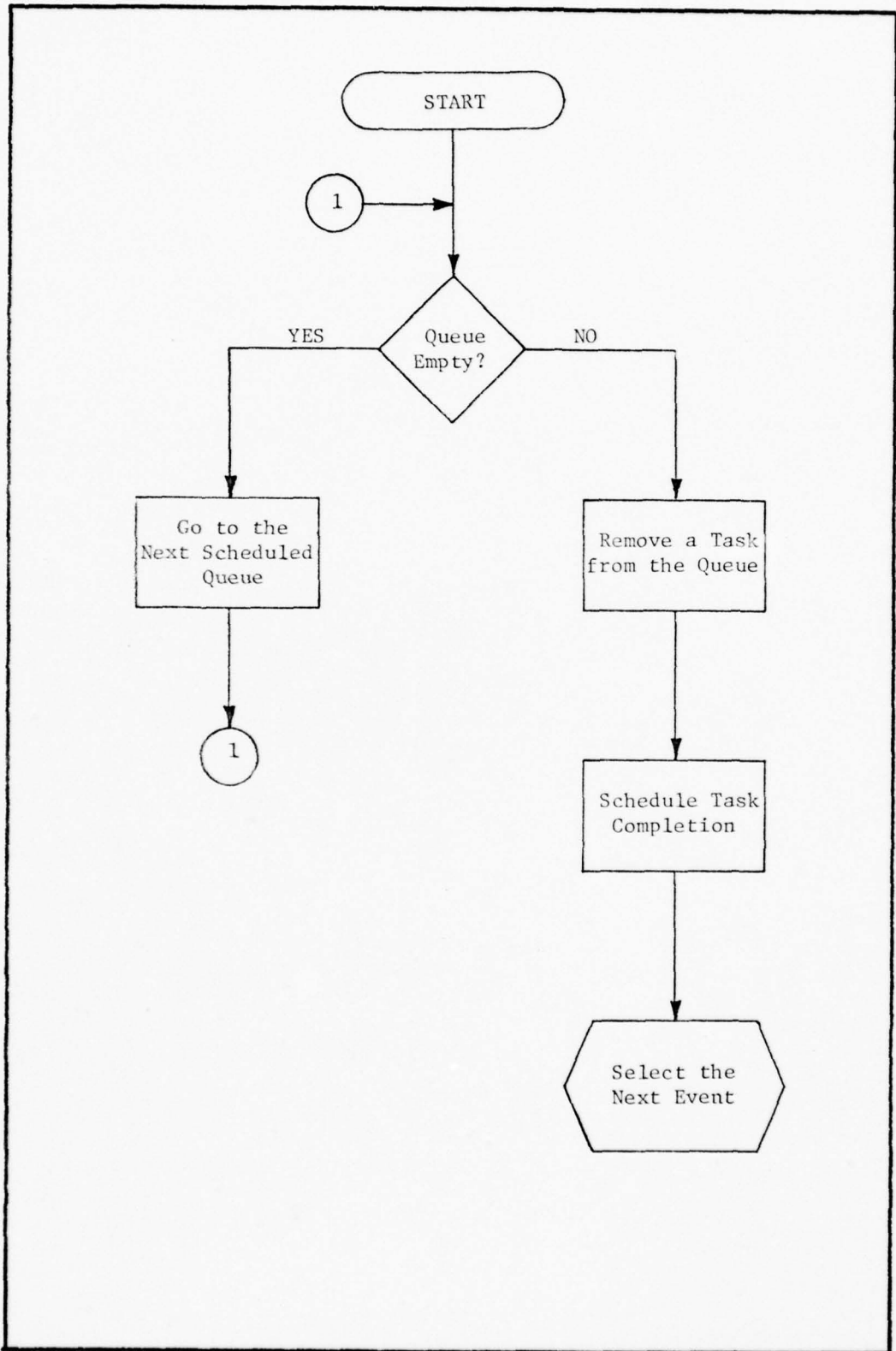


Figure 11. Task "Arrival" and "Departure" Event Combination

The QUEUE value is the integer value of the rate group name. The QUEUE value of the 1.5625 offset rate group tasks is a "2" and the 1.5625 rate group tasks are given a "1." The main loop tasks have a similar attribute called TRACK. Tasks with a TRACK value of "1" ("2") are assigned to the first (second) main loop task list.

An entity called TASK is created to get tasks into the simulation. The TASK attributes of NAME, SERVICE, JB.BLOCK, and T.QUEUE are compatible with the executed \*\*.JOB attributes. Main loop tasks have a T.QUEUE value of "51" or "52" for track #1 and track #2 identification respectively. This is necessary because the 1.5625 and offset rate groups have the values "1" and "2."

An additional attribute called DISPATCH is used to identify one of the OFP modes (see page 20) in which a task is to be executed. All tasks are read from data cards except those scheduled in a rate group after the completion of a certain main loop task. All attribute values of a task are defined on a data card. Tasks that are scheduled in all "air-to-air" modes and all "FCC air-to-ground" attack modes are given the DISPATCH values of "20" and "21" respectively. Tasks scheduled in all modes are given the value of "0." A list of task attribute values appears in Appendix B.

The method of scheduling the proper rate group is achieved by reading the rate group executing order from data cards and saving the order in a circular queue called RG.ORDER. In addition to the 50 Hz rate group, one rate group is selected on each minor cycle. The simulator, therefore, has five first-in-first-out sets to contain the rate group scheduling order and the four types of tasks: TS.QUEUE (for 50 Hz rate group tasks), RG.-QUEUE (for other tasks), ML.TRACK (for main loop tasks), INITIAL (for input tasks), and RG.ORDER (for the rate group execution sequence).



## Events

Eight events are used in the simulator. The timer interrupt is emulated by the event CLOCK. This event begins each minor frame and first stops the execution of a main loop task, if required. Then it advances the minor frame counter, FRAME, and at the end of a major frame, it collects some main loop task statistics. If a rate group task has not completed, the simulator indicates the error by calling the event TIME.OUT. If the last rate group task has completed, the rate group statistics are collected and the rate group global summation variables are set to zero. The number of major frames that have executed are checked to determine the end of the simulation. The next clock event is scheduled in 20 milliseconds and two mode initialization tasks are deleted. The arrival of a 50 Hz task is scheduled after allowing for the execution time required by the interrupt handler of the executive. A flowchart for the CLOCK event is shown in Figure 12.

The events TS.TASK and RG.TASK are conceptually identical. They start and complete the time slice (50 Hz) and other rate group tasks. Main loop tasks are first interrupted, then a task is removed from the associated queue. The last job in each queue has a zero service time. This job is used to signal when all tasks in the queue have been executed. A task is removed from the top of the queue, the service time is checked, and the rate group (main loop) list of tasks is started if the service time equals zero for a 50 Hz task (rate group task). If the service time is not zero, the same event is scheduled at the task completion time. If the \*\*.BLOCK attribute is a "1," a main loop task is begun. All tasks are returned to their respective queues except tasks whose \*\*.BLOCK value equals a "2." For tasks stored in the RG.QUEUE, only tasks with the same value in the attribute QUEUE are removed during a minor frame.

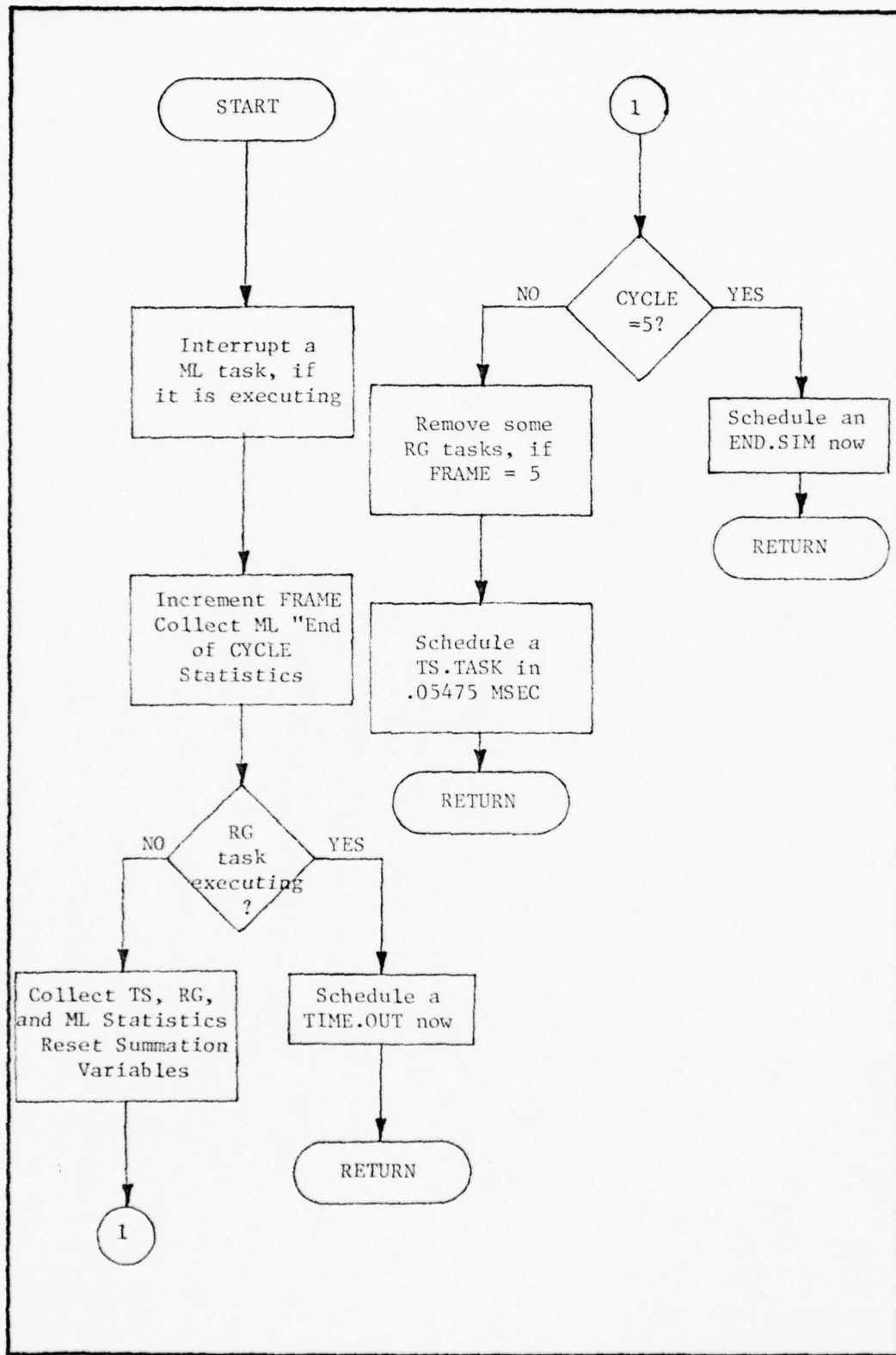


Figure 12. Event CLOCK Flowchart

The event RATE.GROUP removes the number to identify the rate group tasks to be activated during the current minor frame from the queue RG.ORDER. The value is placed in a global variable, "X." The global variable is used by the events RG.TASK and CLOCK. Rate group summation variables are reset to zero prior to starting the execution of the rate group tasks.

