

AD-A259 329



SOFTWARE DESIGN DOCUMENT

FOR THE

GENERIC AVIONICS DATA BUS TOOL KIT

Prepared For:
Ada Joint Program Office
Ada Technology Insertion Program

Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	<i>per form</i>
By <i>SD</i>	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
<i>A-1</i>	

Prepared By:
Computer Technology and Simulation Department
Systems Engineering Test Directorate
Flight Test Engineering Group
Naval Air Warfare Center / Aircraft Division
Patuxent River NAS, MD 20670

October 4, 1991

92-33116



92 12 30 016

REPORT DOCUMENTATION PAGE

Form Approved
OPM No. 0704-0188

2

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources gathering and maintaining the data needed, and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Information and Regulatory Affairs, Office of Management and Budget, Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank) Software Design Document for the		2. REPORT DATE October 4, 1991		3. REPORT TYPE AND DATES COVERED	
4. TITLE AND SUBTITLE Software Design Document for the Generic Avionics Data Bus Tool Kit				5. FUNDING NUMBERS	
6. AUTHOR(S)					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Computer Technology & Simulation Department Systems Engineering Test Directorate Flight Test Engineering Group Naval Air Warfare Center/Aircraft Division Patuxent River NAS, MD 20670				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Ada Joint Program Office The Pentagon, Rm. 3E118 Washington, D.C. 20301-3080				10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES					
12a. DISTRIBUTION/AVAILABILITY STATEMENT				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This Software Design Document describes the structure of the Generic Avionics Data Bus Tool Kit (GADBTk), which is part of the Ada Technology Insertion Program's (ATIP) Ada binding project. The Computer Software Configuration Item (CSCI) described in this design document is a collection of related routines that will be combined to form a bus monitor application. The primary goal of the project is to produce reusable data bus software components.					
DISTRIBUTION STATEMENT Approved for public release Distribution Unlimited					
14. SUBJECT TERMS				15. NUMBER OF PAGES 107	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNCLASSIFIED		

TABLE OF CONTENTS

LIST OF FIGURES	viii
1 SCOPE	1
1.1 Identification	1
1.2 System Overview	1
1.3 Document Overview	1
2 REFERENCED DOCUMENTS	2
2.1 Government Documents	2
2.2 Non-Government Documents	2
3 PRELIMINARY DESIGN	3
3.1 CSCI Overview	3
3.1.1 CSCI Architecture	6
3.1.1.1 Bus_Interface to Monitor_Exec	7
3.1.1.2 Bus_Interface to Activity_Monitor	7
3.1.1.3 Monitor_Exec to Activity_Monitor	7
3.1.1.4 Monitor_Exec to Input_Parser	8
3.1.1.5 Monitor_Exec to Output_Formatter	8
3.1.1.6 Output_Formatter to Input_Parser	9
3.1.1.7 Output_Formatter to Activity_Monitor	9
3.1.2 System States and Modes	11
3.1.3 Memory and Processing Time Allocation	12
3.2 CSCI Design Description	14
3.2.1 Input_Parser	14
3.2.2 Output_Formatter	16
3.2.3 Activity_Monitor	18
3.2.4 Bus_Interface	19
3.2.5 Monitor_Executive	21

4 DETAILED DESIGN	23
4.1 DD CSC Input Parser	23
4.1.1 IP_User_Command_Interface.Start_Input	24
4.1.1.1 Start_Input Design Specification Constraints.	24
4.1.1.2 Start_Input Design	24
4.1.2 IP_User_Command_Interface.Shutdown_Input	25
4.1.2.1 Shutdown_Input Design Specification Constraints.	25
4.1.2.2 Shutdown_Input Design	25
4.1.3 IP_User_Command_Interface.Get_Next_Command	26
4.1.3.1 Get_Next_Command Design Specification Constraints.	26
4.1.3.2 Get_Next_Command Design	26
4.2 DD CSC Output_Formatter	27
4.2.1 OF_User_Output_Interface.Set_Mode	29
4.2.1.1 Set_Mode Design Specification Constraints.	29
4.2.1.2 Set_Mode Design	29
4.2.2 OF_User_Output_Interface.Set_Format	30
4.2.2.1 Set_Format Design Specification Constraints.	30
4.2.2.2 Set_Format Design	30
4.2.3 OF_User_Output_Interface.Set_Update_Rate	30
4.2.3.1 Set_Update_Rate Design Specification Constraints.	30
4.2.3.2 Set_Update_Rate Design	30
4.2.4 OF_User_Output_Interface.Write	31
4.2.4.1 Write Design Specification Constraints.	31
4.2.4.2 Write Design	31
4.2.5 OF_User_Output_Interface.Request_Channel	31
4.2.5.1 Request_Channel Design Specification Constraints.	32
4.2.5.2 Request_Channel Design	32

4.2.6 OF_User_Output_Interface.Release_Channel	32
4.2.6.1 Release_Channel Design Specification Constraints.	32
4.2.6.2 Release_Channel Design	32
4.2.7 OF_User_Output_Interface.Clear_Display	33
4.2.7.1 Clear_Display Design Specification Constraints.	33
4.2.7.2 Clear_Display Design	33
4.2.8 OF_User_Output_Interface.Start_Display	33
4.2.8.1 Start_Display Design Specification Constraints.	33
4.2.8.2 Start_Display Design	34
4.3 DD CSC Activity_Monitor	35
4.3.1 AM_Background_Monitor.Turn_Background_Monitor_On	36
4.3.1.1 Turn_Background_Monitor_On Design Specification Constraints.	36
4.3.1.2 Turn_Background_Monitor_On Design	37
4.3.2 AM_Background_Monitor.Set_Interface_Unit_Number	37
4.3.2.1 Set_Interface_Unit_Number Design Specification Constraints.	37
4.3.2.2 Set_Interface_Unit_Number Design	37
4.3.3 AM_Background_Monitor.Change_BM_Parameters	38
4.3.3.1 Change_BM_Parameters Design Specification Constraints.	38
4.3.3.2 Change_BM_Parameters Design	38
4.3.4 AM_Background_Monitor.Get_Bus_Activity	38
4.3.4.1 Get_Bus_Activity Design Specification Constraints.	38
4.3.4.2 Get_Bus_Activity Design	38
4.3.5 AM_Background_Monitor.Turn_Background_Monitor_Off	39
4.3.5.1 Turn_Background_Monitor_Off Design Specification Constraints.	39
4.3.5.2 Turn_Background_Monitor_Off Design	39
4.3.6 AM_Background_Monitor.Kill_Background_Monitor_Task	40
4.3.6.1 Kill_Background_Monitor_Task Design Specification Constraints.	40

4.3.6.2 Kill_Background_Monitor_Task Design	40
4.4 DD CSC Bus_Interface	41
4.4.1 BI_1553_Interface.Select_Unit	43
4.4.1.1 Select_Unit Design Specification Constraints.	43
4.4.1.2 Select_Unit Design	43
4.4.2 BI_1553_INTERFACE.Start_RT	44
4.4.2.1 Start_RT Design Specification Constraints.	44
4.4.2.2 Start_RT Design	44
4.4.3 BI_1553_INTERFACE.Stop_RT	45
4.4.3.1 Stop_RT Design Specification Constraints.	45
4.4.3.2 Stop_RT Design	45
4.4.4 BI_1553_INTERFACE.Start_BC	46
4.4.4.1 Start_BC Design Specification Constraints.	46
4.4.4.2 Start_BC Design	46
4.4.5 BI_1553_INTERFACE.Stop_BC	46
4.4.5.1 Stop_BC Design Specification Constraints.	46
4.4.5.2 Stop_BC Design	47
4.4.6 BI_1553_INTERFACE.Start_Monitor	47
4.4.6.1 Start_Monitor Design Specification Constraints.	47
4.4.6.2 Start_Monitor Design	47
4.4.7 BI_1553_INTERFACE.Stop_Monitor	48
4.4.7.1 Stop_Monitor Design Specification Constraints.	48
4.4.7.2 Stop_Monitor Design	48
4.4.8 BI_1553_INTERFACE.Transmit	49
4.4.8.1 Transmit Design Specification Constraints.	49
4.4.8.2 Transmit Design	49
4.4.9 BI_1553_INTERFACE.Receive	50

4.4.9.1 Receive Design Specification Constraints.	50
4.4.9.2 Receive Design	50
4.4.10 BI_1553_INTERFACE.Monitor_RT	50
4.4.10.1 Monitor_RT Design Specification Constraints.	51
4.4.10.2 Monitor_RT Design	51
4.5 DD CSC Monitor_Executive	52
4.5.1 Ex_1553_Monitor.Rotate	57
4.5.1.1 Rotate Design Specification Constraints.	57
4.5.1.2 Rotate Design	57
4.5.2 Ex_1553_Monitor.Initialize	57
4.5.2.1 Initialize Design Specification Constraints.	57
4.5.2.2 Initialize Design	57
4.5.3 Ex_1553_Monitor.Get_Selection_Data	58
4.5.3.1 Get_Selection_Data Design Specification Constraints.	58
4.5.3.2 Get_Selection_Data Design	58
4.5.4 Ex_1553_Monitor.Deactivate	59
4.5.4.1 Deactivate Design Specification Constraints.	59
4.5.4.2 Deactivate Design	59
4.5.5 Ex_1553_Monitor.Display_Data	60
4.5.5.1 Display_Data Design Specification Constraints.	60
4.5.5.2 Display_Data Design	60
5 CSCI DATA	61
5.1 Intra Package Global Data	61
5.2 Inter Package Global Definitions	68
6 CSCI DATA FILES	72
6.1 Data File to CSC-CSU Cross Reference	72
6.1.1 PORTS_1553.	72

8 NOTES	73
8.1 Definition Of Terms and Abbreviations	73
APPENDIX A	74
10.1 Data Flows	74
10.2 Data Stores	76
10.2 Terminators	76
10.2 Processes	76
10.2 Control Flows	78
10.3 Primitives	79
APPENDIX B	89
10 ADDITIONAL ADA STRUCTURE CHARTS FOR CSC's	89
10.1 Daul Port Manager	90
10.2 Screen Manager	91
10.3 Asynchronous Input	92
10.4 UT Bit Manipulations	93
10.5 UT Variable String	94
APPENDIX C	95
10 PROGRAM DEPENDANCY CHART FOR MONITOR APPLICATION	95

LIST OF FIGURES

Figure 1 1553 Bus Interface System Boundaries	3
Figure 2 Bus Monitor Application System Boundaries	5
Figure 3 CSC - to - CSC Interface Relationships	6
Figure 4 Monitor Application State Transition Diagram	11
Figure 5 Input_Parser Data Flow Diagram	14
Figure 6 Input_Parser Control Flow Diagram	14
Figure 7 Output_Formatter Data Flow Diagram	16
Figure 8 Output_Formatter Control Flow Diagram	17
Figure 9 Activity_Monitor Data Flow Diagram	18
Figure 10 Activity_Monitor Control Flow Diagram	18
Figure 11 Bus_Interface Data Flow Diagram	19
Figure 12 Bus_Interface Control Flow Diagram	20
Figure 13 Monitor_Executive Data Flow Diagram	21
Figure 14 Monitor_Executive Control Flow Diagram	22
Figure 15 Input Parser Ada Entity Diagram	23
Figure 16 Output Formatter Ada Entity Diagram	27
Figure 17 Activity_Monitor Ada Entity Diagram	35
Figure 18 Bus Interface Ada Entity Diagram	41
Figure 19 Monitor Executive Ada Entity Diagram	52
Table 1 Monitor Application State Transition Table	12
Table 2 Monitor Application State / CSC Table	12

1 SCOPE

1.1 Identification

This Software Design Document describes the structure of the Generic Avionics Data Bus Tool Kit (GADBTK), which is part of the Ada Technology Insertion Program's (ATIP) Ada binding project. The Computer Software Configuration Item (CSCI) described in this design document is a collection of related routines that will be combined to form a bus monitor application. The primary goal of the project is to produce reusable data bus software components.

1.2 System Overview

The Air Combat Environment Test and Evaluation Facility's (ACETEF) Maned Flight Simulator (MFS) laboratory is a man in the loop high fidelity flight simulator that was designed to provide avionics system stimulation to actual aircraft hardware components while pilots are flying simulated combat scenarios. Other ACETEF laboratories provide radio frequency (RF) simulation/stimulation and environment simulation. The output of the simulations at the various component laboratories can be combined to provide system stimulus to an actual aircraft which is suspended in an RF shielded anechoic chamber. Alternately individual subsystem can be stimulated and tested in a laboratory setting. The ACETEF is used to test the electronic systems of an aviation platform in a completely secure ground test facility at substantial cost savings over conventional flight testing. The Generic Avionics Data Bus Tool Kit (GADBTK) CSCI will be used in the ACETEF to provide simulation software interface to the actual system data buses under test. It will also supply monitor and diagnostic tools to aid in evaluating the test equipment.

Additional requirements for a robust Ada - 1553 binding were added because funding for this project was underwritten by the Ada Technology Insertion Program (ATIP) which is sponsored by the Ada Joint Program Office (AJPO) to promote Ada reuse in the DOD community.

1.3 Document Overview

This document will detail the preliminary and detailed design of each Computer Software Component (CSC) / Computer Software Unit (CSU) that comprise the Generic Avionics Data Bus Tool Kit. The preliminary design will focus on the object relationships that exist within the system. The documentation of these relationships will include pictorial information as well as text to support the design. The primary drawing convention used in the entity relationship diagrams is the Bohr method. The detailed design will specify the entire system in Ada. The document is intended to capture the rational of the design structure of the code. This document will also provide requirements traceability to show how each part of this design meets specific requirements analysis criteria.