The event MAIN.LOOP accomplishes a similar function to the event RATE.GROUP, in that it determines which main loop track should be executed. Sequentially, the following steps are accomplished. An interrupted main loop task is restarted, when required. Next, two jobs are added to the time slice queue if the mode is FIX (13) and the execution of track #1 has completed. If the mode is FCC air-to-ground attack and track #1 has completed, one additional task is added to the time slice queue. The next track to be executed is selected based on the ratio of the time spent in each track thus far. Track #2 executes four times the rate of track #1 for modes 10 through 15, and the tracks are scheduled equally for modes 1 to 9.

The event ML.TASK starts and completes the execution of main loop tasks. It completes a task by means of a subroutine called ML.COMPLETION. A task with the correct track value is removed from the queue ML.TRACK. The interrupt variables are set to zero. If the task service time is zero (which indicates the bottom of the track has been reached), the event MAIN.LOOP is scheduled to see if the system requires a change to the current track being executed. The variable ML.FLAG is used to identify that a main loop task is being serviced. The value ML.HOLD is used to indicate that a main loop task has been interrupted. The variable ML.SAVE is a global location to save the value of the task service time so that the task completion can be rescheduled when a main loop task is restarted after being interrupted.

The event TIME.OUT prints an output message only if all rate group tasks have not been executed when the timer expires. This event then schedules the event END.SIM to stop the simulation run. This event is only entered after a fault has been detected in the event CLOCK.

#### Subroutines

The routine ML.COMPLETION turns the execution flag of the main loop tasks "off," evaluates (ML.HOLD) the time spent so far in executing the task since it was started or restarted, and adds this time to the total time (ML.TOTAL) spent in the current track.

The routine EXECUTIVE occurs only once during the initialization of the model and takes the tasks that have been stored in the INITIAL input queue and places them in their execution queues based on the current FCC operational mode. The variable DIGRAPH is used to identify the task and mode compatibility. Attributes of the TASKs stored in INITIAL are equated to the attributes of the tasks to be executed (ML.JOB, RG.JOB, TS.JOB). The job order of execution is controlled by the placement of the task data card in the input card stream. The amount of executive component overhead is supplied in the form of an executed task (OFF Segment). The EO Weapons mode (11) requires a task to be removed from the executing rate group queue. This task is executed in all other modes.

#### Special Features

The task service times are all in milliseconds and the statistics are in milliseconds. When a task completion (departure) is scheduled, it is done in "x" number of milliseconds (MSEC).

Two one-dimensional arrays, RG and HOLD, are used to collect the time (statistics) spent in executing rate group tasks. RG is for the total time

and HOLD is for the I/O time. These dimensioned variables use the value of the last current rate group ("X") as the variable index. The value of "X" is also used to add the dimensioned variable to the proper summary variable (for statistical computation).

Tasks that execute only after completion of the 3.125 rate group are assumed to execute without this restriction. This makes the scheduling of these tasks less difficult. Other scheduling simplifications include the deletion of some minor tasks and the "worst case" approach of requiring all tasks to be executed in a given mode even when the tasks require additional conditions. A summary of such tasks are: 6.15/FXPUPRC (50 Hz), 11.11/AGSTRAF and 9.1/GNINIT (25 Hz), 11.10/AGINIT and 6.6/FXDTCZE (6.25 Hz).

Some tasks are not scheduled due to the execution of a similar task. Automatic ballistics are substituted for manual ballistics tasks (11.5/AGMAN). TAKAN data is assumed to be used in the FIX mode (13) in place of RADAR data (6.12/FXPUPRH), and RANGE data is required in the FIX mode instead of either HOME (7.3/CRHOM) or ENDURANCE (7.4/CREDR) data.

## VI. RESULTS AND CONCLUSIONS

This chapter presents a comparison of the model statistics with data derived from the hand calculation of the OFP statistics (without the modeling simplifications), and from a statistical analysis (G.D. Report) provided by General Dynamics, the OFP manufacturer, for the Dogfight and CCIP modes of operation.

The effect of a long (800 minor frame) simulation run was tested and the accuracy of the I/O tasks was evaluated. The conclusions that can be made are stated and suggestions for model improvement are discussed.

### Verification Results

The G.D. Report (Ref 13) results for the Dogfight and CCIP mode are contained in Table IV. The report does not document how these results were obtained.

The hand calculation of the OFP statistics was made by following the scheduling of OFP segments for the first minor frame after power up in the two modes of interest. The results are a summation of the segment execution time by rate group, or frequency of execution, as presently documented (Ref 11).

The simulation was run for five major frames (CYCLES) in the Dogfight (DFG) mode (1) and the data comparison appears in Table IV. The execution times in the table do not include the I/O tasks assigned to the respective rate groups. A significant difference appears in some of the rate groups. For example, in the 3.125 Hz tasks the G.D. Report indicates the execution time to 6.0 milliseconds. It is assumed that the present documentation (which is a year old) does not reflect the task/queue relationships used in the G.D. Report.

Table IV. Simulation Results - Data Comparison

Rate Group	Hand Calculated		G.D. Report		Model Output	
	DFG	CCIP	DFG	CCIP	DFG	CCIP
50	2.347	3.847	2.4	4.6	2.347	3.608
25	9.365	7.207	7.5	7.2	9.365	7.207
12	1.291	1.291	1.3	1.3	1.291	1.291
6	3.707	3.702	3.3	2.5	3.607	3.607
3	0.194	0.894	6.0	6.2	0.194	0.894
1	3.132	3.132	2.8	2.8	3.132	3.132
Offset	3.257	3.257	4.4	4.4	3.257	3.257

Table V. Simulation Results - I/O Task Comparison

I/O Task	Rate Group	Model Time	BMU Time	Difference
3.1/BCI50	50	3.268	3.2497	+0.0183
3.8/BC050	50	2.304	2.3114	-0.0074
3.2/BCI25	25	1.105	1.150	-0.045
3.10/BC025	25	0.203	0.258	-0.055
3.3/BCI12	12	0.407	0.410	-0.003
3.11/BC012	12	0.427	0.434	-0.007
3.4/BCI6	6	0.851	0.864	-0.013
3.12/BC06	6	0.603	0.614	-0.011
3.16/BCI01	1	2.927	3.005	-0.078

As an additional check on the model design, the simulation was run for the CCIP (5) mode. These statistics also appear in Table IV, and the results are similar to the Dogfight mode results.

An attempt was made to obtain the actual time of segment execution from the software test stand. However, the present program source listing and the present AFAL workload did not permit this to be achieved. The output of the model will be far more accurate if the actual segment service times are used in the model of the task descriptions.

The model was run for 25 major frames to determine the effect of a longer run. There is a disadvantage to such a long simulation run and that is the requirement for 250,000 octal words of primary memory in the host computer. This is due to the fact that completed tasks are not destroyed, but are returned to their owner set. This long run did not produce a significant statistical change in the model output. The length of this run was chosen due to the repeating cycle of 800 minor frames (25 major frames or 16 seconds) as described in Chapter IV.

The output of the software test stand bus monitor unit (BMU) was used to evaluate the I/O tasks. A comparison of the BMU measured (average) I/O task execution times and the model I/O tasks is shown in Table V. The model was modified to run for only one major cycle and to track the simulated time that had expired since the last clock. This time was output after each I/O task. It was determined by this comparison that the executing times for the OFP mode dependent tasks as measured by the BMU were different from the documentation.

### Conclusions

The model does achieve all the goals established in the beginning of this work as defined in Chapter I, and should prove to be a satisfactory,



useful tool in the CPE effort of the OFP analysis. The output of the model can be used to accurately predict the time slice and main loop execution times. However, this can only be achieved with proper input data. Therefore, more accurate OFP segment execution times are needed. The model is a true reflection of the "documented" OFP. The reassignment of task/queue relationships (step 6 in the approach of Chapter I) and the subsequent operation of the model to determine the impact on the OFP, is left to the AFAL user. The reader is cautioned not to attach more than generalized significance to the model output until the segment execution times are updated.

The model will always be a good description of the OFP, provided the user keeps the task service time and the task execution conditions current. If a task is added or deleted from the OFP, the user need only insert or remove the associated data card.

#### Suggested Improvements

The G.D. Report proposed a reassignment of the I/O data transmission tasks and a regrouping of the data blocks transmitted during a task. This enhancement is left as an item for further study. It could be implemented using an I/O task queue and an event to start and stop I/O tasks. The present "blocking" attributes would no longer be required.

The G.D. Report also contained statistics on the main loop. These statistics were on the number of solutions per second and the milliseconds per execution. The generation of these particular statistics is possible in several ways. A per second statistic could be generated by an event that is called every second and only gathers the needed statistic. Per execution statistics could be gathered in the event MAIN.LOOP. The model generates statistics on solutions (turns) per major frame (Cycle) and on track execution time per major frame.

Another enhancement would be to have the model change mode automatically. This could save a large amount of time that is now spent in compiling the source program. Another desirable feature would be to reduce the amount of primary memory that is required for a longer simulation run.

## BIBLIOGRAPHY

1. Allen, Byron and Joe Clema. "The Ausail Simulator Complex," Proceedings of the Ninth Annual Simulation Symposium. 123-134. Tampa, Florida: IEEE Computer Society, 1976.
2. Archibald, W. R., and Lt. Col. E. S. Hinton. "Critical Needs in Avionic System Software," Proceedings of the IEEE 1974 National Aerospace and Electronics Conference. 475-481. Dayton, Ohio: The Institute of Electrical and Electronics Engineers, May 1974.
3. Balogh, S. E. "Multiprocessor System Performance Evaluation," Proceedings of the Twelfth CPEUG Meeting. 49-63. Washington, D.C.: The Computer Performance Users Group, November 1976.
4. Bell, Thomas E. Computer Performance Analysis: Objectives and Problems in Simulating Computers. R-1051-PR. Santa Monica, California: The Rand Corporation, July 1972.
5. Blitt, William J. and Richard Heidenreich. The Advanced Logistics System Cyber 73: A Simulation Model, MS thesis. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, December 1975. (AD-A019 841).
6. Buzen, J. P. "Modeling Computer System Performance," Proceedings of the Seventh International Conference of the Computer Measurement Group Incorporated. 230-238. Atlanta, Georgia: Computer Measurement Group, Incorporated, November 1976.
7. Delco Electronics. Magic 362F Military, Standard Avionics Computer. 576-661. Santa Barbara, California: Delco Electronics Division.
8. Des Roches, Joan C. Survey of Simulation Languages and Programs. Bedford, Massachusetts: The MITRE Corporation, July 1971. (AD 730 608).
9. Emshoff, James R. and Roger L. Sisson. Design and Use of Computer Simulation Models. London: The Macmillian Company, 1970.
10. Fishman, George S. Concepts and Methods in Discrete Event Digital Simulation. New York: John Wiley and Sons, Inc., 1973.
11. General Dynamics. Computer Program Product Specification for the F-16 Fire Control Computer Operational Flight Program. Fort Worth, Texas: General Dynamics, #16ZE011-2A, March 1976; #16ZE011-2B Vol. 1, November 1976 and Vol. 2, February 1977.
12. General Dynamics. Computer Program Users Manual for the F-16 Fire Control Computer Operational Flight Program. Fort Worth, Texas: General Dynamics, #16PR249, November 1976.
13. General Dynamics. Fire Control Computer OFP-Interval Timing Analyses. Report at the FCC OFP Timing/Duty Cycle Technical Intervhange Meeting. Fort Worth, Texas: General Dynamics, August 1977.

14. Gwynn, John M., Jr. and Randy J. Rayhor. "Performance Evaluation Through Deterministic Simulation," Proceedings of the Twelfth CPEUG Meeting. 79-83. Washington, D.C.: The Computer Performance Evaluation Users Group, November 1976.
15. Kay, Ira M. "GPSS/SIMSCRIPT - the Dominant Simulation Languages," Proceedings of the Eighth Annual Simulation Symposium. 141-154. Tampa, Florida: IEEE Computer Society, 1975.
16. MacDougall, M. H. "Computer System Simulation: An Introduction," Computing Surveys, 2 (3): 191-209 (September 1970).
17. Maisel, Herbert and Biuliano Gnugnoli. Simulation of Discrete Stochastic Systems. Chicago: Science Research Associates Inc., 1972.
18. Reifer, D. J. A Structured Approach to Modeling Computer Systems. SAMSO-TR-75-3. Los Angeles AFS, California: Space and Missile Systems Organization, August 1974. (AD A004 075).
19. Reitman, Julian. Computer Simulation Applications. New York: John Wiley and Sons, Inc., 1971.
20. Roth, Paul. "Simulation of Computers: A Tutorial Introduction," Symposium on the Simulation of Computer Systems III. Boulder, Colorado: Association of Computing Machinery, August 1975.
21. Shannon, Robert E. Systems Simulation: The Art and Science. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1975.
22. ----- . SIMSCRIPT II.5<sup>T</sup> Reference Handbook. Los Angeles: Consolidated Analysis Centers, Inc., 1976.
23. Svobodova, Liba. Computer Performance Measurement and Evaluation Methods: Analysis and Applications. New York: American Elsevier Publishing Company, Inc., 1976.
24. Trainor, W. Lynn. "Software for Avionics: An Overview," Proceedings of the IEEE 1975 National Aerospace and Electronics Conference. 227-233. Dayton, Ohio: The Institute of Electrical and Electronics Engineers, May 1975.
25. Trainor, W. Lynn. "Support Software Systems for Avionics - A Tutorial," Proceedings of the IEEE 1977 National Aerospace and Electronics Conference. 998-1003. Dayton, Ohio: The Institute of Electrical and Electronics Engineers, May 1977.
26. Zissos, Stephen N. F-16 Independent Assessment Simulator Product Specification. Draft, report number S102. Wright-Patterson AFB, Ohio: Advanced Systems Group, Air Force Avionics Laboratory, September 1976.

APPENDIX A

Computer Source Listing

This appendix contains a listing of the computer program and the data cards used as input to the simulation. Variable names that are summarized by program block in the computer program output listing have been deleted.

```

1 .....
2 **PURPOSE 1-DEFINE THE QUEUE OWNERSHIP AND MEMBERSHIP
3 **
4 **
5 **
6 **
7 **
8 **
9 **
10 **APPROACH THIS PROGRAM SIMULATES THE F-16 FIRE
11 ** CONTROL COMPUTER BY USE OF THE EVENT
12 ** SCHEDULING APPROACH AT THE SEGMENT
13 ** EXECUTION LEVEL. A SEGMENT IS CALLED A
14 ** JOB AND HAS THE FOLLOWING ATTRIBUTES:
15 ** NAME - 10 CHARACTER IDENTIFIER USED
16 ** IN THE OFF PRODUCT SPECIFICATION
17 ** SERVICE - THE TYPICAL TIME IT
18 ** REQUIRES FOR THE JOB TO EXECUTE
19 ** JR.BLOCK - AN IDENTIFIER FOR I/O
20 ** SEGMENTS - BLOCK TIME SLICE ADVANCE
21 ** T.QUEUE - AN IDENTIFIER FOR THE TYPE
22 ** OF SEGMENT - E0.47 (50), OTHER
23 ** RATE GROUPS (<50), OR MAIN LOOP
24 ** DISPATCH - AN IDENTIFIER OF THE MORE
25 ** IN WHICH THE SEGMENT EXECUTES
26 **
27 ** THE USER CAN KEEP THE MODEL CURRENT BY USING
28 ** CURRENT AND ACCURATE VALUES IN SERVICE, T.QUEUE,
29 ** AND DISPATCH
30 **
31 ** THE USER MUST SUPPLY THE DESIRED OFF MODE OF
32 ** OPERATION IN THE EVENT NAMED MAIN
33 **
34 **
35 **
36 **
37 **
38 **
39 **
40 **
41 **
42 **
43 **
44 **
45 **
46 **
47 **
48 **
49 **
50 **
51 **
52 **
53 **

```

\*\*TS = TIME SLICE  
\*\*RG = RATE GROUP  
\*\*ML = MAIN LOOP  
\*\*RG.PTR POINTS TO THE  
\*\*QUEUE(ORG.JOB) BY NAME

```

PROGRAMLE
TEMPORARY ENTITIES
EVERY TS.JOB HAS A TS.SERVICE, AND A TS.NAME, AND A
TS.BLOCK, AND BELONGS TO A TS.QUEUE
EVERY RG.JOB HAS A RG.SERVICE, AND A RG.NAME, AND A
RG.BLOCK, AND A QUEUE, AND BELONGS TO A RG.QUEUE
EVERY ML.JOB HAS A ML.SERVICE, AND A ML.NAME,
AND A TRACK, AND BELONGS TO A ML.TRACK
EVERY RG.PTR HAS A PTR.NAME,
AND BELONGS TO A RG.ORDER

```

```

54 EVERY TASK HAS A NAME, AND A SERVICE, AND A
55 JR.BLOCK, AND A T.QUEUE, AND A DISPATCH,
56 AND BELONGS TO THE INITIAL
57 THE SYSTEM, PLUS A T.QUEUE, A RG.ORDER, A ML.TRACK,
58 AN INITIAL, AND A RG.QUEUE
59 EVENT NOTICES INCLUDE CLOCK, TS.TASK, RATE.GROUP,
60 PG.TASK, MAIN.LOOP, ML.TASK,
61 TIME.OUT, END.SI
62 DEFINE MSFC TO MEAN UNITS
63 DEFINE ML.TRACK, RG.QUEUE, TS.QUEUE,
64 RG.ORDER, AND INITIAL AS FIFD SETS
65 DEFINE ML.START, ML.HOLD, ML.T1, ML.T2, ML.SUM, LAST.SUM,
66 SERVICE,
67 TS.SERVICE, RG.SERVICE, ML.SERVICE, ML.GO, RETAR, ML.TOTAL,
68 ML.SAVE, PG.TIME, R, C, BEGIN, R, TS.SUM, TS.IO,
69 PG.1, PG.2, PG.3, RG.5, RG.12, RG.23,
70 HO.1, HO.2, HO.3, HO.4, HO.12, HO.25,
71 A, O, BLOCK, T, PG.SUM, PG.IO AS REAL VARIABLES
72 DEFINE TS.NAME, RG.NAME, ML.NAME,
73 NAME AS ALPHA VARIABLES
74 DEFINE K, MODE, ML.FLAG, TS.FLAG, TS.BLOCK, RG.BLOCK,
75 NIGRAM, DISPATCH,
76 JR.BLOCK, JOB, T.QUEUE, TPX1, TPX2, F, G,
77 STD.NAME, Y, RG.FLAG, TRACK, QUEUE, Y,
78 CYCLE, AND FRAM AS INTEGER VARIABLES
79 DEFINE EXECUTIVE AS A ROUTINE
80 DEFINE ML.COMPLETION AS A ROUTINE
81 DEFINE PR AND HOLD AS REAL, 1-DIMENSIONAL ARRAYS
82 TALLY T1.MEAN AS MEAN, T1.SD AS STD.DEV OF A
83 TALLY T2.MEAN AS MEAN, T2.SD AS STD.DEV OF B
84 TALLY T3.MEAN AS MEAN, T3.SD AS STD.DEV OF C
85 TALLY T4.MEAN AS MEAN, T4.SD AS STD.DEV OF D
86 TALLY T5.MEAN AS MEAN, T5.SD AS STD.DEV OF E
87 TALLY T6.MEAN AS MEAN, T6.SD AS STD.DEV OF F
88 TALLY T7.MEAN AS MEAN OF F
89 TALLY P1 AS MEAN OF PG.1
90 TALLY P2 AS MEAN OF PG.2
91 TALLY P3 AS MEAN OF PG.3
92 TALLY P4 AS MEAN OF PG.4
93 TALLY P5 AS MEAN OF PG.5
94 TALLY P6 AS MEAN OF PG.6
95 TALLY P7 AS MEAN OF PG.7
96 TALLY H1 AS MEAN OF HO.1
97 TALLY H2 AS MEAN OF HO.2
98 TALLY H3 AS MEAN OF HO.3
99 TALLY H4 AS MEAN OF HO.4
100 TALLY H5 AS MEAN OF HO.5
101 TALLY H12 AS MEAN OF HO.12
102 TALLY H25 AS MEAN OF HO.25
END

```

CDC 6601 CADI SIMSCRIPT II.5 VERSION /4.0-00/ NOS-DE 1

```

1 .....
2 .....
3 .....
4 .....
5 .....
6 .....
7 .....
8 .....
9 .....
10 .....
11 .....
12 .....
13 .....
14 .....
15 .....
16 .....
17 .....
18 .....
19 .....
20 .....
21 .....
22 .....
23 .....
24 .....
25 .....
26 .....
27 .....
28 .....
29 .....
30 .....
31 .....
32 .....
33 .....
34 .....
35 .....
36 .....
37 .....
38 .....
39 .....
40 .....
41 .....
42 .....
43 .....
44 .....
45 .....
46 .....
47 .....
48 .....
49 .....
50 .....
51 .....
52 .....
53 .....
54 .....
55 .....
56 .....
57 .....
58 .....
59 .....
60 .....
61 .....
62 .....
63 .....
64 .....
65 .....
66 .....
67 .....
68 .....
69 .....
70 .....
71 .....
72 .....
73 .....
74 .....
75 .....
76 .....
77 .....
78 .....
79 .....
80 .....
81 .....
82 .....
83 .....
84 .....
85 .....
86 .....
87 .....
88 .....
89 .....
90 .....
91 .....
92 .....
93 .....
94 .....
95 .....
96 .....
97 .....
98 .....
99 .....
100 .....