2 REFERENCED DOCUMENTS

2.1 Government Documents

The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

STANDARDS:

Military

MIL-STD-1553B, 8 September 1986	Digital Time Division Command/Response Multiplex Data Bus
MIL-STD-1815A, 22 January 1983	Ada Programming Language
DOD-STD-2167A, 29 February 1988	Defense System Software Development

2.2 Non-Government Documents

The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

STANDARDS:

AA-EF88A-TE, February 1985	VAX Ada Programmer's Run-Time Reference Manual
F-6/89-10M, 1988	MIL-STD-1553 Designer's Guide, DDC ILC Data Device Corporation
ISBN 0-442-23805-3, 1989	Ada Quality and Style, The Software Productivity Consortium
MFS Users Manual, Volume 3	Software Style Guide and Configuration Management.

3 PRELIMINARY DESIGN

3.1 CSCI Overview

The purpose of the Generic Avionics Data Bus Tool Kit CSCI is to provide the necessary interface building blocks for applications written in the Ada programming language to transfer data on the MIL-STD-1553 multiplex data bus. The abstraction is to provide sufficient hardware independence that only the hidden low level hardware interface specific procedures need be modified to allow operation on different hardware platforms. The tool kit will provide all data structures and operators needed to interface with a standard 1553 implementation. In addition to bus primitives a set of reusable units will be developed to facilitate building 1553 applications. These units will be used to build an engineering monitor/emulator application. The bus monitor application is the actual system that the design is focused on. Throughout this design the bus interface is treated as a part of the monitor application however a separate context diagram is shown for the interface to emphasize the independent nature of that component.

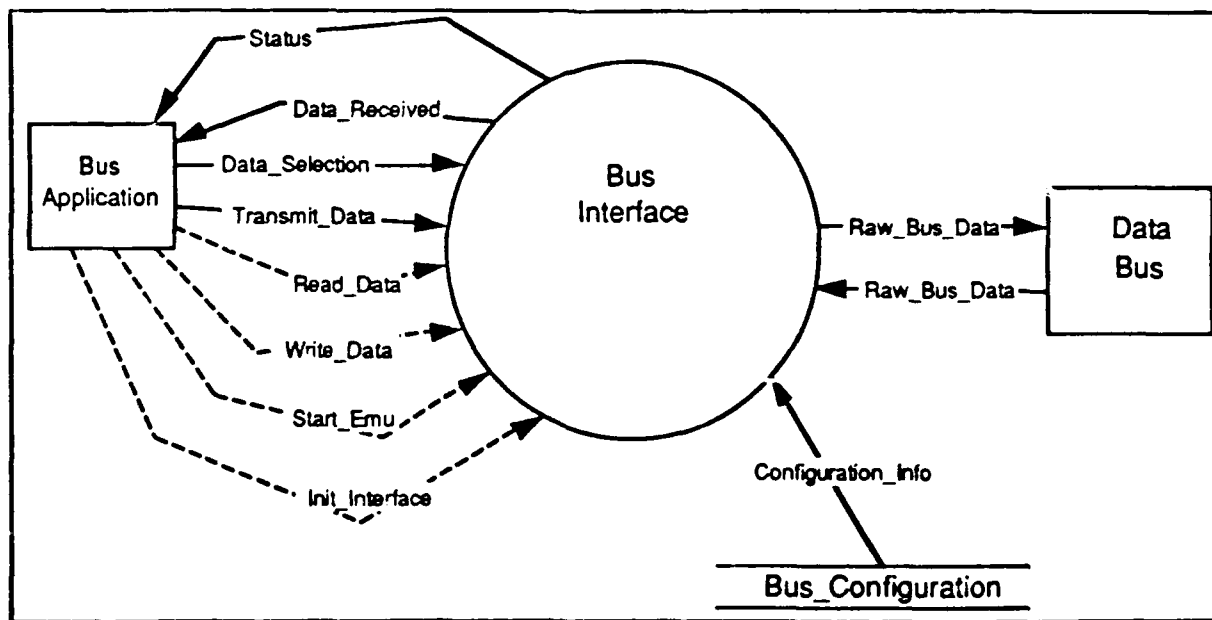


Figure 1 1553 Bus Interface System Boundaries

• INTERFACE DEFINITIONS

Configuration_Info =

BNF: Default_Bus_Number + Interface_Number + Interface_Address + Machine_Type

Data_Received =

BNF: {Bus_Word}

Data_Selection =

BNF: $RT_Num + (SA_Num) + (Tran_Rec) + (Bus_Num) + (Watch_Mask)$

Raw_Bus_Data =

BNF: {Bus_Word}

Transmit_Data =

BNF: {Bus_Word}

Status =

BNF: [Normal + Bus_Error + User_Error + Config_Error]

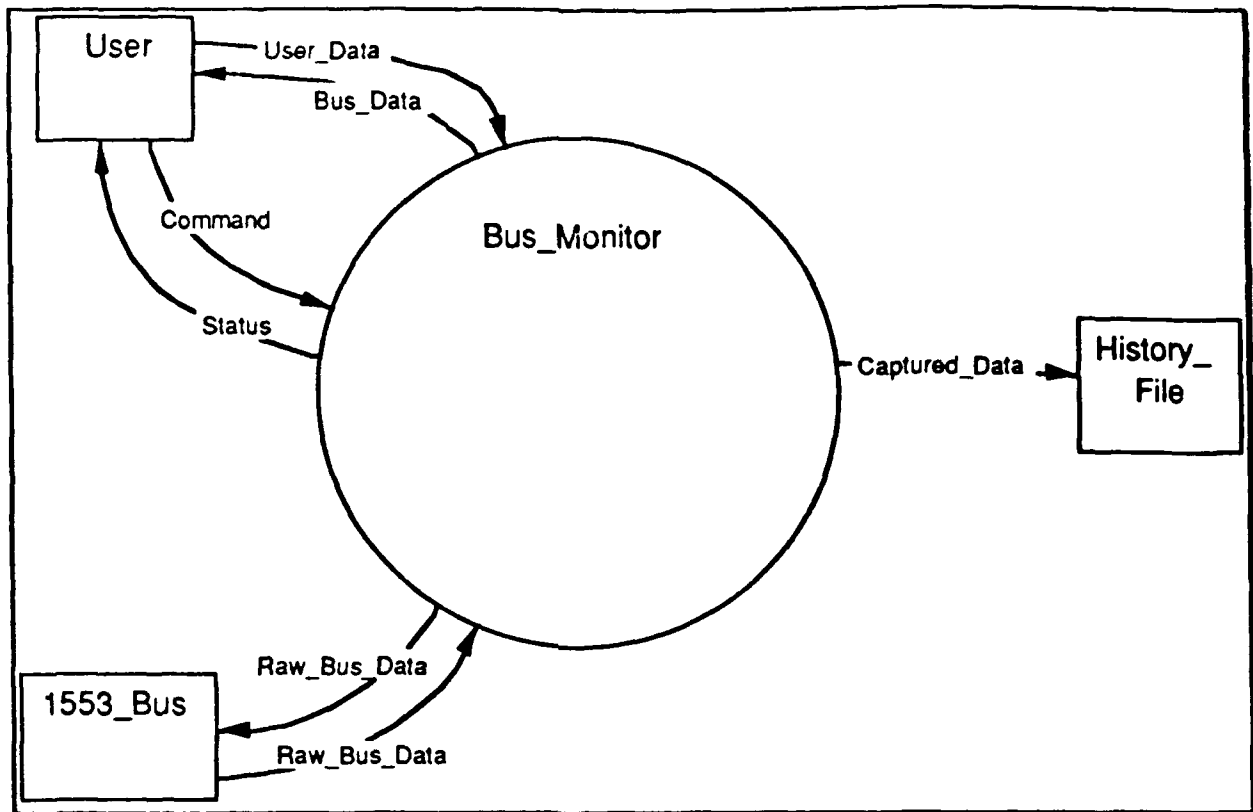


Figure 2 Bus Monitor Application System Boundaries

• INTERFACE DEFINITIONS

Bus_Data =

BNF: [RT_Status | Bus_Message]

Captured_Data =

BNF: {Bus_Word}

User_Data =

BNF: [RT_Num | SA_Num | Tran_Rec | Bus_Num | Bus_Message | Out_File |

Update_Rate | Output_Mode | Watch_Mask]

Command =

BNF: {Character}

Comments: Command input from user is an unformatted string that must match one of the valid commands.

Data Type: String

Data Representation: alpha_numeric

Data Size: 80 bytes

3.1.1 CSCI Architecture

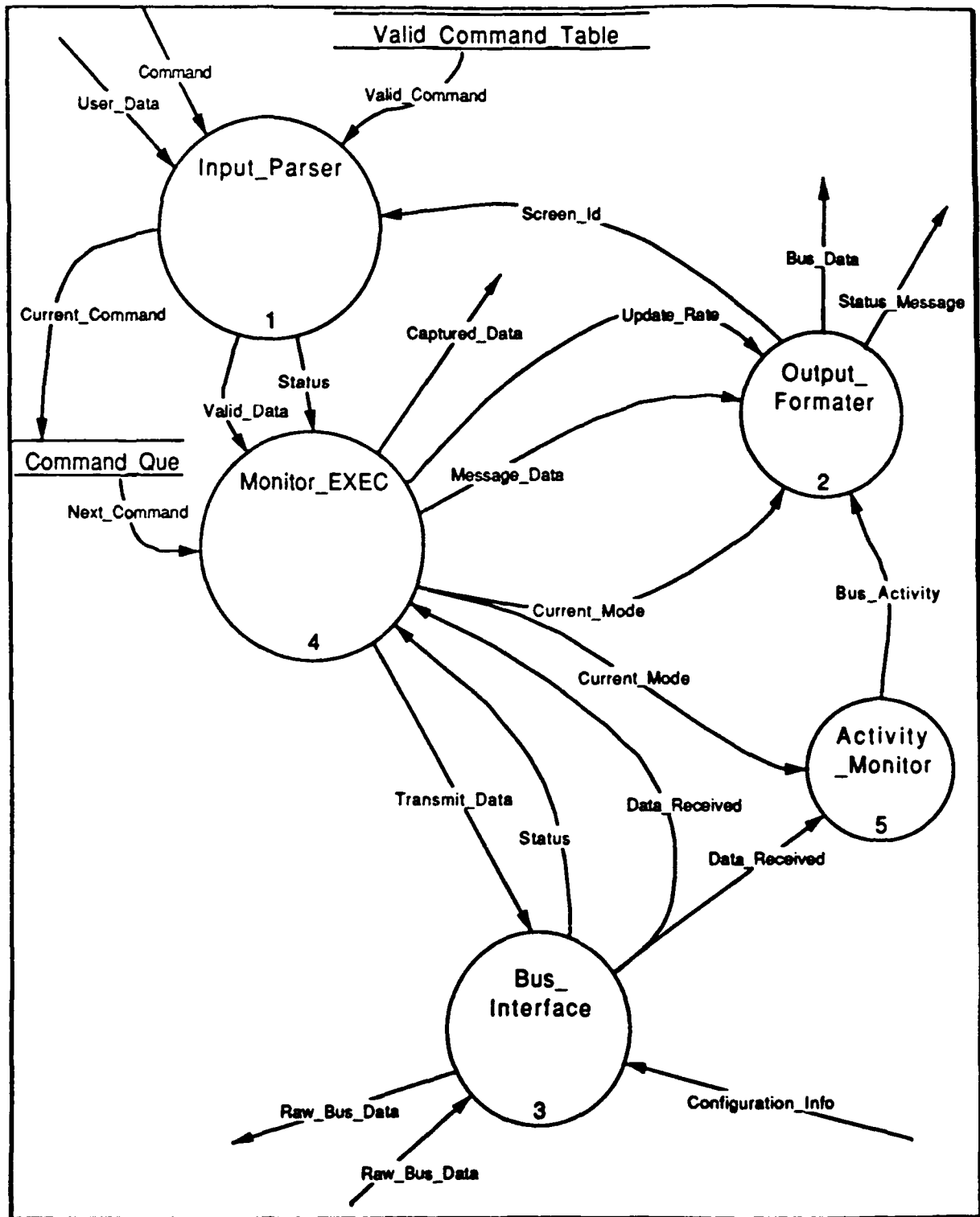


Figure 3 CSC - to - CSC Interface Relationships

3.1.1.1 Bus_Interface to Monitor_Exec

Transmit_Data =

BNF: {Bus_Word}

Bus_Word =

Data Type: Unsigned_Word

Data Representation: numeric

Data Size: 2 bytes

Data Range: (0, 65536)

Bits: 16

Status =

BNF: [Normal + Bus_Error + User_Error + Config_Error]

Data_Received =

BNF: {Bus_Word}

Bus_Word =

Data Type: Unsigned_Word

Data Representation: numeric

Data Size: 2 bytes

Data Range: (0, 65536)

Bits: 16

3.1.1.2 Bus_Interface to Activity_Monitor

Data_Received =

BNF: {Bus_Word}

Bus_Word =

Data Type: Unsigned_Word

Data Representation: numeric

Data Size: 2 bytes

Data Range: (0, 65536)

Bits: 16

3.1.1.3 Monitor_Exec to Activity_Monitor

Current_Mode =

**BNF: [Hex + Decimal + Octal + Binary + Formatted + Activity + Preferred_Data + Idle
+ Logging]**

Data Type: Mode_Type

3.1.1.4 Monitor_Exec to Input_Parser

Command_Queue =

BNF: {Current_Command}

Current_Command =

BNF: [Select_RT + Select_SA + Select_TR + Select_Bus + Set_Display_Mode
+ Set_Log_Mode + Start_Logging]

Data Type: Command_Type

Data Representation: Enumeration

Next_Command =

BNF: Current_Command

Current_Command =

BNF: [Select_RT + Select_SA + Select_TR + Select_Bus + Set_Display_Mode
+ Set_Log_Mode + Start_Logging]

Data Type: Command_Type

Data Representation: Enumeration

Valid_Data =

BNF: [Integer | String | Enumeration]

Status =

BNF: [Normal + Bus_Error + User_Error + Config_Error]

3.1.1.5 Monitor_Exec to Output_Formatter

Update_Rate =

BNF: Seconds

Comments: This message to the output formatter expresses how often to refresh the screen.