```

```

*****
**PURPOSE 1-INITIALIZE THE QUEUES FROM DATA CARDS
**          2-SET PFC MODE AND ML,TRACK RATIO
**          3-SCHEDULE THE FIRST CLOCK AFTER POWER UP
**            DELAY AND SCHEDULE THE END OF THE SIM
*****
**METHOD 1-READ AND FILE TASKS IN APPROPRIATE QUEUES
**         2-USE A MULTIPLIER TO SCHEDULE THE REQUIRED
**           ML,TRACK
*****
**STARTS CLOCK
*****
*****
***** LINES AS FOLLOWS
*****
DATA CARDS-INPUT FILE
*****
NO. NAME BLOCK SERVICE QUEUE MODE -----
20 FOR K = 1 TO 121, 00
21 CREATE A TASK
22 READ NAME(TASK), J1,BLOCK(TASK), SERVICE(TASK),
23 T,QUEUE(TASK), DISPATCH(TASK)
24 PRINT 1 LINE WITH K, NAME(TASK), J1,BLOCK(TASK),
25 SERVICE(TASK), T,QUEUE(TASK), DISPATCH(TASK) AS FOLLOWS
*****
***** FILE TASK LAST IN INITIAL
*****
26 LOOP
27 FOR K = 1 TO 72, 00
28 CREATE A PG.PTR
29 READ PTR,NAME(PG,PTR)
30 FILE PG.PTR LAST IN PG.ORDER
31 LOOP
32 *****
33 ***** MUST SUPPLY MORE AMD RETAR VALUES FOR PROPER TASK SCHEDULING
34 *****
35 ***** MULTIPLIER FOR SCHEDULING ML TRACKS
36 *****
37 *****
38 ***** SUBROUTINE - SELECTS MORE-DEP. TASKS
39 ***** FOR CYCLE = 1 & FRAME = 0 AT 1ST CLOCK
40 ***** **FOR STATISTICS CONTROL IN "CLOCK"
41 ***** **FOR PG(X) INDEX ON THE FIRST CLOCK
42 *****
43 *****
44 *****
45 *****
46 *****
47 *****
48 *****
49 *****
50 *****
51 *****
52 *****
53 *****
54 *****
55 *****
56 *****
57 *****
58 *****
59 *****
60 *****
61 *****
62 *****
63 *****
64 *****
65 *****
66 *****
67 *****
68 *****
69 *****
70 *****
71 *****
72 *****
73 *****
74 *****
75 *****
76 *****
77 *****
78 *****
79 *****
80 *****
81 *****
82 *****
83 *****
84 *****
85 *****
86 *****
87 *****
88 *****
89 *****
90 *****
91 *****
92 *****
93 *****
94 *****
95 *****
96 *****
97 *****
98 *****
99 *****
100 *****

```



CDC 6603 CACI SIMSCRIPT II.5 VERSION /4.0-00/ NOS-9F 1

```

1  ..
2  ..
3  **PURPOSE: 1-STOP MAIN LOOP EXECUTION
4  ..
5  ..
6  2-ADVANCE THE FRAME & CYCLE
7  ..
8  3-CHECK FOR RG TASK COMPLETION
9  ..
10 4-COLLECT TS, PG, AND ML STATS
11 ..
12 5-SCHEDULE THE NEXT CLOCK INTERRUPT AND
13  INITIATE TIME SLICE TASK EXECUTION
14 ..
15 **STARTED FROM 1 MAIN, CLOCK
16 ..
17 **STARTS: CLOCK, TS, JOB, TIME, OUT, END, SIM
18 ..
19 **SIMULATES: 20 MSEC TIMER INTERRUPT AND 1.4/EKINT
20  SERVICE TIME OF .05475 MSEC
21 ..
22 **SUBROUTINES CALLED: ML, COMPLETION
23 ..
24 ..
25 EVENT CLOCK SAVING THE EVENT NOTICE
26 LET BEGIN = TIME * V
27 IF ML FLAG = 1
28 CALL ML, COMPLETION
29 CANCEL THE ML TASK
30 ALWAYS
31 LET FRAME = FRAME + 1
32 IF FRAME = 32
33 IF K = 1
34 LET R = ML, T1
35 LET C = ML, T2
36 LET F = TRK1
37 LET G = TRK2
38 ALWAYS
39 LET FRAME = 0
40 LET CYCLE = CYCLE + 1
41 ALWAYS
42 IF RG FLAG = 0
43 PRINT 1 LINE AS FOLLOWS
44 RG FLAG IS ON
45 SCHEDULE A TIME, OUT NOW
46 RETURN
47 ELSE
48 ..
49 ..
50 **ON THE FIRST CLOCK, NO TASKS HAVE EXECUTED AND BY
51 **ADJOINING THE TEPD AN ERROR WOULD BE INTRODUCED TO THE
52 **DESIGNED STATISTICS

```

```

53 LET RG(X) = RG.SUM
54 LET HOLD(X) = RG.IO
55 IF X = 1
56 IF X = 1
57 LET RG.1 = RG(1)
58 LET HO.1 = HOLD(1)
59 ALWAYS
60 IF X = 2
61 LET RG.2 = RG(2)
62 LET HO.2 = HOLD(2)
63 ALWAYS
64 IF X = 3
65 LET RG.3 = RG(3)
66 LET HO.3 = HOLD(3)
67 ALWAYS
68 IF X = 4
69 LET RG.4 = RG(4)
70 LET HO.4 = HOLD(4)
71 ALWAYS
72 IF X = 12
73 LET RG.12 = RG(12)
74 LET HO.12 = HOLD(12)
75 ALWAYS
76 IF X = 25
77 LET RG.25 = RG(25)
78 LET HO.25 = HOLD(25)
79 ALWAYS
80 LET ML.SUM = ML.T1 + ML.T2
81 LET TS.SUM = ML.SUM - LAST.SUM
82 LET LAST.SUM = ML.SUM
83 LET O = RG.TIME
84 LET I = TS.SUM
85 LET LOCK = TS.IO
86 ALWAYS
87 LET K = 0
88 LET RG.TIME = 0.
89 LET TS.SUM = 0.
90 LET IO.IO = 0.
91 IF FRAME = 0
92 LET ML.T1 = 0
93 LET ML.T2 = 0
94 LET LAST.SUM = 0.
95 LET TOW1 = 0
96 LET TRK2 = 0
97 ALWAYS
98 IF CYCLE = 1
99 SCHEDULE AN END.SIM NOW
100 RETURN
101 ELSE
102 SCHEDULE THE CLOCK IN 20 MSFC
103
104
105

```

\*\*K = 1, ON THE FIRST CLOCK

\*\*COLLECT STATS - ML.TIME PER FRAME

\*\*COLLECT STATS - TS.TIME PER FRAME

\*\*RESET RS TIME PER FRAME FOR "0" STATS

BEST AVAILABLE COPY

PAGE 10

12/09/77 10.37.22.

CDC 6600 CACI SYNOPSIS II.5 VERSION /4.0-00/ NOS-BE 1

```
125 IF FRAME = 6
126 FOR EACH JOB IN PG.QUEUE,
127 WITH PG.NAME(JOB) = "GXIGHINIT",
128 FIND THE FIRST CASE, IF NONE,
129 GO TO 'FORWARD'.
130 ALWAYS
131 REMOVE JOB FROM PG.QUEUE
132 DESTROY THE PG.JOB CALLED JOB
133 *FORWARD,
134 FOR EACH JOB IN PG.QUEUE,
135 WITH PG.NAME(JOB) = "GXIGHINIT",
136 FIND THE FIRST CASE, IF NONE,
137 GO TO 'BEYOND'.
138 ALWAYS
139 REMOVE JOB FROM PG.QUEUE
140 DESTROY THE PG.JOB CALLED JOB
141 *BEYOND,
142 ALWAYS
143 SCHEDULE A TS.TASK IN .05475 MSEC
144 RETURN
145 END
```

\*\* TIME IS FOR 1.4/EXEC

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
***
**PURPOSE: 1-STOP MAINLOOP TASK IS REQUIRED AND SAVE
            INFORMATION NEEDED TO RESTART THE ML.JOB
            WHEN IT BECOMES POSSIBLE
**
** 2-SELECT AND START THE NEXT TS.TASK AND
            SCHEDULE COMPLETION
**
** 3-CALL RATE.GROUP IS ALL TS.TASKS HAVE
            BEEN EXECUTED
**
** 4-CALL MAINLOOP IF TS.TASK INVOLVES I/O
**
**STARTED FROM: CLOCK, TS.TASK
**
**STARTS: TS.TASK, RATE.GROUP, MAINLOOP
**
**SIMULATES: 50 HZ TASK ARRIVAL/DEPARTURE EVENT
***
EVENT TS.TASK SAVING THE EVENT NOTICE      **FLAG = 1 IF ML IS EXECUTING
IF ML.FLAG = 1
  CALL ML.COMPLETION
  CANCEL THE ML.TASK
ALWAYS
REMOVE THE FIRST TS.JOB FROM TS.QUEUE      **FIND THE NEXT TS.JOB
IF TS.SERVICE(TS.JOB) = 0                  **LAST JOB IN EACH QUEUE IS 0
  LET TS.FLAG = 0
  FILE TS.JOB LAST IN THE TS.QUEUE
  SCHEDULE A RATE.GROUP NOW
  RETURN
ELSE
LET TS.SUM = TS.SUM + TS.SERVICE(TS.JOB)
LET TS.FLAG = 1
SCHEDULE A TS.TASK IN TS.SERVICE(TS.JOB) MSEC
IF TS.BLOCK(TS.JOB) = 1                    **I/O BLOCKS TS ADVANCE
  LET P = TIME.V - BEGIN + TS.SERVICE(TS.JOB)
  LET TS.IO = TS.IO + TS.SERVICE(TS.JOB)
  SCHEDULE A MAINLOOP NOW
  **THEREFORE GO TO MAINLOOP JOBS
ALWAYS
IF TS.BLOCK(TS.JOB) = 2
  RETURN
ELSE
FILE TS.JOB LAST IN THE TS.QUEUE
RETURN
END

```

BEST AVAILABLE COPY

PAGE 15

12/09/77 10.37.22.

CDC 6603 RADI SIMSCRIPT II.5 VERSION /4.0-00/ NOS-86 1

```
1 ***
2 **
3 **PROPOSED DETERMINE THE NEXT SET OF RATE.GROUP TASKS
4 ** TO BE EXECUTED
5 **
6 **METHOD THE EXECUTION SEQUENCE FOR THE SETS OF
7 ** RG.TASKS IS FILED IN THE SET "RG.ORDER"
8 **
9 **STARTEN FROM TS.TASK
10 **
11 **STARTEN RG.TASK
12 **
13 **SIMULATES: 1.1/EVSCH WITH SERVICE TIME OF .05481 MSEC
14 **
15 **
16 EVENT RATE.GROUP SAVING THE EVENT NOTICE
17 REMOVE THE FIRST RG.PTR FROM RG.ORDER **Y IS THE INDEX TO RG.O
18 LET Y = PTR.NAME(RG.PTR)
19 FILE RG.PTR LAST IN THE RG.ORDER
20 SCHEDULE A RG.TASK IN .05481 MSEC ** TIME IS 1.1/EVSCH AVERAGE
21 LET RG.SUM = .0
22 LET RG.I0 = .0
23 RETURN
24 END
25
```

BEST AVAILABLE COPY