Data Type: Integer

Data Representation: numeric

Data Range: (0, 60)

Default Value: 3

Units of Measure: Seconds

Accuracy: 1

Message_Data =

BNF: Message_ID_Block + Message_Content

Message_ID_Block =

BNF: Message_Type + (Window_Id)

Message_Content =

BNF: [{Character} | {Bus_Word}]

Comments: This field contains the text of the message to be displayed

Data Type: String

Data Representation: alphanumeric

Message_Type =

BNF: [Info + Error + Bus_Data + Prompt]

Data Type: Output_Message_Type

Data Representation: enumeration

Window_Id =

Data Type: Integer

Data Representation: numeric

Data Size: 1 byte

Data Range: (1, 4)

Default Value: 1

Current_Mode =

BNF: [Hex + Decimal + Octal + Binary + Formatted + Activity + Preferred_Data + Idle
+ Logging]

Data Type: Mode_Type

3.1.1.6 Output_Formatter to Input_Parser

Screen_Id =

Column + Line

Line =

Data Type: Integer

Data Representation: numeric

Data Range: (1, 24)

Default Value: 1

Column =

Data Type: Integer

Data Representation: numeric

Data Range: (1, 80)

Default Value: 1

3.1.1.7 Output_Formatter to Activity_Monitor

Bus_Activity =

BNF: {RT_Status}

RT_Status =

BNF: [Active + Inactive + Unknown + No_Response]

Comments: Most recently observed status of a given RT

Data Type: An_RT_Status

Data Representation: enumeration

3.1.2 System States and Modes

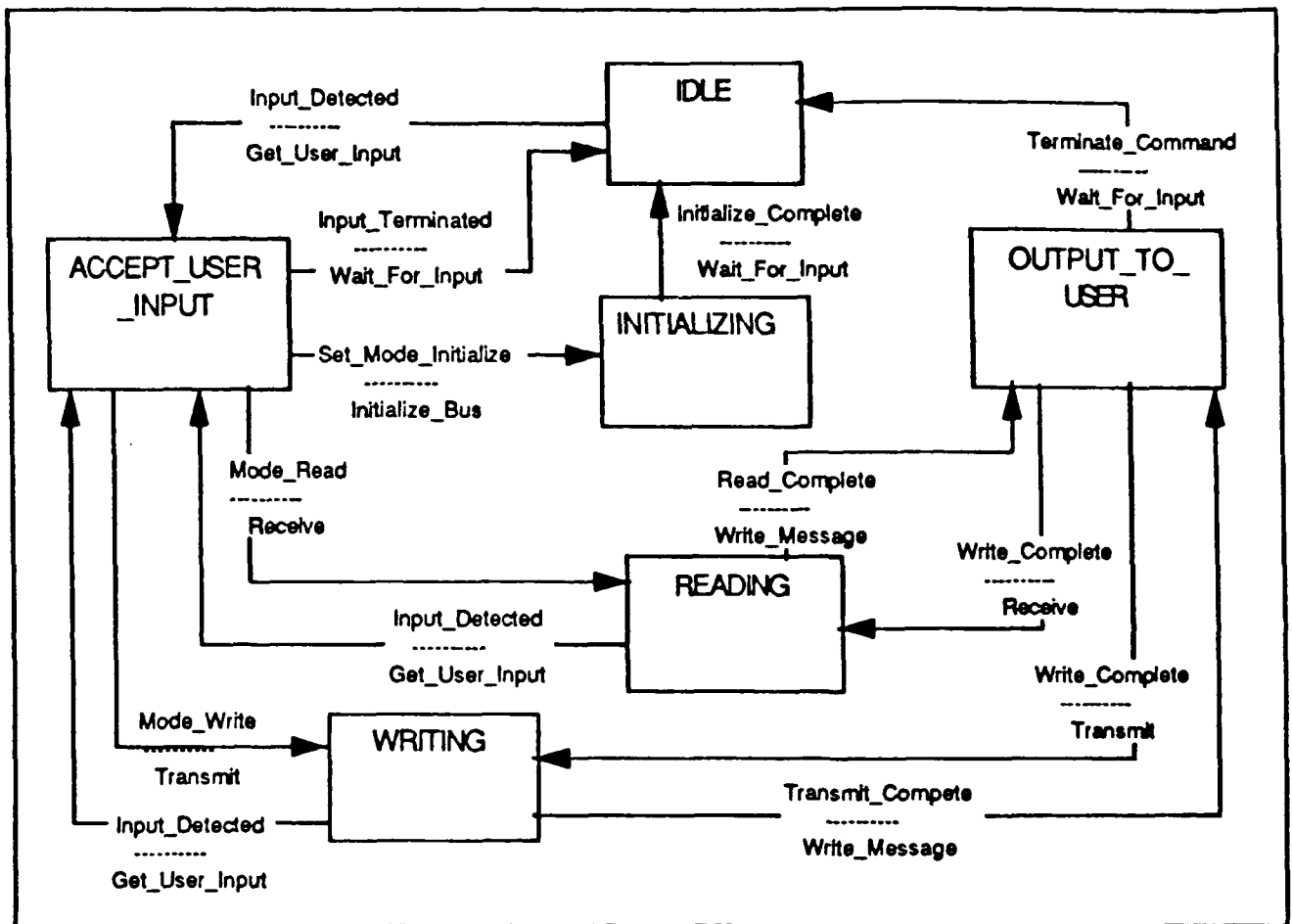


Figure 4 Monitor Application State Transition Diagram

STATE	EVENT	ACTION	NEXT STATE
ACCEPT_USER_INPUT	Input_Terminated	Wait_For_Input	IDLE
	Mode_Write	Transmit	WRITING
	Mode_Read	Receive	READING
	Set_Mode_Initialize	Initialize_Bus	INITIALIZING
IDLE	Input_Detected	Get_User_Input	ACCEPT_USER_INPUT
INITIALIZING	Initialize_Complete	Wait_For_Input	IDLE
OUTPUT_TO_USER	Terminate_Command	Wait_For_Input	IDLE
	Write_Complete	Receive	READING
	Write_Complete	Transmit	WRITING
READING	Read_Complete	Write_Message	OUTPUT_TO_USER
	Input_Detected	Get_User_Input	ACCEPT_USER_INPUT
WRITING	Transmit_Complete	Write_Message	OUTPUT_TO_USER
	Input_Detected	Get_User_Input	ACCEPT_USER_INPUT

Table 1 Monitor Application State Transition Table

CSC \ State	IDLE	ACCEPT_USER INPUT	INITIALIZING	READING	WRITING	OUTPUT_TO USER
Input_Parser						
Output_Formater						
Activity_Monitor						
Bus_Interface						
Monitor_Executive						

Table 2 Monitor Application State / CSC Table

For information on data flow between the CSCs while operation in the given states see figure 3.

3.1.3 Memory and Processing Time Allocation

Not Applicable.

3.2 CSCI Design Description

The following diagrams detail the preliminary design of the CSCs identified in figure 3.

3.2.1 Input_Parser

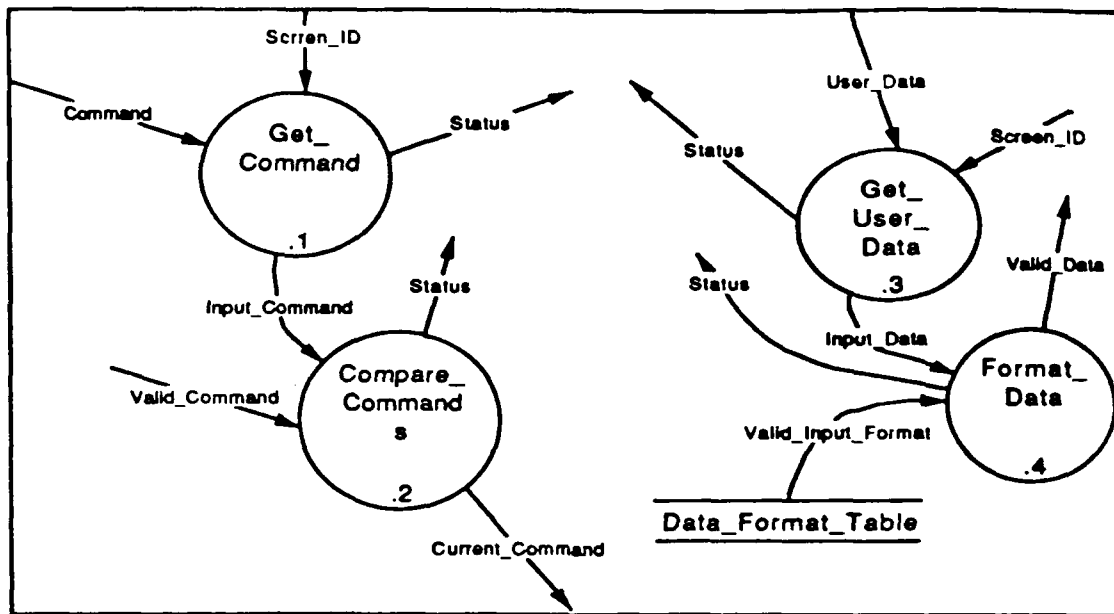


Figure 5 Input_Parser Data Flow Diagram

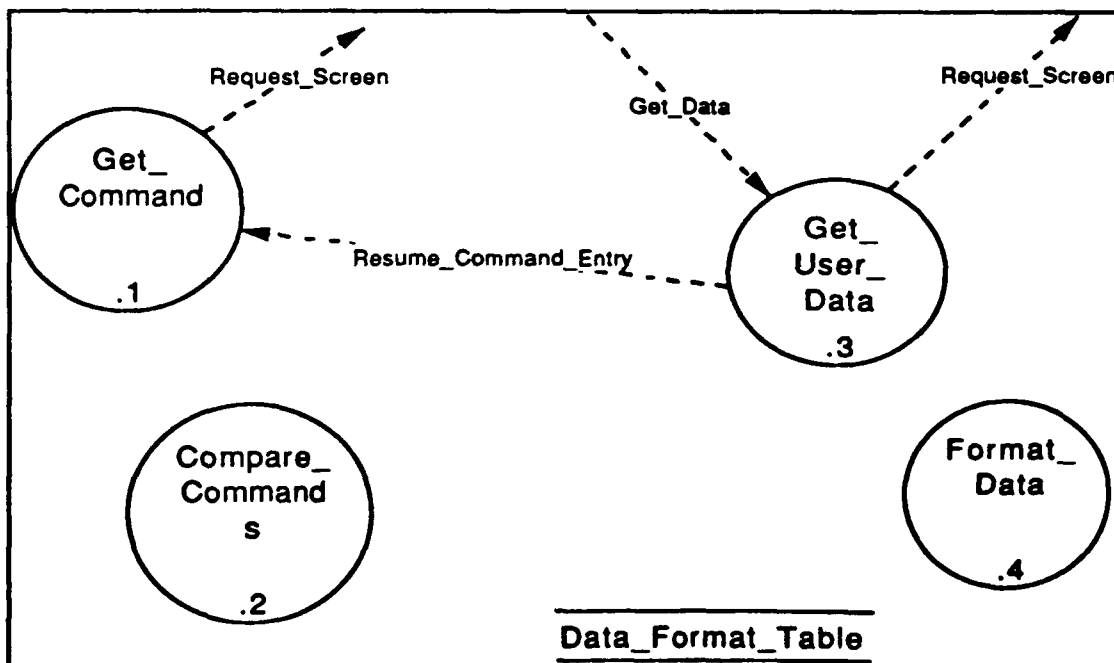


Figure 6 Input_Parser Control Flow Diagram

The input parser CSC meets the requirements identified as the user command interface in the SRS for the Generic Avionics Data Bus Tool Kit. These requirements are identified specifically in figure #2 and paragraph 3.8.1.

3.2.2 Output_Formatter

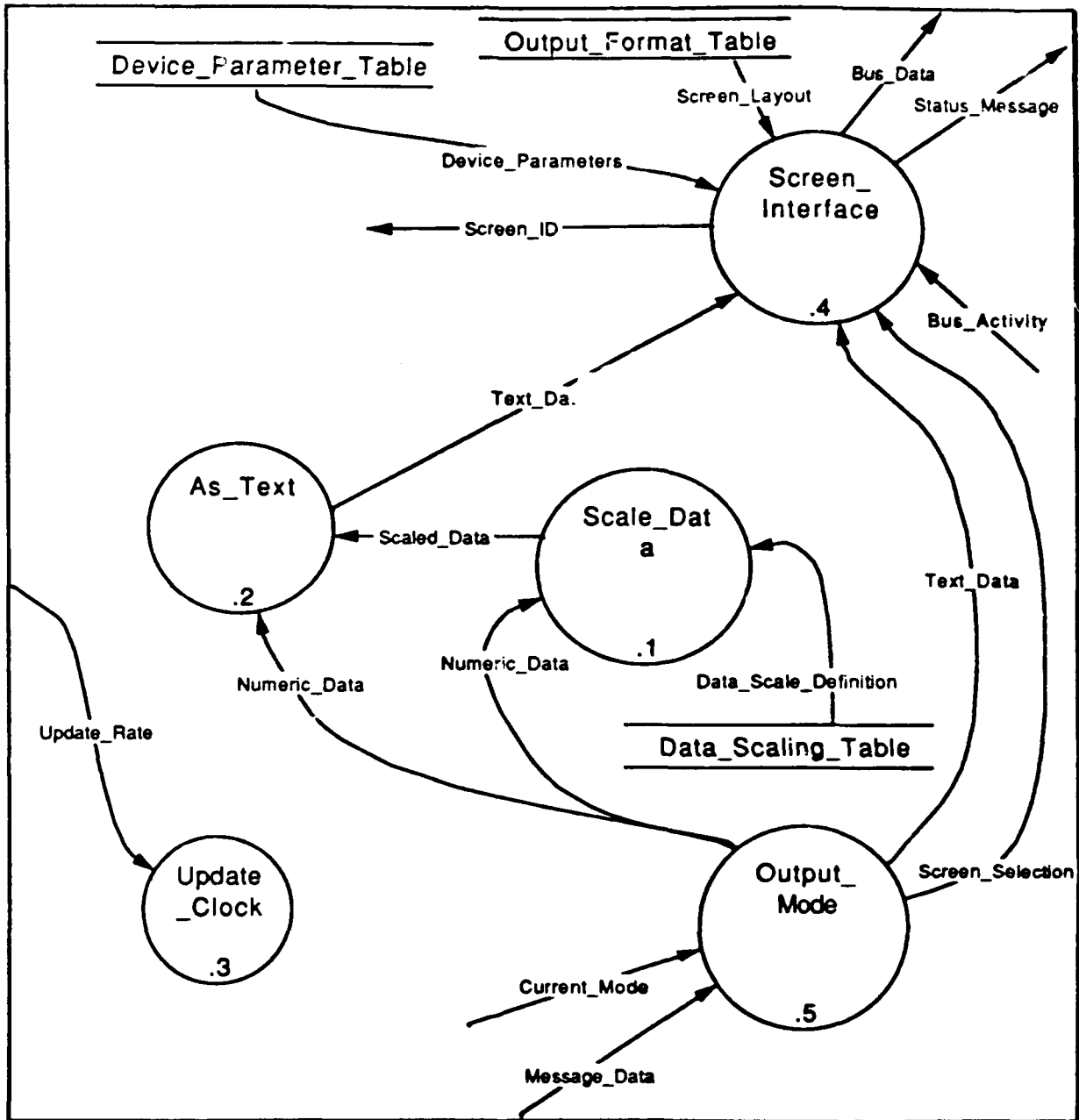


Figure 7 Output_Formatter Data Flow Diagram

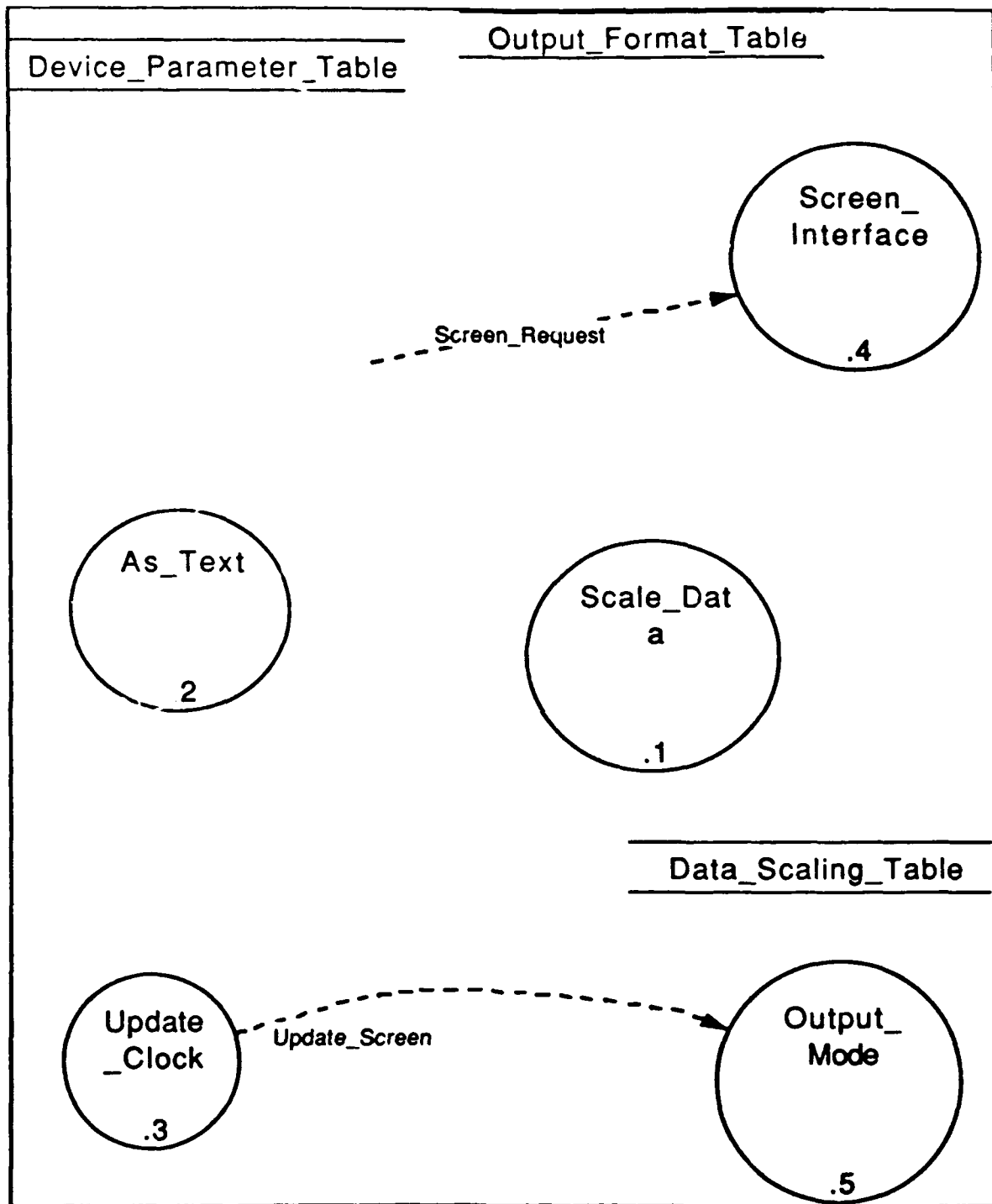


Figure 8 Output_Formatter Control Flow Diagram

The Output Formatter meets the requirements for the user output interface identified in figure #2 and paragraphs 3.2.2.2 and 3.8.4 of the SRS for the Generic Avionics Data Bus Tool Kit.

3.2.3 Activity_Monitor

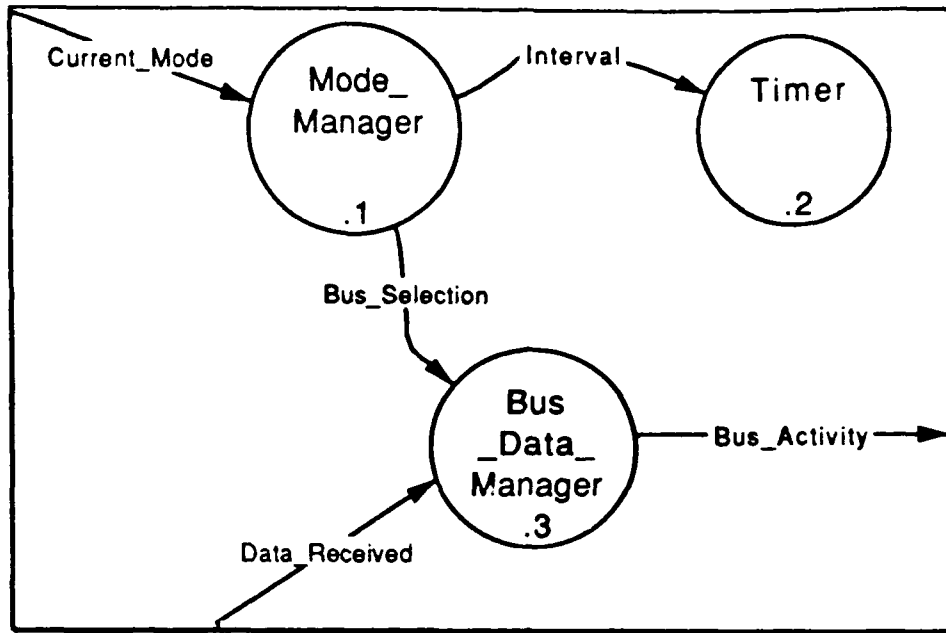


Figure 9 Activity_Monitor Data Flow Diagram

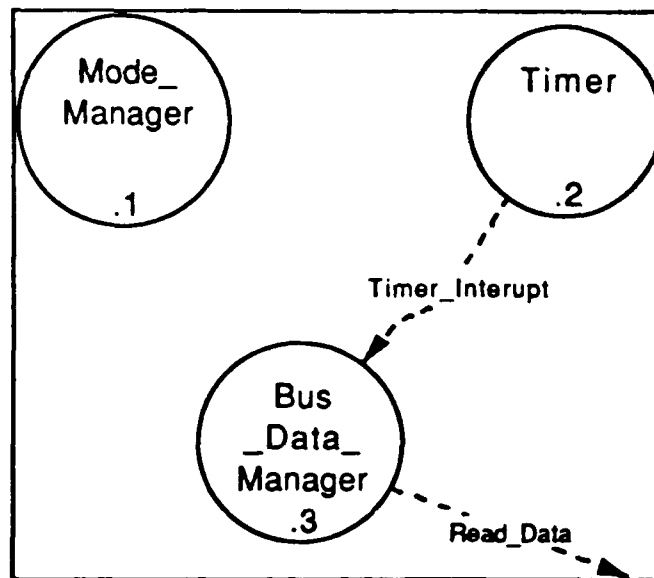


Figure 10 Activity_Monitor Control Flow Diagram

The Activity Monitor will partially fulfil the requirements of paragraph 3.2.2.2 of the SRS for the Generic Avionics Data Bus Tool Kit.