```
1 ***
2 **
3 **PURPOSE: 1-STOP MAINLOOP TASK IF REQUIRED AND SAVE
4 ** THE INFORMATION NEEDED TO RESTART WHEN
5 ** WHEN IT BECOMES POSSIBLE
6 **
7 **
8 ** 2-SELECT AND START THE NEXT RATE.GROUP TASK
9 ** AND SCHEDULE ITS COMPLETION
10 **
11 ** 3-CALL MAINLOOP IF ALL RG.TASKS HAVE BEEN
12 ** EXECUTED OR THE RG.TASK INVOLVES I/O
13 **
14 ** *STARTED FROM: RATE.GROUP, RG.TASK
15 **
16 ** *STARTS: RG.TASK, MAINLOOP
17 **
18 ** *SIMULATES: OTHER HZ RATE GROUP TASK ARRIVAL /
19 ** DEPARTURE EVENT
20 **
21 ** *SUBROUTINES CALLED: ML.COMPLETION
22 **
23 **
24 **
25 ** EVENT RG.TASK SAVING THE EVENT NOTICE **FLAG = 1 IF ML IS EXECUTING
26 ** IF ML.FLAG = 1
27 ** CALL ML.COMPLETION
28 ** CANCEL THE ML.TASK
29 **
30 ** ALWAYS
31 ** FOR EACH RG.JOB IN RG.QUEUE, WITH QUEUE(RG.JOB) = X,
32 ** FIND THE FIRST CASE
33 ** REMOVE THE RG.JOB FROM RG.QUEUE
34 ** IF RG.SERVICE(RG.JOB) = .0
35 ** LET RG.ELAG = 0
36 ** FILE RG.JOB LAST IN RG.QUEUE
37 ** SCHEDULE A MAINLOOP NOW
38 ** RETURN
39 **
40 ** ELSE
41 ** LET RG.FLAG = 1
42 ** LET RG.SUM = RG.SUM + RG.SERVICE(RG.JOB)
43 ** LET RG.TIME = RG.TIME + RG.SERVICE(RG.JOB)
44 ** SCHEDULE A RG.TASK IN RG.SERVICE(RG.JOB) *SEC
45 ** IF RG.ALLOC(RG.JOB) = 1
46 ** LET Q = TIME.V - BEGIN + RG.SERVICE(RG.JOB)
47 ** LET RG.IO = RG.IO + RG.SERVICE(RG.JOB)
48 ** SCHEDULE A MAINLOOP NOW
49 ** *THUS GO TO A MAINLOOP JOB
50 **
51 ** ALWAYS
52 ** FILE RG.JOB LAST IN THE RG.QUEUE
53 ** RETURN
54 **
55 ** END
```

```

1 ***
2 **
3 **SUPPOSE: 1-RESTART INTERRUPTED ML.JOB, IF REQUIRED
4 **           AND SCHEDULE ITS COMPLETION
5 **
6 **           2-DETERMINE THE MAIN.LOOP TRACK TO RE
7 **             EXECUTED NEXT
8 **
9 **STARTED FROM 1 TS.TASK, PG.TASK, ML.TASK
10 **
11 **STARTS ML.TASK
12 **
13 **SIMULATES OEP EXECUTIVE FUNCTIONS
14 **
15 **
16 **
17 EVENT MAIN.LOOP SAVING THE EVENT NOTICE ** TRUE IF A ML.JOB WAS INTERRUPTED
18 IF ML.HOLD > 0. **RESTART THE OLD ML.JOB
19 LET ML.START = TIME.V
20 LET ML.FLAG = 1
21 ***
22 **
23 **FIND THE TIME STILL REQUIRED TO EXECUTE THE OLD JOB
24 **
25 **
26 LET ML.GO = ML.SAVE - ML.TOTAL
27 SCHEDULE A ML.TASK IN ML.GO MSEC
28 RETURN
29 ELSE
30 IF 400F = 17
31 THEN IF Y = 1
32 CREATE A TS.JOB
33 LET TS.NAME(TS.JOB) = "6X14IFXPOS"
34 LET TS.ALLOC(TS.JOB) = 2
35 LET TS.SERVICE(TS.JOB) = 1.716
36 FILE TS.JOB FIRST IN TS.QUEUE
37 CREATE A TS.JOB
38 FILE TS.JOB FIRST IN TS.QUEUE
39 ALWAYS
40 IF DIGRAM = 31
41 THEN IF Y = 1
42 CREATE A TS.JOB
43 LET TS.NAME(TS.JOB) = "11X91AGTGO"
44 LET TS.ALLOC(TS.JOB) = 2
45 LET TS.SERVICE(TS.JOB) = .7
46 FILE TS.JOB FIRST IN TS.QUEUE
47 ALWAYS
48 LET TS.NAME(TS.JOB) = "6X7IFXNV07"
49 LET TS.ALLOC(TS.JOB) = 2
50 LET TS.SERVICE(TS.JOB) = .016
51
52
53

```

BEST AVAILABLE COPY

CDC 6603 CDCI SIMSCRIPT II.5 VERSION /4.0-00/ NOS-RF 1 12/09/77 10.37.22. PAGE 20

```
54 ***
55 **
56 **TRACK 2 EXECUTED 4 TIMES THE RATE OF TRACK 1
57 **FOR MODES 10-15 AND IS EQUAL FOR MODES 1-9
58 **SEE VOL.2, APPENDIX 3, PART 2, PAGE 12
59 **
60 ***
61 LET Y = 2
62 IF ML.T1 < ML.T2 * RETAR
63 LET Y = 1
64 ALWAYS
65 SCHEDULE A ML.TASK NOW
66 RETURN
67 END
```



BEST AVAILABLE COPY

```

1  ***
2  **
3  **PURPOSE: 1-COLLECT TIME OF A COMPLETING ML JOB
4  **
5  **
6  ** 2-SELECT AND START THE NEXT MAIN.LOOP TASK
7  ** AND SCHEDULE ITS COMPLETION
8  **
9  ** 3-CALL MAIN.LOOP IF ALL ML.TASKS OF CURRENT
10 ** TRACK HAVE BEEN EXECUTED
11 **
12 ** STARTED FROM: MAIN.LOOP, ML.TASK
13 **
14 ** STARTS: MAIN.LOOP, ML.TASK
15 **
16 ** STIMULATES: ML TASK ARRIVAL/DEPARTURE EVENT
17 **
18 **
19 **
20 **
21 **
22 **
23 **
24 **
25 **
26 **
27 **
28 **
29 **
30 **
31 **
32 **
33 **
34 **
35 **
36 **
37 **
38 **
39 **
40 **
41 **
42 **
43 **
44 **
45 **
46 **
47 **
48 **
49 **
50 **
51 **
52 **

```

BEST AVAILABLE COPY

12/09/77 10.37.22. PAGE 24

CDC 6600 CACI SIMSCRIPT II.5 VERSION /4.0-00/ NOS-DE 1

```
1 ***
2 **
3 **PURPOSE: PRINT A MESSAGE ONLY IF THE RATE.G99JP IS
4 ** NOT DONE EXECUTING AT THE CLOCK INTERRUPT
5 **
6 **STARTED FROM: CLOCK
7 **
8 **STARTS: END.SIM
9 **
10 **SIMULATES: A FIRE CONTROL SYSTEM FAILURE
11 **
12 ***
13 EVENT TIME OUT
14 LIST ML.T1, ML.T2, CG.TIME, TIME.V, CYCLE, X, Y
15 POINT 7 LINES WITH FRAME AND MORE AS FOLLOWS
16 ***
17 THE SYSTEM EXPERIENCED A TIME OUT
18
19 IN FRAME ** OF MODE **
20
21 ***
22 SCHEDULE AN END.SIM NOW
23 RETURN
24 END
```

BEST AVAILABLE COPY

CDC 660J DACTI SIMSCRIPT II.5 VERSION /4.0-00/ NDS-RE 1

```

1 .....
2 .....
3 .....
4 .....
5 .....
6 .....
7 .....
8 .....
9 .....
10 .....
11 .....
12 .....
13 .....
14 .....
15 .....
16 .....
17 .....

```

```

EVENT END,SIM
BEGIN REPORT ON A NEW PAGE
LIST CYCLE, FRAME, TIME,V
BEGIN HEADING
SKIP 2 LINES
PRINT 3 LINES THUS

```

```

-----
STATISTICS SUMMARY
-----

```

```

18 SKIP 1 LINE
19 END OF HEADING
20 PRINT 14 LINES WITH TS,HEAD, TS,SO, TS,ATIME, BK,SD,
21 COREP, CE,MAX, DEPE, CE,SO AS FOLLOWS
TS,STATS - PER FRAME (A, CLOCK)
THE AVERAGE TIME-ALICE SERVICE TIME IS .....
THE STANDARD DEVIATION OF THE TS,SERVICE IS .....
THE AVERAGE TIME TS WAS ELICITED FOR I/O IS .....
THE STANDARD DEVIATION OF THE TS,BLOCK TIME IS .....

```

```

ML,STATS - PER FRAME (7)
THE AVERAGE MATN,LOOP TIME IS .....
THE MAXIMUM MATN,LOOP TIME IS .....

```

```

RG,STATS - PER FRAME (6)
THE AVERAGE RATE,GROUP TIME IS .....
THE STANDARD DEVIATION OF RG,TIME IS .....

```

```

22 POINT 13 LINES WITH P1, H1, P2, H2, P3, H3, R6, H5,
23 P17, H12, P25, H25 AS FOLLOWS
RG,STATS - QUEUE EXECUTION AND I/O TIMES
THE AVERAGE RG( 1) QUEUE SERVICE TIME IS .....
THE I/O TIME IS .....

```

```

THE AVERAGE RG( 2) QUEUE SERVICE TIME IS .....
THE I/O TIME IS .....

```

```

THE AVERAGE RG( 3) QUEUE SERVICE TIME IS .....
THE I/O TIME IS .....

```

```

THE AVERAGE RG( 5) QUEUE SERVICE TIME IS .....
THE I/O TIME IS .....

```

BEST AVAILABLE COPY

CDC 6603 CACI SIMSCRIPT II.5 VERSION /4.0-00/ NOS-RE 1 12/09/77 10.37.22. PAGE 27

THE AVERAGE RG(12) QUEUE SERVICE TIME IS \*.\*.\*.\*.\*  
THE I/O TIME IS \*.\*.\*.\*.\*

THE AVERAGE RG(25) QUEUE SERVICE TIME IS \*.\*.\*.\*.\*  
THE I/O TIME IS \*.\*.\*.\*.\*

24 PRINT 9 LINES WITH T1.MEAN, T1.SD, T1.TURN, T2.MEAN, T2.SD, T2.TURN AS FOLLOWS  
ML STATS--TRACK 1--TIME(ML.T1) AND TURNS (F) PER CYCLE  
THE AVERAGE EXECUTION TIME IS \*.\*.\*.\*.\* MSEC  
THE S.D. OF THE EXECUTION TIME IS \*.\*.\*.\*.\*  
THE TRACK SOLUTION RATE IS \*.\*.\*.\*.\*

ML STATS--TRACK 2--TIME(ML.T2) AND TURNS (G) PER CYCLE  
THE AVERAGE EXECUTION TIME IS \*.\*.\*.\*.\* MSEC  
THE S.D. OF THE EXECUTION TIME IS \*.\*.\*.\*.\*  
THE TRACK SOLUTION RATE IS \*.\*.\*.\*.\*

25 END \*\* OF REPORT  
26 STOP  
27 END \*\* OF EVENT

BEST AVAILABLE COPY

```
1 ***
2 **
3 **PURPOSE: 1-INTERRUPT A ML.TASK EXECUTION
4 **
5 **          2-COLLECT TIME OF THE INTERRUPTED M..TASK
6 **
7 **CALLED BY: CLOCK, TS.TASK, PG.TASK, ML.TASK
8 **
9 ***
10 ROUTINE ML.COMPLETION
11 LET ML.FLAG = C
12 LET ML.HOLD = ML.START
13 LET ML.TOTAL = ML.TOTAL + ML.HOLD
14 IF V = 1
15   LET ML.T1 = ML.T1 + ML.HOLD
16   RETURN
17 ELSE
18   LET ML.T2 = ML.T2 + ML.HOLD
19 RETURN
20 END
21 END
```

BEST AVAILABLE COPY

```

1 *** * * * *
2 **PURPOSE: MOVE JOBS FROM THE INPUT QUEUE (INITIAL)
3 ** TO THE EXECUTION QUEUES
4 **
5 **CALLED BY: MAIN
6 **
7 *** * * * *
8
9 ROUTINE EXECUTIVE
10 LET DIGRAPH = 0
11 NEXT
12 FOR EACH JOB IN INITIAL,
13 WITH DISPATCH(JOB) = DIGRAPH,
14 FIND THE FIRST CASE, IF NONE,
15 GO TO BYPASS.
16 ALWAYS
17 REMOVE JOB FROM INITIAL
18 IF T.QUEUE(JOB) = 0
19 CREATE A TS.JOB
20 LET TS.NAME(TS.JOB) = NAME(JOB)
21 LET TS.BLOCK(TS.JOB) = JB.BLOCK(JOB)
22 LET TS.SERVICE(TS.JOB) = SERVICE(JOB)
23 IF SERVICE(JOB) = 0
24 FILE TS.JOB LAST IN TS.QUEUE
25 ALWAYS
26 IF SERVICE(JOB) ^= 0
27 FILE TS.JOB FIRST IN TS.QUEUE
28 ALWAYS
29
30 ALWAYS
31 IF T.QUEUE(JOB) < 0
32 CREATE A PG.JOB
33 LET PG.NAME(PG.JOB) = NAME(JOB)
34 LET PG.BLOCK(PG.JOB) = JB.BLOCK(JOB)
35 LET PG.SERVICE(PG.JOB) = SERVICE(JOB)
36 LET QUEUE(PG.JOB) = T.QUEUE(JOB)
37 IF SERVICE(JOB) = 0
38 FILE PG.JOB LAST IN PG.QUEUE
39 ALWAYS
40 IF SERVICE(JOB) ^= 0
41 FILE PG.JOB FIRST IN PG.QUEUE
42 ALWAYS
43
44 ALWAYS
45 IF T.QUEUE(JOB) > 50
46 CREATE A ML.JOB
47 LET ML.NAME(ML.JOB) = NAME(JOB)
48 LET ML.SERVICE(ML.JOB) = SERVICE(JOB)
49 LET TRACK(ML.JOB) = T.QUEUE(JOB) - 50
50 IF SERVICE(JOB) = 0
51 FILE ML.JOB LAST IN ML.TRACK
52 ALWAYS
53 IF SERVICE(JOB) ^= 0
54 FILE ML.JOB FIRST IN ML.TRACK
55 ALWAYS

```

BEST AVAILABLE COPY

CDC 6603 CACI SCRIPT II.5 VERSION /4.0-00/ NOS-BE 1 12/09/77 10.37.22. PAGE 32