3.2.4 Bus_Interface

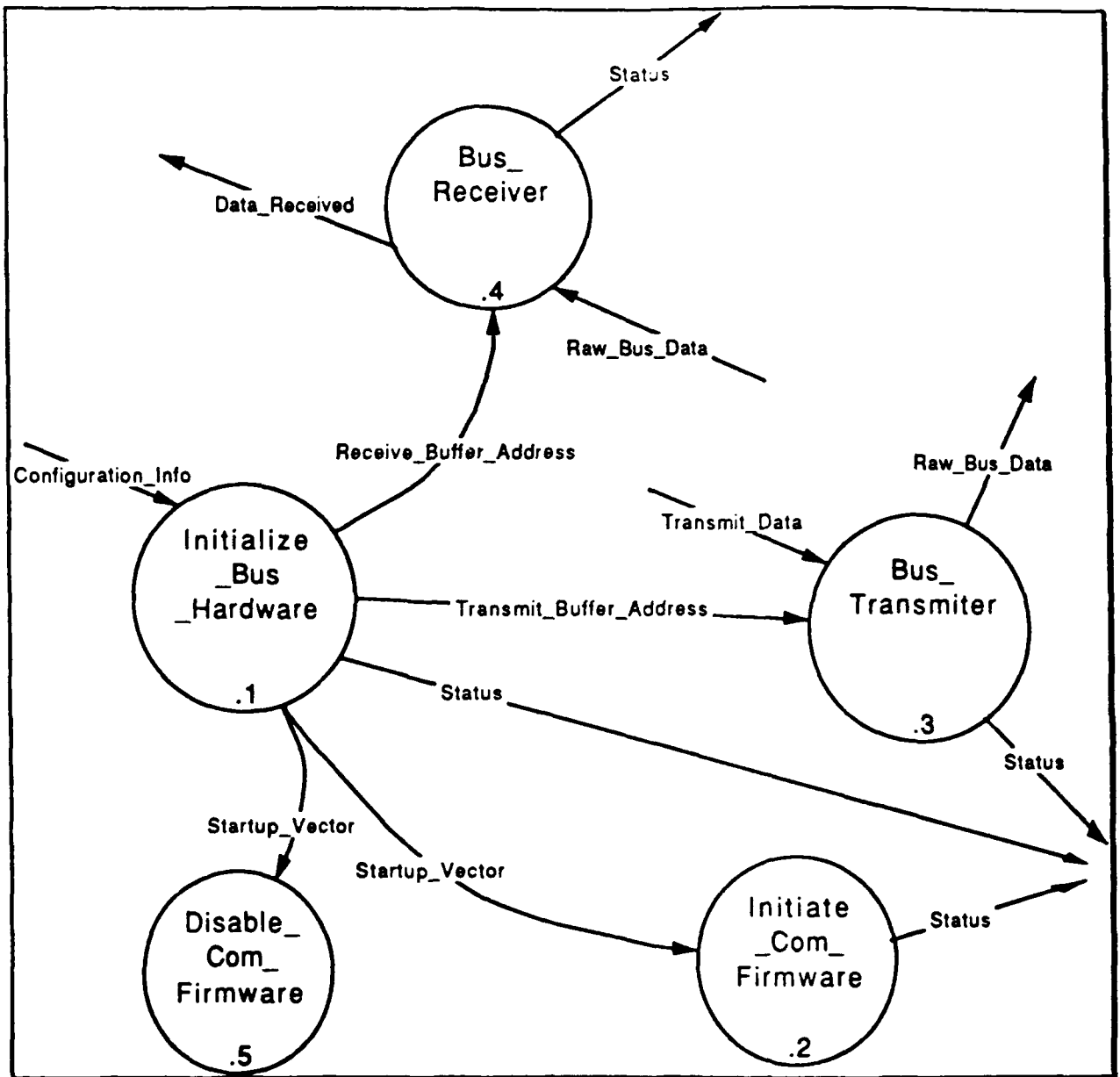


Figure 11 Bus_Interface Data Flow Diagram

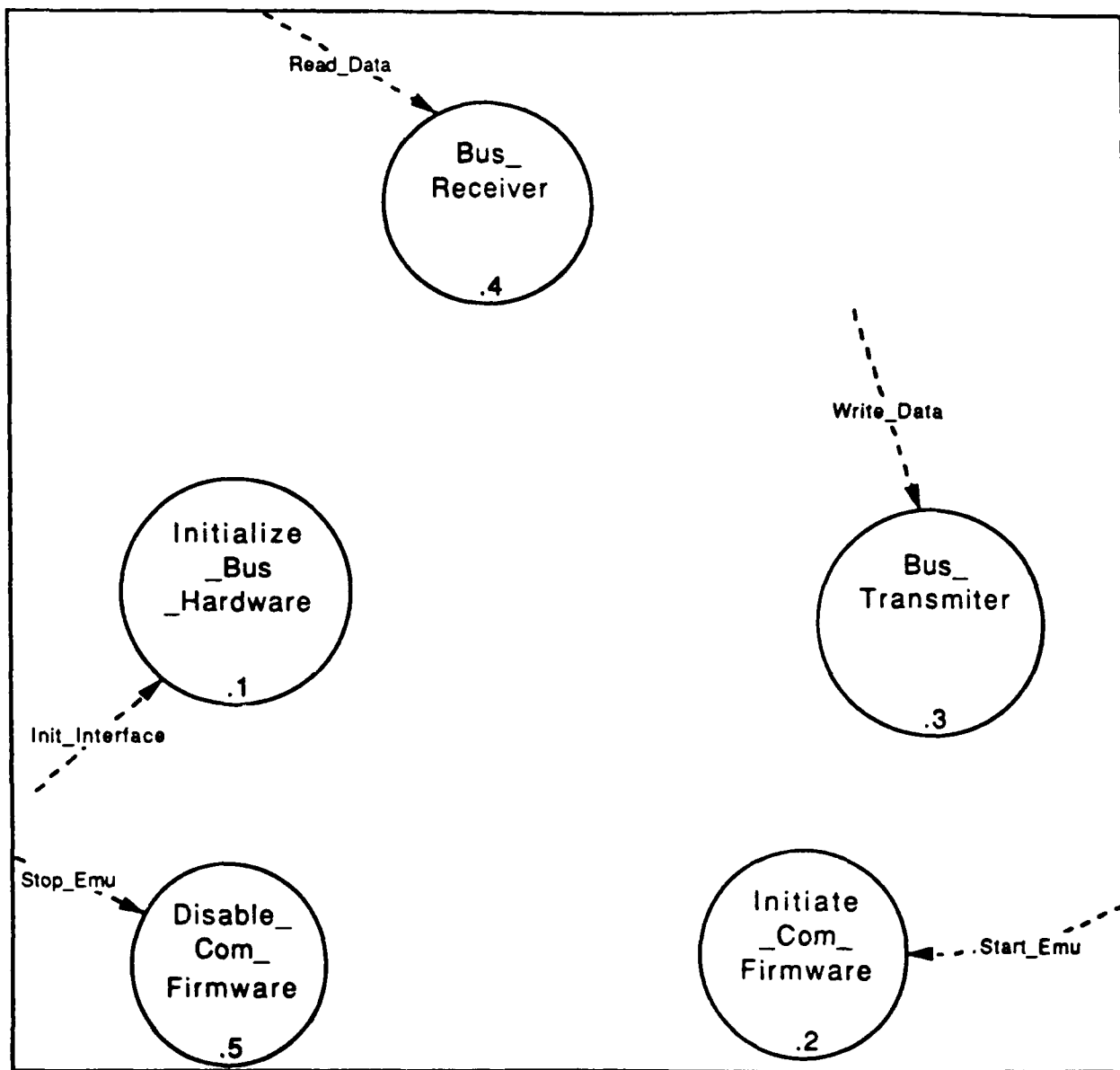


Figure 12 Bus_Interface Control Flow Diagram

The Bus Interface fulfills the requirements identified in 3.2.1 and the associated subparagraphs of the SRS for the Generic Avionics Data Bus Tool Kit.

3.2.5 Monitor_Executive

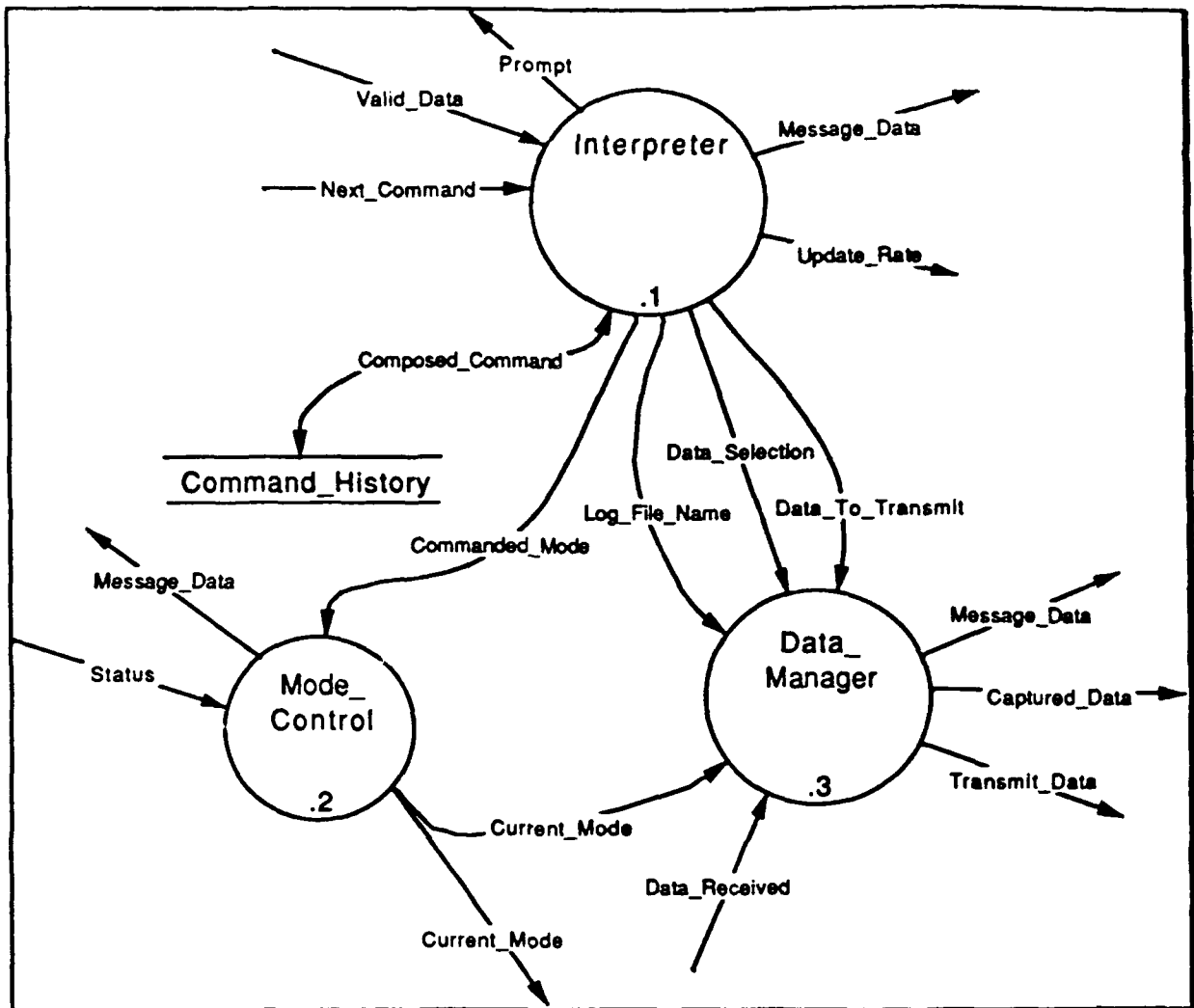


Figure 13 Monitor_Executive Data Flow Diagram

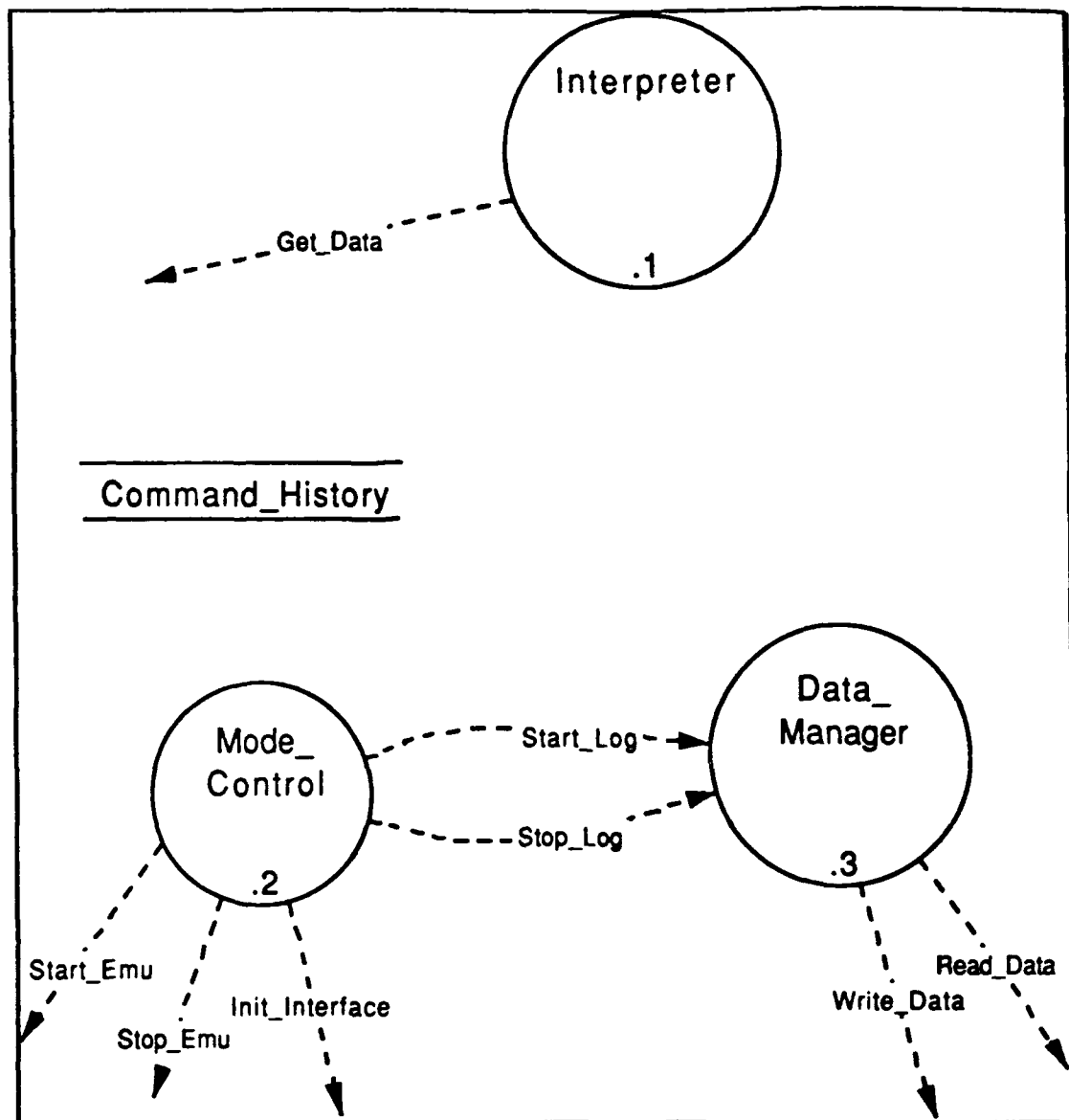


Figure 14 Monitor_Executive Control Flow Diagram

The Monitor Executive meets the requirements derived from paragraph 3.2.2 of the SRS for the Generic Avionics Data Bus Tool Kit. The Monitor Executive is not a direct requirement of the referenced document. Its requirement was derived from the need to coordinate the transfer of data and control information between the other CSC units.

4 DETAILED DESIGN

The following paragraphs present the detailed design of the CSCI identified as the Generic Avionics Data Bus Tool Kit. The format followed will be to present the Ada entity diagrams for each CSC then any Ada entity diagrams necessary for each CSU finally the Ada specification for the unit and a textual description will be presented. For more Ada structure charts see appendix B.