```
54 ALWAYS
55 GO TO 'NEXT'
56 *PASS*
57 IF DICQBH = 0
58   LET DICQBH = MODE
59   GO TO 'NEXT'
60 ALWAYS
61 IF DICQBH < 5
62   LET DICQBH = 20
63   GO TO 'NEXT'
64 ALWAYS
65 IF DICQBH < 11
66   LET DICQBH = 21
67   GO TO 'NEXT'
68 ALWAYS
69 IF DICQBH = 11
70   FOR EACH JOB IN PG.QHUE,
71     WITH PG.NAME(JOB) = "5X8IFXCURC",
72     FIND THE FIRST CASE, IF NONE,
73     GO TO 'ADVANCE'
74 ALWAYS
75 REMOVE JOB FROM PG.QHUE
76 DESTROY THE PG.JOB CALLED JOB
77 ALWAYS
78 *ADVANCE*
79 RETURN
80 END
```

BEST AVAILABLE COPY

\*\*\*\*\*  
 DATA CARDS-INPUT FILE  
 \*\*\*\*\*

NO.	NAME	BLOCK	SERVICE	QUEUE	MODE
1	IXIEXDISP	0	.03325	50	0
2	IXIEXISD	1	7.26400	50	0
3	IXIEXDIS	0	.15800	50	0
4	IXIEXISD	0	.20000	50	0
5	IXIEXISD	0	.50000	50	0
6	IXIEXISD	0	.10000	50	0
7	IXIEXISD	0	.01600	50	6
8	IXIEXISD	0	.07200	50	13
9	IXIEXISD	0	.70000	50	17
10	IXIEXISD	0	.09100	50	0
11	IXIEXISD	0	.70000	50	12
12	IXIEXISD	0	.50000	50	21
13	IXIEXISD	0	.50000	50	20
14	IXIEXISD	0	.50000	50	3
15	IXIEXISD	0	.50000	50	10
16	IXIEXISD	0	.50000	50	13
17	IXIEXISD	0	.50000	50	7
18	IXIEXISD	0	.50000	50	11
19	IXIEXISD	0	.60000	50	5
20	IXIEXISD	0	.40000	50	9
21	IXIEXISD	0	1.10000	50	5
22	IXIEXISD	0	.08125	50	0
23	IXIEXISD	1	2.70400	50	0
24	IXIEXISD	0	.15900	50	0
25	IXIEXISD	0	.06431	50	0
26	IXIEXISD	0	0.	50	0
27	IXIEXISD	0	.08325	25	0
28	IXIEXISD	1	1.40500	25	0
29	IXIEXISD	0	.15300	25	0
30	IXIEXISD	0	.10000	25	0
31	IXIEXISD	0	.70000	25	0
32	IXIEXISD	0	1.80000	25	0
33	IXIEXISD	0	1.40000	25	0
34	IXIEXISD	0	.50000	25	0
35	IXIEXISD	0	.90000	25	0
36	IXIEXISD	0	1.40000	25	1
37	IXIEXISD	0	2.10000	25	4
38	IXIEXISD	0	.50000	25	21
39	IXIEXISD	0	.50000	25	10
40	IXIEXISD	0	1.60000	25	2
41	IXIEXISD	0	.45800	25	20
42	IXIEXISD	0	.40000	25	20
43	IXIEXISD	0	.40000	25	11
44	IXIEXISD	0	.10000	25	5
45	IXIEXISD	0	1.40000	25	0
46	IXIEXISD	1	.20310	25	0
47	IXIEXISD	0	.15900	25	0
48	IXIEXISD	0	.00750	25	0
49	IXIEXISD	0	0.	25	0



# BEST AVAILABLE COPY

50	IXIEXNISP	0	.09325	12	0
51	IXIEXNISP	1	.40700	12	0
52	IXIEXNISP	0	.15800	12	0
53	IXIEXNISP	0	.36000	12	0
54	IXIEXNISP	0	.20000	12	0
55	IXIEXNISP	0	.20000	12	0
56	IXIEXNISP	0	.18400	12	0
57	IXIEXNISP	1	.42700	12	0
58	IXIEXNISP	0	.15800	12	0
59	IXIEXNISP	0	.02750	12	0
60	IXIEXNISP	0		12	0
61	IXIEXNISP	0	.09325	6	0
62	IXIEXNISP	1	.85100	6	0
63	IXIEXNISP	0	.15800	6	0
64	IXIEXNISP	0	.10000	6	0
65	IXIEXNISP	0	.40000	6	0
66	IXIEXNISP	0	.50000	6	0
67	IXIEXNISP	0	.10000	6	0
68	IXIEXNISP	0	.70000	6	0
69	IXIEXNISP	0	.10000	6	0
70	IXIEXNISP	0	.10000	6	1
71	IXIEXNISP	0	.20700	6	2
72	IXIEXNISP	1	1.89000	6	4
73	IXIEXNISP	0	.60700	6	0
74	IXIEXNISP	0	.15800	6	0
75	IXIEXNISP	0	.10750	6	0
76	IXIEXNISP	0		6	0
77	IXIEXNISP	0	.08325	3	0
78	IXIEXNISP	0	.02000	3	0
79	IXIEXNISP	0	.70000	3	21
80	IXIEXNISP	0	.08325	3	0
81	IXIEXNISP	0	.00750	3	0
82	IXIEXNISP	0		3	0
83	IXIEXNISP	0	.08325	1	0
84	IXIEXNISP	1	2.02700	1	0
85	IXIEXNISP	0	.15800	1	0
86	IXIEXNISP	0	2.80000	1	0
87	IXIEXNISP	0	.08325	1	0
88	IXIEXNISP	0	.00750	1	0
89	IXIEXNISP	0		1	0
90	IXIEXNISP	0	.08325	2	0
91	IXIEXNISP	0	.40000	2	0
92	IXIEXNISP	0	.08325	2	0
93	IXIEXNISP	0	2.00000	2	0
94	IXIEXNISP	0	.08325	2	0
95	IXIEXNISP	0	.00750	2	0
96	IXIEXNISP	0		2	0
97	IXIEXNISP	0	.14912	51	0
98	IXIEXNISP	0	.10000	51	0
99	IXIEXNISP	0	.00000	51	1
00	IXIEXNISP	0	.00000	51	2
*1	IXIEXNISP	0	.47.40000	51	1
*2	IXIEXNISP	0	.47.80000	51	2
*3	IXIEXNISP	0	.60000	51	20
*4	IXIEXNISP	0	.60000	51	20

BEST AVAILABLE COPY

*5	AXIIOSE	0	0	0	0	20
*6	AXIIOAMAL	0	0	0	0	20
*7	AXIIOAGIDE	0	0	0	0	21
*8	AXIIOEYEN	0	0	0	0	13
*9	AXIIOEYEN	0	0	0	0	0
*10	AXIIOEYEN	0	0	0	0	0
*11	AXIIOEYEN	0	0	0	0	0
*12	AXIIOEYEN	0	0	0	0	0
*13	AXIIOEYEN	0	0	0	0	0
*14	AXIIOEYEN	0	0	0	0	0
*15	AXIIOEYEN	0	0	0	0	0
*16	AXIIOEYEN	0	0	0	0	0
*17	AXIIOEYEN	0	0	0	0	0
*18	AXIIOEYEN	0	0	0	0	0
*19	AXIIOEYEN	0	0	0	0	0
*20	AXIIOEYEN	0	0	0	0	0
*21	AXIIOEYEN	0	0	0	0	0
*22	AXIIOEYEN	0	0	0	0	0
*23	AXIIOEYEN	0	0	0	0	0
*24	AXIIOEYEN	0	0	0	0	0
*25	AXIIOEYEN	0	0	0	0	0
*26	AXIIOEYEN	0	0	0	0	0
*27	AXIIOEYEN	0	0	0	0	0
*28	AXIIOEYEN	0	0	0	0	0
*29	AXIIOEYEN	0	0	0	0	0
*30	AXIIOEYEN	0	0	0	0	0
*31	AXIIOEYEN	0	0	0	0	0

APPENDIX B

OFP Segment Definitions

This appendix contains excerpts from the Product Spec which describe the OFP segments and their timing.

Table VI. Executive Segment Definitions

Segment No./ Symbolic Tag	Segment Task	Execution Conditions	Execution Time (MSEC)
1.1/EXSCH	Scheduler - determines which timeslice rate group is to be executed next.	Always executed as the last task in the 50 Hz rate group.	0.06481 (Ave.)
1.2/EXDISP	Dispatcher - invokes execution of all tasks, both main loop and timeslice.	Every time control of the system is passed from one task to another.	0.08325 (TS-TS) 0.15800 (ML-TS) 0.14912 (ML-ML, TS-ML, Ave.)
1.3/EXTSEND	End-of-timeslice - signals dispatcher to resume main loop execution	Always executed as the last timeslice task in every 20 MSEC time period.	0.00750
1.4/EXINT	Interrupt processors - accepts control from interrupt trap location, responds to condition which caused interrupt, returns from interrupt.	When clock or external interrupt occurs.	0.05475

Table VII. System Control Segment Definitions

Segment No./ Symbolic Tag	Segment Task	Task Queue	Execution Conditions	Execution Time (MSEC)
2.1/SCMODE	Reads external state indicators, determines FCS mode	TS 6.25 Hz	Permanently Scheduled	0.30
2.2/SCOUT	Issues subsystem commands, sets data control bits and converts output data	TS 6.25 Hz	Permanently Scheduled	1.80
2.3/SCDATA	Format symbol and scale positions	TS 25 Hz	Permanently Scheduled	1.40
2.4/SC50	Determine scheduling and execution of tasks in the 50 rate group	TS 50 Hz	Permanently Scheduled	0.20
2.5/SC25	Determine scheduling and execution of tasks in the 25 rate group	TS 25 Hz	Permanently Scheduled	0.10
2.6/SC12	Determine scheduling and execution of tasks in the 12 rate group	TS 12.5 Hz	Permanently Scheduled	0.03
2.7/SC6	Determine scheduling and execution of tasks in the 6 rate group	TS 6.5 Hz	Permanently Scheduled	0.10
2.8/SC3	Determine scheduling and execution of tasks in the 3 rate group	TS 3.125 Hz	Permanently Scheduled	0.02
2.9/SCTRK1	Determine scheduling and execution of tasks in main loop track 1	Main loop track 1	Permanently Scheduled	0.10
2.10/SCTRK2	Determine scheduling and execution of tasks in main loop track 2	Main loop track 2	Permanently Scheduled	0.72

Table VIII. Bus Control Segment Definitions (Sheet 1 of 3)

Segment No./ Symbolic Tag	Segment Task	Task Queue	Execution Conditions	Execution Time (MSEC)
3.1/BCI50	Start 50 Hz Subsystem-to-controller and subsystem-to-subsystem transmissions Read discretes Report I/O incomplete Block timeslice execution Cleared last failed command	TS 50 Hz	Run before 50 Hz application segments	3.268
3.2/BCI25	Start 25 Hz Subsystem-to-controller and subsystem-to-subsystem transmissions	TS 25 Hz	Run before 25 Hz application segments	1.105
3.3/BCI12	Start 12.5 Hz Subsystem-to-controller and subsystem-to-subsystem transmissions	TS 12.5 Hz	Run before 12.5 Hz application segments	0.407
3.4/BCI6	Start 6.25 Hz subsystem-to-controller and subsystem-to-subsystem transmissions	TS 6.25	Run before 6.25 Hz application segments	0.851
3.5/BCIF50	Format 50 Hz inputs	TS 50 Hz	Run after 3.1 Run before 50 Hz application segments	0.700
3.6/BCIF25	Format 25 Hz inputs	TS 25 Hz	Run after 3.2 Run before 25 Hz application segments	0.300
3.7/BCIF12	Format 12.5 Hz inputs	TS 12.5 Hz	Run after 3.3 Run before 12.5 Hz application segments	0.200

Table VIII. Bus Control Segment Definitions (Sheet 2 of 3)

Segment No./ Symbolic Tag	Segment Task Definition	Task Queue	Execution Conditions	Execution Time (MSEC)
3.8/BCIF6	Format 6.25 Hz inputs	TS 6.25 Hz	Run after 3.4 Run before 6.25 Hz application segments	0.400
3.9/BC050	Format 50 Hz outputs (including position update data when necessary) Start 50 Hz controller-to- subsystem transmissions (including position update block when necessary), block timeslice execution	TS 50 Hz	Run after 50 Hz application segments	0.400
3.10/BC025	Format 25 Hz outputs Start 25 Hz Controller-to- subsystem transmissions	TS 25 Hz	Run after 25 Hz application segments	0.05475
3.11/BC012	Start 12.5 Hz controller- to-subsystem transmissions	TS 12.5 Hz	Run after 12.5 Hz application segments	0.008
3.12/BC06	Start 6.25 Hz controller- to-subsystem transmissions	TS 6.25 Hz	Run after 6.25 Hz application segments	0.008
3.13/BCMERI	Respond to transmission errors; Report subsystem poll failures; Evaluate and report SDI tests	*	Poll results must be reset following use Test counter set before test started	0.074
3.14/BCMRFMI	Unblock timeslice execution	*	-	0.012

Table VIII. Bus Control Segment Definitions (Sheet 3 of 3)

Segment No./ Symbolic Tag	Segment Task	Task Queue	Execution Conditions	Execution Time (MSEC)
3.15/BCADI	Process fuel remaining signals; process fuel flow rate signals; report synchro failure; timetag A/d data	*	-	0.500
3.16/BCI01	Format 1.5625 Hz outputs Start fuel flow rate synchro conversion; Start 1.5625 Hz subsystem-to-controller, subsystem-to-subsystem, and controller-to-subsystem transmissions	TS 1.5625 Hz	Run before 1.5625 Hz application segments	1.500
3.17/BCTEST	Test A/D converters; set bus test counter; reset poll results; start bus controller tests and subsystem poll; block time-slice execution	TS 1.5625 Hz	Run before 1.5625 Hz Offset appli- cation seg- ments	0.400
3.18/BCSSC	Evaluate and filter results of subsystem poll; optimize bus transmission sequences according to poll results	TS 1.5625 Hz Offset	Run after 3.17	2.600

\* Process interrupt, processed as required



Table IX. Initialization/Error Handling Segment Definitions\*

Segment No./ Symbolic Tag	Segment Task	Execution Time (MSEC)
4.1/IEPU	Power-up Initialization	0.09
4.2/IEPDN	Power-down	0.02625
4.3/IEAP	Error-Address/Parity	0.01125
4.4/IEAR	Error-Arithmetic (discretes)	0.013
4.5/IECOP	Error-COP timer	0.01125

\* Hardware interrupts, processed as required (not scheduled).

Table X. Navigation Support Segment Definitions

Segment No./ Symbolic Tag	Segment Task	Task Queue Time	Execution Conditions	Execution Time (MSEC)
5.1/NSALT	Computes system altitude corrected for non-standard day variations	Slice 6.25 Hz	Permanently Scheduled	0.500
5.2/NSINT	Updates system altitude by integrating vertical velocity	Time Slice 50 Hz	Permanently Scheduled	0.100
5.3/NSWNS	Calculates take-off landing flag, wind parameters, and wander angle	Main Loop Track 2	Permanently Scheduled	1.200
5.4/NSDSS	Calculates dynamic side-slip for the HUD snapshot algorithm	Time Slice 25 Hz	Run only in air modes	0.458
5.5/NSNAV	Calculates the platform-to-body direction cosine matrix and various navigational parameters	Time Slice 25 Hz	Permanently Scheduled	1.800

Table XI. Fixtaking Segment Definitions (Sheet 1 of 3)

Segment No./ Symbolic Tag	Segment Task Description	Task Queue	Execution Conditions	Execution Time (MSEC)
6.1/FXNAVRG	Range calculations for Navigation, Fixtaking, CCRP, and LADD modes	TS 50 Hz	All NAV and A/a modes, CCRP, and LADD after 6.5, before 6.9	0.500
6.2/FXBCNRG	Range calculations for the Beacon mode	TS 50 Hz	Runs in Beacon mode after 6.5, before 6.9	0.500
6.3/FXDTRNG	Range calculation for Dive Toss mode	TS 50 Hz	Dive Toss and EO modes after 6.5	0.400
6.4/FXCCIPR	Range calculations for CCIP mode	TS 50 Hz	CCIP and STRAFE modes after 6.5 and 11.7	0.400
6.5/FXAGRNG	Calculates height-above terrain and range-to-target-Z from air-to-ground ranging data	TS 50 Hz	All modes before 6.1, 6.2, 6.3, and 6.4	0.300
6.6/FXDTCZE	Initializes dive toss cursor offsets	TS 50 Hz	Runs for one execution upon depressing designate in dive toss	0.016
6.7/FXNVCZE	Initializes ground map cursor offsets	TS 50 Hz	Runs once upon selecting the cursor zero option on the FCNP and when position fix is performed	0.016

Table XI. Fixtaking Segment Definitions (Sheet 2 of 3)

Segment No./ Symbolic Tag	Segment Task	Task Queue	Execution Conditions	Execution Time (MSEC)
6.8/FXCURCO	Calculates cursor rate of movement	TS 12.5 Hz	Runs in all modes except E0	0.184
6.9/FXRDISP	Calculates FCR display data	TS 25 Hz	All modes	0.100
6.10/FXHDISP	Calculates HUD display data	TS 25 Hz	All modes	0.900
6.11/FXALTCL	Corrects system altitude	TS 50 Hz	Runs once upon designation in RDR alt cal and FCR locked on after 6.5	0.081
6.12/XPUPRH	Calculates position error using RADAR or HUD data	ML TRK1 1 Hz	Runs once upon designation in RADAR or HUD fix modes	0.800
6.13/EXTRNGC	Calculates position error using TACAN data	ML TRK1 1 Hz	Runs once upon designation in TACAN fix mode	1.200
6.14/EXPOSCR	Corrects INU position	TS 50 Hz	Runs once upon completion of ML during fix	1.316
6.15/XPUPRC	Corrects range-to-selected point by amount of cursor offset	TS 50 Hz	Runs once in fix modes upon receipt of con- trol acknowledged runs before 6.1	0.032

Table XI. Fixtaking Segment Definitions (Sheet 3 of 3)

Segment No./ Symbolic Tag	Segment Task Description	Task Queue	Execution Conditions	Execution Time (MSEC)
6.16/FXTISLS	Provides TISL calculations	TS 25 Hz	Runs in all modes runs after 6.10	0.600
6.17/FXSTEER	Calculates range and bearing calculations for fixtaking and EM	ML TRK2 1 Hz	Runs in all modes	4.500

Table XII. Cruise Energy Management Segment Definitions

Segment No./ Symbolic Tag	Segment Task Definition	Task Queue*	Execution Conditions	Execution Time (MSEC)
7.1/CRCONT	Calculations of fuel, weight, and FCNP displays including predicted fuel at Home	ML TRK 2 (1 Hz, Min)	Permanently Scheduled	6.3
7.2/CRRNG	Provides HUD values for RNG Option	ML TRK 2 (12 Hz, Min)	Run when RNG is selected (after 7.1)	1.5
7.3/CRHOM	Provides HUD values for HOM Option	ML TRK 2 (12 Hz, Min)	Run when HOM is selected (after 7.1)	0.2
7.4/CREDR	Provides HUD values for EDR Option	ML TRK 2 (12 Hz, Min)	Run when EDR is selected (after 7.1)	0.2

\* TS - Time Slice  
 ML TRK 1 - Main Loop, Track one  
 ML TRK 2 - Main Loop, Track two

Table XIII. Combat Energy Management Segment Definitions

Segment No./ Symbolic Tag	Segment Task	Task Queue*	Execution Conditions	Execution Time (MSEC)
8.1/COSE	Computations of specific energy, maximum maneuver region, and the combat arena bounds	ML TRK 1 (12 Hz, Min)	Run in A/A modes**	2.2
8.2/COSER	Computation of the specific energy rate	ML TRK 1 (12 Hz, Min)	Run in A/A modes**	0.6
8.3/COSUST	Computation of the sustainable normal load factor	ML TRK 1 (12 Hz, Min)	Run in A/A modes**	0.6
8.4/COAVAL	Computations of the available normal load factor and the limit normal load factor	ML TRK 1 (12 Hz, Min)	Run in A/A modes**	1.0

\*TS - Time Slice

ML TRK 1 - Main Loop, Track One

ML TRK 2 - Main Loop, Track Two

\*\*Dogfight, air-to-air missiles, snapshot, and LCOS modes

Table XIV. Air-to-Air Gunnery Component Segment Definitions

Segment No./ Symbolic Tag	Segment Task	Task Queue	Execution Conditions	Execution Time (MSEC)
9.1/GNINIT	Gunnery parameter initialization	TS 25 Hz and 6.25 Hz	LCOS selected, change in radar lock status, or HUD angular rates not good; run once	0.2
9.2/GNAIR	Gunnery range computa- tions; projectile velocity and time of flight; pro- jectile range at one second time-of-flight; sighting solution	TS 25 Hz	After 9.1; run all the time in LCOS	2.1



Table XV. Air-to-Air Missile Segment Definitions

Segment No./ Symbolic Tag	Segment Task Definition	Task Queue	Execution Conditions	Execution Time (MSEC)
10.1/AMBUF	Buffering variables for main loop time consist- ency of data	ML TRK1	Mode selected and radar velo- city data valid	0.5
10.2/AMDLZ	Dynamic launch zone computation (VEGAS)	ML TRK1	Mode selected and radar velocity data valid	47.8
10.3/AMSL	HUD symbology and wing twist for radar	TS 25 Hz	Mode selected	1.6
10.4/AMINIT	Air-to-air missile initializations	TS 25 Hz	Once after mode selected; and before 10.1, 10.2, 10.3	0.1

AD-A048 164

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2  
A DISCRETE-EVENT DIGITAL SIMULATION MODEL OF THE F-16 FIRE CONT--ETC(U)  
DEC 77 L R HANSON

UNCLASSIFIED

AFIT/GE/MA/77-1

NL

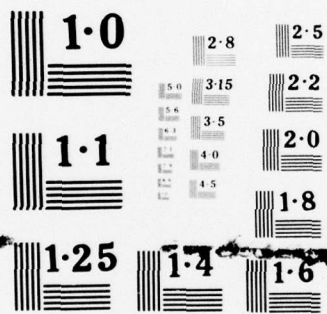
2 OF 2  
ADA  
048164



END  
DATE  
FILMED

1- 78

DDC



NATIONAL BUREAU OF STANDARDS  
MICROCOPY RESOLUTION TEST CHART

Table XVI. Air-to-Ground Segment Definitions (Sheet 1 of 4)

Segment No./ Symbolic Tag	Segment Task	Task Queue	Execution Conditions	Execution Time (MSEC)
11.1/AGIREAL	Trajectory integration for current release conditions	ML TRK1 (10 Hz, Min*)	All A/G modes after segments 11.6 and 11.10	28.8
11.2/AGI45T	Trajectory integration for 45° toss	ML TRK 2 (1 Hz, Min)	Run in dive toss, beacon, & CCRP modes after segments 11.6 and 11.10	27.9
11.3/AGILEVT	Trajectory integration for level toss	ML TRK 2	Run in dive toss mode after segments 11.6 and 11.10	27.9
11.4/AGLADD	LADD - steering and unsafe release cues; time to go to pullup	TS (25 Hz)	LADD mode after Segments 11.1 and 11.7	0.6

\*Minimum  
"AG Modes" includes CCIP, dive toss, CCRP, beacon, strafe, and LADD modes. It does  
not include EO.

Table XVI. Air-to-Ground Segment Definitions (Sheet 2 of 4)

Segment No./ Symbolic Tag	Segment Task	Task Queue	Execution Conditions	Execution Time (MSEC)
11.5/AGMAN	Computation of manual ballistics	TS (50 Hz)	AG modes with manual ballistics after segment 11.10	0.6
11.6/AGMODLS	Initialize parameters for wind, atmosphere, and coriolis models	TS (3.125 Hz)	All AG modes	0.7
11.7/AGXTRAP	Extrapolate bomb range from partials	TS (50 Hz)	All AG modes except manual ballistics	0.7
11.8/AGBKWY	Pullup anticipation, breakaway, time to go, release anticipation	TS (25 Hz)	All AG modes and EO	0.4

"AG Modes" include CCIP, dive toss, CCRP, beacon, strafe, and LADD modes. It does not include EO.

Table XVI. Air-to-Ground Segment Definitions (Sheet 3 of 4)

Segment No./ Symbolic Tag	Segment Task Definition	Task Queue	Execution Conditions	Execution Time (MSEC)
11.9/AGTCO	Release command	TS (50 Hz)	All AG modes after seg- ments 11.1 and 11.7 or after seg- ment 11.5	0.3
11.10/ACINIT	Air-to-ground mode initializations	TS (6.25 Hz)	One time before other AG seg- ments on entry to an AG mode	0.095
11.11/AGSTRAF	In-range indication	TS (25 Hz)	STRAFE (AG) guns) or CCIP rockets after segment 11.1 and 11.7	0.2

"AG modes" include CCIP, dive toss, CCRP, beacon, strafe, and LADD modes. It does not include EO.

Table XVI. Air-to-Ground Segment Definitions (Sheet 4 of 4)

Segment No./ Symbolic Tag	Segment Task	Task Queue	Execution Conditions	Execution Time (MSEC)
11.12/AGSTEER	Azimuth steering	TS (25 Hz)	Run in CCIP, dive toss, CCRP, bea- con, & LADD modes after segment 11.9/ AGTGO	0.3
11.13/AGCCIP	Bomb fall line and release anticipation for CCIP mode	TS (25 Hz)	CCIP mode after segment 11.8	0.1

"AG modes" include CCIP, dive toss, CCRP, beacon, strafe, and LADD modes. It does not include EO.

Table XVII. Stores Data Select Segment Definitions

Segment No./ Symbolic Tag	Segment Task	Task Queue*	Execution Conditions	Execution Time (MSEC)
12.1/SSINV	Process Inventory Data	Main loop Track 2	Scheduled Permanently	2.5
12.2/SSMOD	Process SMS Mode Change	Time slice 6.25 Hz	Scheduled Permanently	0.1



Table XVIII. Data Entry/Display Segment Definition

Segment No./ Symbolic Tag	Segment Task	Task Queue	Execution Conditions	Execution Time (MSEC)
13/DE	Data entry/display	TS 12.5 Hz	None	0.2

TS = Time slice

Table XIX. Self-Test Segment Definition

Segment No./ Symbol Tag	Segment Task	Task Queue	Execution Conditions	Execution Time (MSEC)
14.1/STSYS	Collects, filters, analyzes, formats, and reports avionics system faults, and tests FCC functional integrity	TS 1.5625 Hz	None	2.8

APPENDIX C

List of Abbreviations

AAF-3	Office Symbol of the Advanced Systems Group
ADC	Analog-to-Digital Converter
ADP	Automatic Data Processing
AFAL	Air Force Avionics Laboratory
AFIT	Air Force Institute of Technology
AFLC	Air Force Logistics Command
BMU	Bus Monitor Unit
CADC	Central Air Data Computer
CCIP	Continuously Computed Impact Point
CCRP	Continuously Computed Release Point
CPC	Computer Program Component
CPE	Computer Performance Evaluation
CPU	Central Processing Unit
DFG	Dogfight Mode of Operation
EO	Electro-Optical (Weapons)
FCC	Fire Control Computer
FCNP	Fire Control/Navigation Panel
FCR	Fire Control Radar
FCS	Fire Control System
G.D.	General Dynamics
HUD	Heads Up Display
INU	Inertial Navigation Unit
I/O	Input/Output (Data Movement)
LADD	Low Altitude Drogue Delivery
LCOS	Lead-Computing Optical Sighting

ML	Main Loop
MS	Avionic Mission Software
MUX	Multiplex (Buses)
OFP	Operational Flight Program
OTP	Operational Test Program
REO	Radar/EO Display Set
RG	Rate Group
SMS	Stores Management Subsystem
SS	Avionic Support Software
TISL	Target Identification Set Laser
TS	50 Hz Rate Group, Time Slice
URT	Universal Remote Terminal

VITA


Larry R. Hanson was born on 17 January 1943 in Shelley, Idaho. He was graduated from Shelley Senior High School in 1961 and attended Ricks Junior College of Rexburg, Idaho, where he received an Associate of Arts degree in 1966. He entered the USAF in July of 1966, and upon completion of Automatic Flight Control System technical training at Chanute AFB, Illinois, he was assigned to Minot AFB, North Dakota. He entered the Airman's Education and Commissioning Program at the University of Wyoming in January of 1968 and graduated in January of 1970 with a Bachelor of Science in Electrical Engineering. Upon completion of Officer Training School in May 1970, he was commissioned a 2nd Lieutenant in the USAF. He attend-d pilot training at Williams AFB, Arizona and received his rating in June 1971. The next five years were spent as a SAC combat crewmember in EC-135 aircraft at Ellsworth AFB, South Dakota. The last year he served as an Aircraft Commander. He entered the AFIT residence school in June 1976.

Permanent address: P.O. Box 425

Shelley, Idaho 83274

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/MA/77-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A DISCRETE-EVENT DIGITAL SIMULATION MODEL OF THE F-16 FIRE CONTROL COMPUTER OPERATIONAL FLIGHT PROGRAM USING SIMSCRIPT II.5	5. TYPE OF REPORT & PERIOD COVERED AFIT Thesis	
	6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s) Larry R. Hanson, Captain, USAF	8. CONTRACT OR GRANT NUMBER(s)	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) WPAFB, OH 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE December 1977	
	13. NUMBER OF PAGES 105	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED	
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; IAW AFR 190-17 <div style="border: 1px solid black; padding: 5px; display: inline-block;"> <b>DISTRIBUTION STATEMENT A</b>            Approved for public release;            Distribution Unlimited         </div> JERRAL F. GUESS, Captain, USAF Director of Information		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES APPROVED FOR PUBLIC RELEASE AFR 190-17  JERRAL F. GUESS, CAPT, USAF Director of Information		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Operational Flight Program F-16 Fire Control Computer Computer Performance Evaluation Simulation Modeling SIMSCRIPT II.5		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) In support of a request from the Air Force Avionics Laboratory, a model of the F-16 Fire Control Computer (FCC) Operational Flight Program (OFF) was developed. The initial specification required that this model allow for possible changes to the rate of accomplishment of various OFF mode-dependent tasks. The reconfiguration of the input and output tasks and the processing reserve were of particular interest. In order to determine the most useful approach, the various computer system modeling relationships are first reviewed. Based on the author's background, the real world system and the modeling goals, a discrete		

*Handwritten initials/signature*

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(Block 20 cont'd)

event queue level simulation using SIMSCRIPT II.5 is selected as the desired approach. The basic features of the F-16 FCC and the OFP are discussed and a description of the task movement in the system is provided. This description is used to detail the various rate groups and their member tasks. The model is verified by comparison against data derived from an actual system run and a statistical analysis provided by the OFP manufacturer. The verification process showed that all the original design objectives were met, although several areas of possible improvement to the model are indicated and discussed.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)