4.1 DD CSC Input Parser

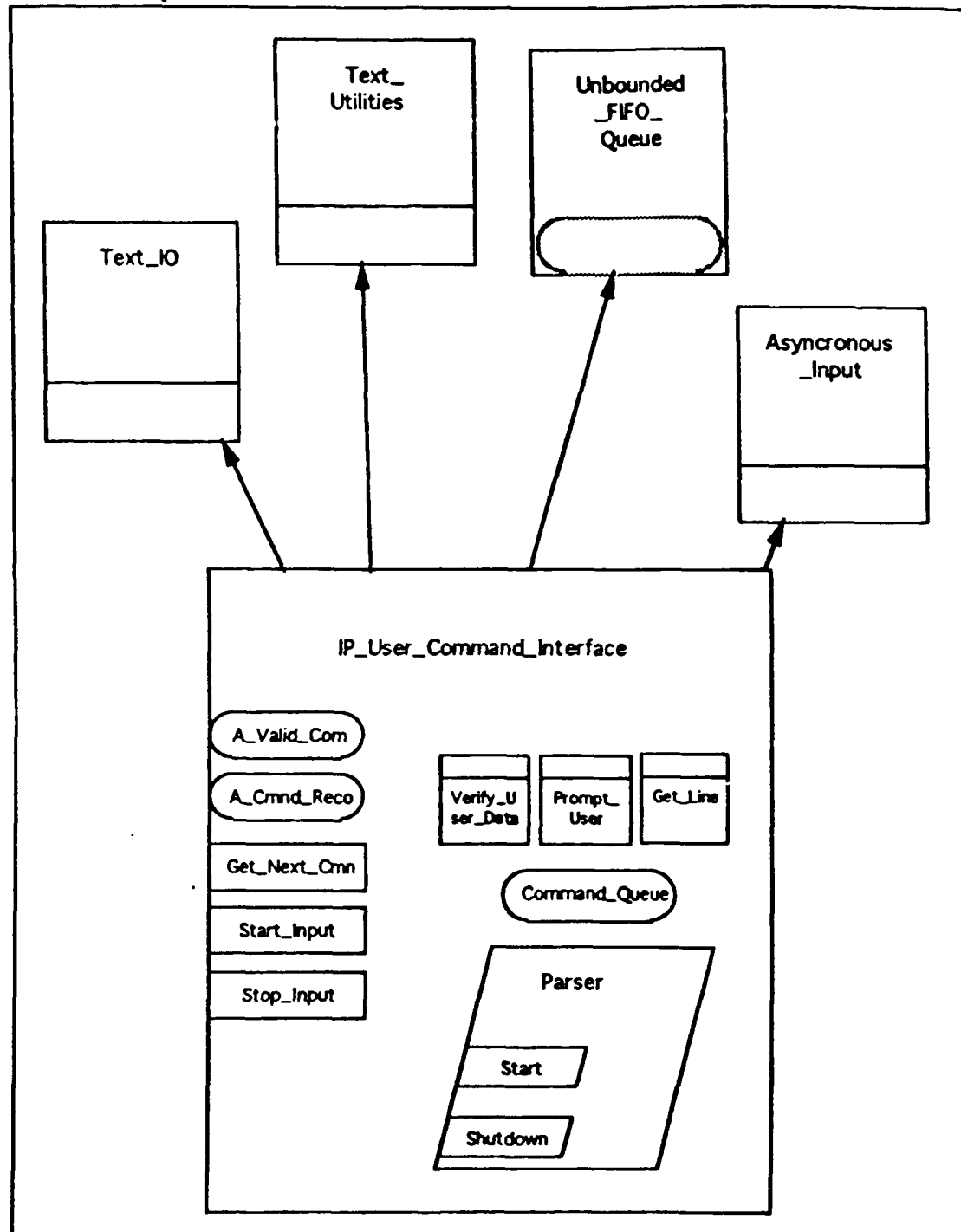


Figure 15 Input Parser Ada Structure Diagram 23


```
package IP_USER_COMMAND_INTERFACE is
```

```
type A_VALID_COMMAND is ( SELECT_RT,      SELECT_SA,      SELECT_TRANS,      SELECT_RCV,
                           SELECT_BUS,    MODE_HEX,        MODE_DECIMAL,
                           MODE_BINARY,    MODE_OCTAL,        MODE_FORMATTED,
                           MODE_ACTIVITY,  MODE_STATUS,      MODE_PREFERRED_DATA,
                           MODE_LOGGING,   MODE_RAW_DATA,    START_LOGGING,      STOP_LOGGING,
                           SET_LOG_NAME,   SET_SNAP_LEN,    CANCEL_SELECTION,
                           RESET_DISP,    EXECUTE,         QUIT,                NONE);
```

```
subtype A_COMMAND_DATA is string (1..80);
```

```
type A_VALID_COMMAND_RECORD is
```

```
record
```

```
    Command      :A_VALID_COMMAND;
```

```
    Command_Data :A_COMMAND_DATA;
```

```
    Data_Length  :INTEGER;
```

```
end record;
```

```
procedure Start_Input;
```

```
procedure Get_Next_Command (COMMAND: out A_VALID_COMMAND_RECORD);
```

```
procedure Shutdown_Input;
```

```
end IP_USER_COMMAND_INTERFACE;
```

4.1.1 IP_User_Command_Interface.Start_Input

This control interface socket is used to start the input parser processor. All local data to the package body is initialized and asynchronous input is enabled.

4.1.1.1 Start_Input Design Specification Constraints.

The input parser depends on the output formatter so if the output formatter is not running when this call is made then input processing will block until the output formatter is started.

4.1.1.2 Start_Input Design

a. Inputs: n.a.

b. Outputs: n.a.

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: n.a.

f. Error Handling: none.

g. Used Elements: Task Parser entry Start_Parser : declared in local package body.

4.1.2 IP_User_Command_Interface.Shutdown_Input

This procedure will allow the task Parser to end in an orderly manner.

4.1.2.1 Shutdown_Input Design Specification Constraints.

n.a.

4.1.2.2 Shutdown_Input Design

a. Inputs: n.a.

b. Outputs: n.a.

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: n.a.

f. Error Handling: TASKING_ERROR : If tasking error is raised a diagnostic message is directed to the output screen. The error is not propagated since it most likely is raised if the Parser task has already exited.

g. Used Elements: Task Parser entry Kill_Parser : declared in local package body.

4.1.3 IP_User_Command_Interface.Get_Next_Command

This procedure returns the next command the parser has read from the user.

4.1.3.1 Get_Next_Command Design Specification Constraints.

n.a.

4.1.3.2 Get_Next_Command Design

a. Inputs: n.a.

b. Outputs: Command : A_VALID_COMMAND_RECORD

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: If que empty return null command else return command.

f. Error Handling: none.

g. Used Elements: Function Que_Empty : declared in Command_Ques.
Function Next_Entry : declared in Command_Ques.

4.2 DD CSC Output_Formatter

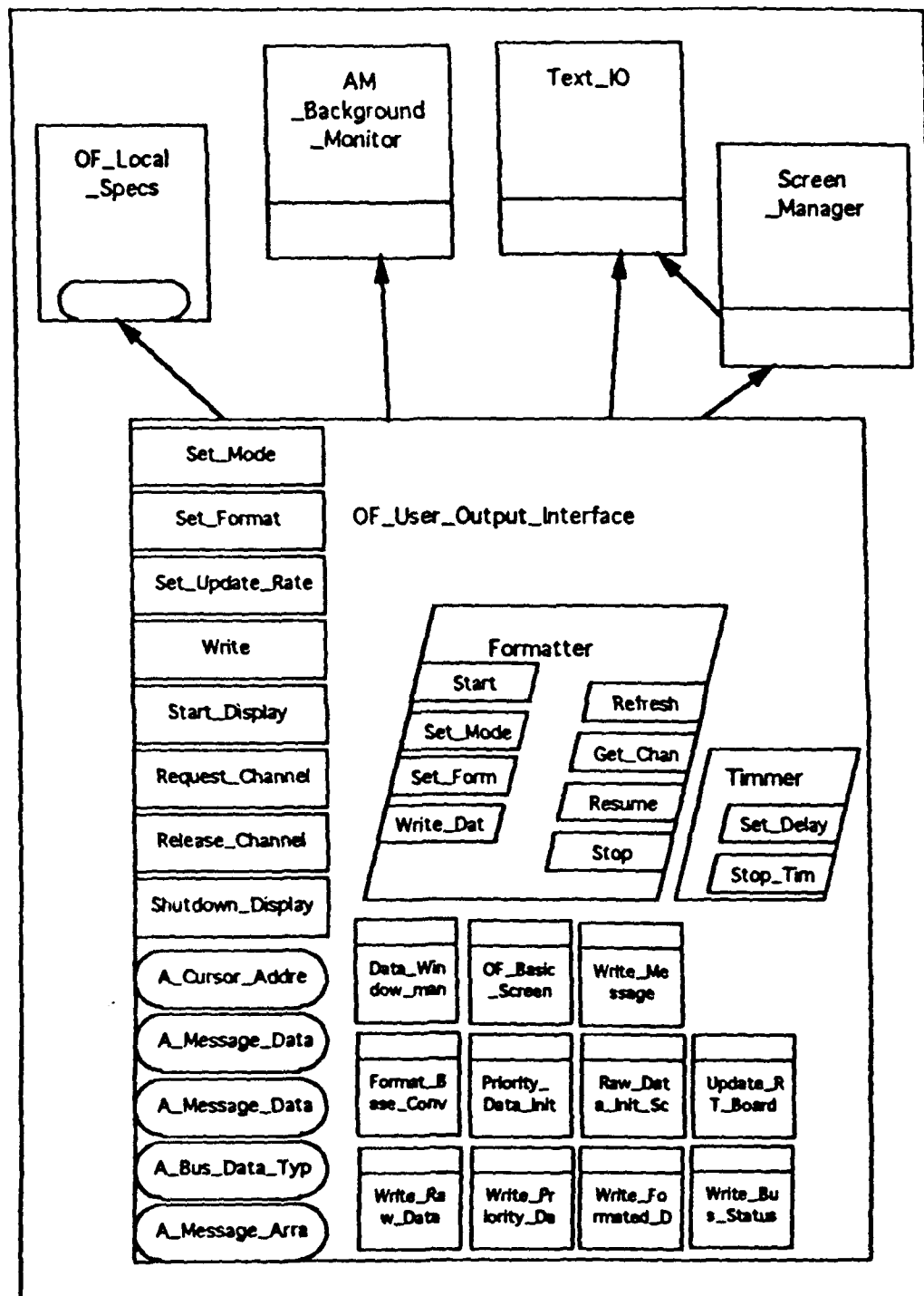


Figure 16 Output Formatter Ada Structure Diagram

```

with TD_MODE_DEFINITIONS;      use TD_MODE_DEFINITIONS;
with BI_1553_WORD_DEFINITIONS; use BI_1553_WORD_DEFINITIONS;
with TD_TIME_TYPES;           use TD_TIME_TYPES;
with UT_VARIABLE_STRING;      use UT_VARIABLE_STRING;

package OF_USER_OUTPUT_INTERFACE is

    subtype MESSAGE_TEXT_TYPE is TEXT_TYPE(512);
    subtype ERROR_TEXT_TYPE is TEXT_TYPE(80);

    type A_MESSAGE_ARRAY is array (A_WORD_COUNT) of A_DATA_WORD;

    type BUS_DATA_TYPE is
        record
            COMMAND      : A_DATA_COMMAND_WORD;
            BUS_NUMBER   : INTEGER;
            UNIT_NUMBER  : INTEGER;
            RESPONSE     : A_STATUS_WORD;
            BUS_DATA     : A_MESSAGE_ARRAY;
            ACTIVE       : BOOLEAN;
        end record;

    type A_DATA_MESSAGE_TYPE is (TEXT, BUS_DATA, ERROR);

    type A_MESSAGE_DATA_BLOCK (MESSAGE_TYPE : A_DATA_MESSAGE_TYPE) is
        record
            case MESSAGE_TYPE is
                when TEXT =>
                    MESSAGE : MESSAGE_TEXT_TYPE;
                when ERROR =>
                    ERROR_DATA : ERROR_TEXT_TYPE;
                when BUS_DATA =>
                    BUS_DATA : BUS_DATA_TYPE;
            end case;
        end record;

    type A_CURSOR_ADDRESS is
        record
            Column : POSITIVE;
            Line   : POSITIVE;
        end record;

```

```

procedure Set_Mode (Monitor_Mode : in A_MONITOR_MODE);

procedure Set_Format (Format_Mode : in A_FORMAT_MODE);

procedure Set_Update_Rate (Update_Interval : in SECONDS);

procedure Write (Message : in A_MESSAGE_DATA_BLOCK);

procedure Request_Channel (Cursor : out A_CURSOR_ADDRESS);

procedure Release_Channel;

procedure Clear_Display;

procedure Start_Display;

procedure Shutdown_Display;

end OF_USER_OUTPUT_INTERFACE;

```

4.2.1 OF_User_Output_Interface.Set_Mode

Place the output formatter in the specified operational mode.

4.2.1.1 Set_Mode Design Specification Constraints.

n.a.

4.2.1.2 Set_Mode Design

a. Inputs: Monitor_Mode : A_MONITOR_MODE

b. Outputs: n.a.

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: n.a.

c. Local Data: Rate_Change : BOOLEAN

d. Signals: n.a.

e. Algorithms: n.a.

f. Error Handling: none.

g. Used Elements: Task Timer entry Set_Delay : declared in local package body.

4.2.4 OF_User_Output_Interface.Write

This procedure causes the data passed through the interface to be displayed on the data portion of the output screen.

4.2.4.1 Write Design Specification Constraints.

If the time needed to display the data written is greater than the time till the next call to this procedure there is no guarantee that the data will be displayed before the new data is displayed.

4.2.4.2 Write Design

a. Inputs: Message : A_MESSAGE_DATA_BLOCK

b. Outputs: n.a.

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: n.a.

f. Error Handling: none.

g. Used Elements: Task Formatter entry Write_Data : declared in local package body.

4.2.5 OF_User_Output_Interface.Request_Channel

When this procedure is called output processing is stopped and the coordinates of the input buffer area of the screen are returned to the caller. Output remains suspended until the

f. Error Handling: none.

g. Used Elements: Task Formatter entry Set_Mode : declared in local package body.

4.2.2 OF_User_Output_Interface.Set_Format

Define the data format the output will be displayed in by the output formatter task.

4.2.2.1 Set_Format Design Specification Constraints.

n.a.

4.2.2.2 Set_Format Design

a. Inputs: Format_Mode : A_FORMAT_MODE

b. Outputs: n.a.

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: n.a.

f. Error Handling: none.

g. Used Elements: Task Formatter entry Set_Format : declared in local package body.

4.2.3 OF_User_Output_Interface.Set_Update_Rate

This interface allows the rate for update of the RT status map portion of the display to be set.

4.2.3.1 Set_Update_Rate Design Specification Constraints.

Default rate will be set at 3 seconds.

4.2.3.2 Set_Update_Rate Design

a. Inputs: Update_Interval : SECONDS

b. Outputs: n.a.

Release_Channel interface is called.

4.2.5.1 Request_Channel Design Specification Constraints.

n.a.

4.2.5.2 Request_Channel Design

a. Inputs: n.a.

b. Outputs: Cursor : A_CURSOR_ADDRESS

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: n.a.

f. Error Handling: none.

g. Used Elements: Task Formatter entry Get_Channel : declared in local package body.

4.2.6 OF_User_Output_Interface.Release_Channel

This procedure causes the display processor to resume screen output after a call to Request_Channel has suspended output.

4.2.6.1 Release_Channel Design Specification Constraints.

n.a.

4.2.6.2 Release_Channel Design

a. Inputs: n.a.

b. Outputs: n.a.

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: n.a.

f. Error Handling: none.

g. Used Elements: Task Formatter entry Resume : declared in local package body.

4.2.7 OF_User_Output_Interface.Clear_Display

This procedure forces the display processor to reinitialize the display. This means the display is cleared and the background for the current mode and format are repainted. Display of data will resume the next time new data is written to the formatter.

4.2.7.1 Clear_Display Design Specification Constraints.

The data on the display at the time the call is made will not be redisplayed unless the same data is placed in a subsequent write operation.

4.2.7.2 Clear_Display Design

a. Inputs: n.a.

b. Outputs: n.a.

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: n.a.

f. Error Handling: none.

g. Used Elements: Task Formatter entry Initialize_Display : declared in local package body.

4.2.8 OF_User_Output_Interface.Start_Display

This procedure enables screen output from the display processor.

4.2.8.1 Start_Display Design Specification Constraints.

n.a.

4.2.8.2 Start_Display Design

a. Inputs: n.a.

b. Outputs: n.a.

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: Start Background_Monitor
Enable Display_Processor output

f. Error Handling: none.

g. Used Elements: Procedure Turn_Background_Monitor_On : declared in package
AM_Background_Monitor.
Task Formatter entry Start : declared in local package body.

4.3 DD CSC Activity_Monitor

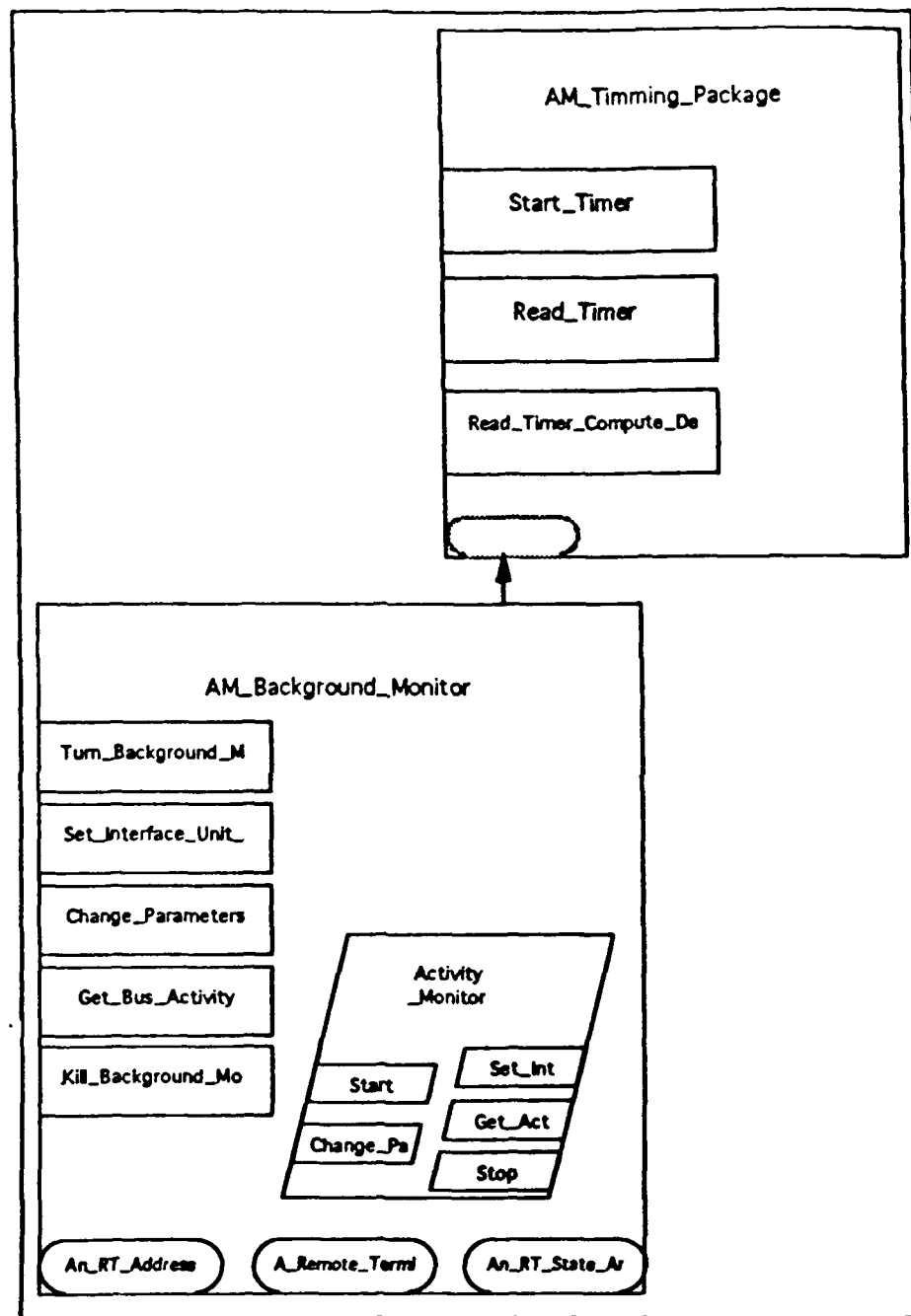


Figure 17 Activity Monitor Ada Structure Diagram

```

with BI_1553_WORD_DEFINITIONS;
with TD_TIME_TYPES;          use TD_TIME_TYPES;
with Interface_1553;         use Interface_1553;

package AM_BACKGROUND_MONITOR is

    package WDEF renames BI_1553_WORD_DEFINITIONS;

    subtype AN_RT_ADDRESS is WDEF.A_REMOTE_TERMINAL_ADDRESS range
        WDEF.A_REMOTE_TERMINAL_ADDRESS'first..
        WDEF.A_REMOTE_TERMINAL_ADDRESS'pred(
        WDEF.A_REMOTE_TERMINAL_ADDRESS'last);

    type A_REMOTE_TERMINAL_STATE is (ACTIVE_NOW, -- RT responding.
        ACTIVE,      -- RT was active, but ceased.
        MC_ONLY,     -- MC commanding, no response.
        INACTIVE);  -- No activity observed.

    type AN_RT_STATE_ARRAY is array (AN_RT_ADDRESS) of A_REMOTE_TERMINAL_STATE;

    procedure Turn_Background_Monitor_ON (History_Length : in SECONDS := 120;
        Sample_Rate      : in SAMPLES_PER_SECOND := 10);

    procedure Set_Interface_Unit_Number (Unit_Number : in A_Interface_Unit);

    procedure Turn_Background_Monitor_OFF;

    procedure Change_BM_Parameters (History_Length : in SECONDS := 120;
        Sample_Rate      : in SAMPLES_PER_SECOND := 10);

    procedure Get_Bus_Activity (Activity_Record : out AN_RT_STATE_ARRAY);

    procedure Kill_Background_Monitor_Task;

end AM_BACKGROUND_MONITOR;

```

4.3.1 AM_Background_Monitor.Turn_Background_Monitor_On

This procedure will initialize the timers and starts the task to keep the RT activity Map data.

4.3.1.1 Turn_Background_Monitor_On Design Specification Constraints.

n.a.

4.3.1.2 Turn_Background_Monitor_On Design

- a. Inputs: History_Length : SECONDS
 Sample_Rate : SAMPLES_PER_SECOND
- b. Outputs: n.a.
- c. Local Data: Past_Duration : SECONDS
 The_Sample_Rate : SAMPLES_PER_SECOND
- d. Signals: n.a.
- e. Algorithms: n.a.
- f. Error Handling: none.
- g. Used Elements: Task Activity_Monitor entry Turn_On : declared in local package body.

4.3.2 AM_Background_Monitor.Set_Interface_Unit_Number

This procedure selects the CTSD MMBI interface unit that will be monitored.

4.3.2.1 Set_Interface_Unit_Number Design Specification Constraints.

n.a.

4.3.2.2 Set_Interface_Unit_Number Design

- a. Inputs: Unit_Number : A_UNIT_NUMBER
- b. Outputs: n.a.
- c. Local Data: The_1553_Interface_Unit : A_UNIT_NUMBER
- d. Signals: n.a.
- e. Algorithms: n.a.

f. Error Handling: none.

g. Used Elements: n.a.

4.3.3 AM_Background_Monitor.Change_BM_Parameters

This procedure allows the operational parameters to be modified dynamically during execution.

4.3.3.1 Change_BM_Parameters Design Specification Constraints.

n.a.

4.3.3.2 Change_BM_Parameters Design

a. Inputs: History_Length : SECONDS
Sample_Rate : SAMPLES_PER_SECOND

b. Outputs: n.a.

c. Local Data: Past_Duration : SECONDS
The_Sample_Rate : SAMPLES_PER_SECOND

d. Signals: n.a.

e. Algorithms: n.a.

f. Error Handling: none.

g. Used Elements: Task Activity_Monitor entry Change_Settings : declared in local package body.

4.3.4 AM_Background_Monitor.Get_Bus_Activity

Interface to take a snapshot of the current Bus Activity Map.

4.3.4.1 Get_Bus_Activity Design Specification Constraints.

n.a.

4.3.4.2 Get_Bus_Activity Design

a. Inputs: n.a.

b. **Outputs:** Activity_Record : AN_RT_STATE_ARRAY

c. **Local Data:** Current_RT_State_Table : AN_RT_STATE_ARRAY

d. **Signals:** n.a.

e. **Algorithms:** n.a.

f. **Error Handling:** none.

g. **Used Elements:** Task Activity_Monitor entry Report_Activity : declared in local package body.

4.3.5 AM_Background_Monitor.Turn_Background_Monitor_Off

Suspend background monitor.

4.3.5.1 Turn_Background_Monitor_Off Design Specification Constraints.

n.a.

4.3.5.2 Turn_Background_Monitor_Off Design

a. **Inputs:** n.a.

b. **Outputs:** n.a.

c. **Local Data:** n.a.

d. **Signals:** n.a.

e. **Algorithms:** n.a.

f. **Error Handling:** none.

g. **Used Elements:** Task Activity_Monitor entry Turn_Off : declared in local package body.

4.3.6 AM_Background_Monitor.Kill_Background_Monitor_Task

4.3.6.1 Kill_Background_Monitor_Task Design Specification Constraints.

4.3.6.2 Kill_Background_Monitor_Task Design

- a. Inputs:** n.a.
- b. Outputs:** n.a.
- c. Local Data:** n.a.
- d. Signals:** n.a.
- e. Algorithms:** n.a.
- f. Error Handling:** none.
- g. Used Elements:** Task Activity_Monitor entry Kill : declared in local package body.

4.4 DD CSC Bus_Interface

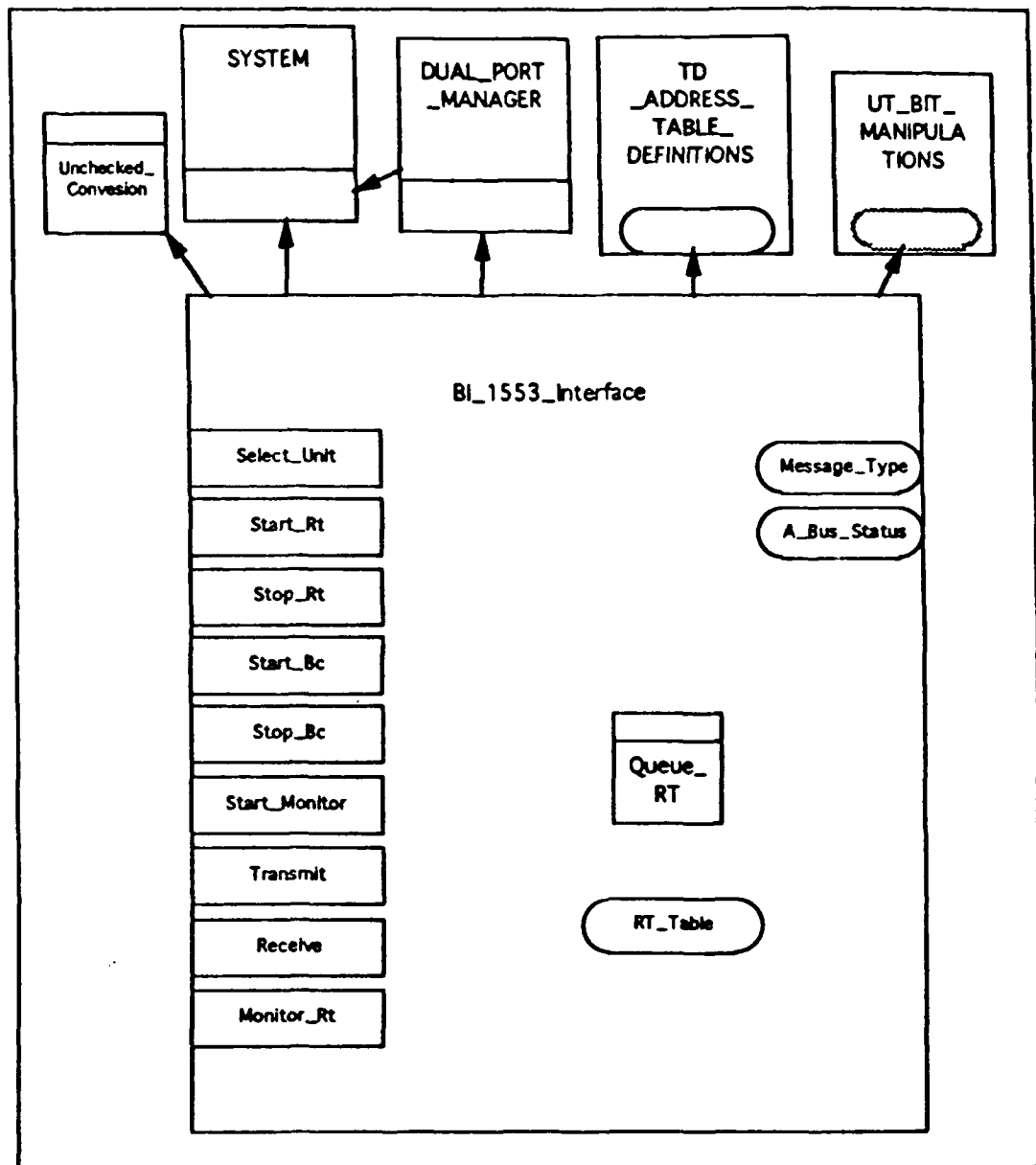


Figure 18 Bus Interface Ada Structure Diagram

```

with BI_1553_WORD_DEFINITIONS;      use BI_1553_WORD_DEFINITIONS;

package BI_1553_INTERFACE is

    BUS_DATA_FORMAT_ERROR      : exception;
    HARDWARE_FAILURE           : exception;
    KERNAL_SYSTEM_ERROR        : exception;
    RANGE_ERROR                 : exception;

    type MESSAGE_TYPE is (COMMAND,RESPONCE);
    type A_BUS_STATUS is (INITIALIZED, NOT_REINITIALIZED,
                          NOT_INITIALIZED, OUT_OF_MEMORY,
                          PARAMETER_ERROR);

    subtype AN_INTERFACE is INTEGER range 0..3;
    subtype A_MFS_BUS is INTEGER range 0..7;
    subtype A_UNIBUS is INTEGER range 0..2;

    subtype A_TIME_TAG is INTEGER range INTEGER'first..INTEGER'last;

    procedure Select_Unit (Interface_Number : AN_INTERFACE := 0;
                           Bus_Number       : A_MFS_BUS    := 0;
                           Unibus_Number    : A_UNIBUS      := 0);

    procedure Start_RT (Remote_Terminal : in    A_REMOTE_TERMINAL_ADDRESS;
                        Status           :      out A_BUS_STATUS);

    procedure Stop_RT (Remote_Terminal : in    A_REMOTE_TERMINAL_ADDRESS);

    procedure Start_BC (Status :      out A_BUS_STATUS);

    procedure Stop_BC;

    procedure Start_Monitor (Remote_Terminal : in    A_REMOTE_TERMINAL_ADDRESS;
                             Status           :      out A_BUS_STATUS);

    procedure Stop_Monitor (Remote_Terminal : in    A_REMOTE_TERMINAL_ADDRESS);

    generic
        type A_TRANSMIT_MESSAGE_BUFFER is limited private;
        type A_BUS_OVERHEAD_WORD is limited private;

```

```

procedure Transmit (Data      : in A_TRANSMIT_MESSAGE_BUFFER;
                   Bus_Word   : in A_BUS_OVERHEAD_WORD;
                   Form       : in MESSAGE_TYPE;
                   Subaddress  : in A_SUBADDRESS := 0);

generic
  type A_RECEIVE_MESSAGE_BUFFER is private;
  type A_BUS_OVERHEAD_WORD is private;
procedure Receive (Data      : out A_RECEIVE_MESSAGE_BUFFER;
                  Bus_Word   : out A_BUS_OVERHEAD_WORD;
                  Time       : out A_TIME_TAG;
                  Form       : in MESSAGE_TYPE;
                  RT_Number  : in A_REMOTE_TERMINAL_ADDRESS;
                  Subaddress : in A_SUBADDRESS);

generic
  type A_MONITOR_MESSAGE_BUFFER is private;
procedure Monitor_RT (Command_Word : out A_DATA_COMMAND_WORD;
                    Data           : out A_MONITOR_MESSAGE_BUFFER;
                    Status_Word    : out A_STATUS_WORD;
                    Time           : out A_TIME_TAG;
                    Form           : in MESSAGE_TYPE;
                    RT_Number      : in A_REMOTE_TERMINAL_ADDRESS;
                    Subaddress     : in A_SUBADDRESS);

end BI_1553_INTERFACE;

```

4.4.1 BI_1553_Interface.Select_Unit

This procedure is used in an environment where multiple 1553 hardware interfaces exist in parallel to select the specific unit to operate on. The bus to default to is also selected with this procedure. Additionally for use on the CTSD MMBI interface adaptor the VAX Unibus number is needed to uniquely identify the unit.

4.4.1.1 Select_Unit Design Specification Constraints.

n.a.

4.4.1.2 Select_Unit Design

a. Inputs: Interface_Number : AN_INTERFACE
 Bus_Number : A_MFS_BUS

Unibus_Number : A_UNIBUS

- b. Outputs: n.a.
- c. Local Data: Interface : A_INTERFACE_UNIT
RT_Table : AN_INTERFACE_TABLE
- d. Signals: n.a.
- e. Algorithms: if input parameters are in valid range assign to local copies else raise RANGE_ERROR.
- f. Error Handling: none.
- g. Used Elements: n.a.

4.4.2 BI_1553_INTERFACE.Start_RT

This procedure allocates memory, will initialize data structures, and activates the interface hardware for the specified RT.

4.4.2.1 Start_RT Design Specification Constraints.

n.a.

4.4.2.2 Start_RT Design

- a. Inputs: Remote_Terminal : A_REMOTE_TERMINAL_ADDRESS
- b. Outputs: Status : A_BUS_STATUS
- c. Local Data: RT : A_REMOTE_TERMINAL_ADDRESS
Stat : A_BUS_STATUS
DP_Unibus_Address : NATURAL
Data_Block_Size : INTEGER
- d. Signals: n.a.
- e. Algorithms: Verify interface usability.

Create a data pointer offset table.
Allocate memory.
Initialize memory.
Start 1553 processor.

f. Error Handling: none.

g. Used Elements: Procedure Verify_Interface : declared in local package body.
Procedure Create_Offset_Table : declared in local package body.
Procedure Allocate_Buffers : declared in local package body.
Procedure Queue_RT : declared in local package body.

4.4.3 BI_1553_INTERFACE.Stop_RT

This procedure is used to halt the interface processor and release allocated memory.

4.4.3.1 Stop_RT Design Specification Constraints.

n.a.

4.4.3.2 Stop_RT Design

a. Inputs: Remote_Terminal : A_REMOTE_TERMINAL_ADDRESS

b. Outputs: n.a.

c. Local Data: Tbus_Address : A_TBUS_ADDRESS
Tbus_Status : A_TBUS_STATUS
RT_Offset : A_TBUS_ADDRESS

d. Signals: n.a.

e. Algorithms: Calculate the control register address for the interface.
Write halt code to control register.
Release allocated memory blocks.

f. Error Handling: none.

g. Used Elements: Procedure Write_Tbus : declared in package Interface_1553.
Procedure Release : declared in package Dual_Port_Manager.

4.4.4 BI_1553_INTERFACE.Start_BC

This procedure is used to allocate memory, initialize data structures and start the interface processor in Bus Controller mode.

4.4.4.1 Start_BC Design Specification Constraints.

n.a.

4.4.4.2 Start_BC Design

a. Inputs: n.a.

b. Outputs: Status : A_BUS_STATUS

c. Local Data: BC_Data_Block : A_BUS_CONTROLLER_DESCRIPTOR
Tbus_Address : A_TBUS_ADDRESS
Tbus_Status : A_TBUS_ERROR

d. Signals: n.a.

e. Algorithms: Verify hardware accessibility.
Allocate memory.
Initialize data structures.
Calculate control register address.
Write enable to control register.

f. Error Handling: none.

g. Used Elements: Procedure Read_Tbus : declared in package Interface_1553.
Procedure Write_Tbus : declared in package Interface_1553.
Procedure Allocate : declared in package Dual_Port_Manager.

4.4.5 BI_1553_INTERFACE.Stop_BC

Procedure releases memory and halts interface.

4.4.5.1 Stop_BC Design Specification Constraints.

n.a.

4.4.5.2 Stop_BC Design

- a. Inputs:** n.a.
- b. Outputs:** n.a.
- c. Local Data:** Tbus_Address : A_TBUS_ADDRESS
Tbus_Status : A_TBUS_STATUS
RT_Offset : A_TBUS_ADDRESS
- d. Signals:** n.a.
- e. Algorithms:** Calculate the control register address for the interface.
Write halt code to control register.
Release allocated memory blocks.
- f. Error Handling:** none.
- g. Used Elements:** Procedure Write_Tbus : declared in package Interface_1553.
Procedure Release : declared in package Dual_Port_Manager.

4.4.6 BI_1553_INTERFACE.Start_Monitor

This procedure is used to allocate memory, initialize data structures, and start monitoring all traffic associated with a particular RT.

4.4.6.1 Start_Monitor Design Specification Constraints.

n.a.

4.4.6.2 Start_Monitor Design

- a. Inputs:** Remote_Terminal : A_REMOTE_TERMINAL_ADDRESS
- b. Outputs:** Status : A_BUS_STATUS
- c. Local Data:** RT : A_REMOTE_TERMINAL_ADDRESS
Stat : A_BUS_STATUS
DP_Unibus_Address : NATURAL
Data_Block_Size : INTEGER

d. Signals: n.a.

e. Algorithms: Verify interface usability.
Create a data pointer offset table.
Allocate memory.
Initialize memory.
Start 1553 processor.

f. Error Handling: none.

g. Used Elements: Procedure Verify_Interface : declared in local package body.
Procedure Create_Offset_Table : declared in local package body.
Procedure Allocate_Buffers : declared in local package body.
Procedure Queue_RT : declared in local package body.

4.4.7 BI_1553_INTERFACE.Stop_Monitor

Procedure to release resources and halt interface processor allocated to monitoring operations.

4.4.7.1 Stop_Monitor Design Specification Constraints.

n.a.

4.4.7.2 Stop_Monitor Design

a. Inputs: Remote_Terminal : A_REMOTE_TERMINAL_ADDRESS

b. Outputs: n.a.

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: n.a.

f. Error Handling: none.

g. Used Elements: Procedure Stop_Rt : declared in local package specification.

4.4.8 BI_1553_INTERFACE.Transmit

This is a generic procedure to that will place the user defined data on the bus. The Generic formal parameter A_TRANSMIT_MESSAGE_BUFFER is the user defined data buffer that contains the message data words of the bus message. The A_BUS_OVERHEAD_WORD is either a command word type or a status word depending on the interface type (RT or BC). The Form parameter indicates how the Bus_Word parameter is to be interpreted. The user sets the Form parameter to Command if the currently selected interface has been configured as a BC or Response if the currently selected interface is started as an RT. The Subaddress field is used if the Form is Response to find the correct buffer to place the message into for transmission.

4.4.8.1 Transmit Design Specification Constraints.

n.a.

4.4.8.2 Transmit Design

- a. Inputs:**
 - Data : A_TRANSMIT_MESSAGE_BUFFER
 - Bus_Word : A_BUS_OVERHEAD_WORD
 - Form : MESSAGE_TYPE
 - Subaddress : A_SUBADDRESS
- b. Outputs:** n.a.
- c. Local Data:**
 - Command_Word : A_TRANSMIT_COMMAND
 - Status_Word : A_STATUS_WORD
- d. Signals:** Transmit_Complete : VAX ast
- e. Algorithms:**
 - if Form is Command then get command word and data
 - else get status word and data
 - send message
- f. Error Handling:** none.
- g. Used Elements:**
 - Procedure Write_Status : declared in local package body.
 - Procedure Copy_Block : declared in local package body.

4.4.9 BI_1553_INTERFACE.Receive

This generic procedure reads data from the bus. The generic formal parameters are analogous to the ones in the Transmit procedure. In this RT number and the subaddress must be supplied as the input parameters to filter the desired data from the bus traffic. The Time output is provided to give access to the clock on the interface at the time the data is read.

4.4.9.1 Receive Design Specification Constraints.

n.a.

4.4.9.2 Receive Design

- a. Inputs:** Form : MESSAGE_TYPE
 RT_Number : A_REMOTE_TERMINAL_ADDRESS
 Subaddress : A_SUBADDRESS

- b. Outputs:** Data : A_RECEIVE_MESSAGE_BUFFER
 Bus_Word : A_BUS_OVERHEAD_WORD
 Time : A_TIME_TAG

- c. Local Data:** Status_Word : A_STATUS_WORD
 Base_Address : ADDRESS
 WC : INTEGER
 Command_Word : A_DATA_COMMAND_WORD
 Time_Value : A_TIME
 Status : A_INTERFACE_ERROR
 Direction : A_TRANSFER_DIRECTION

- d. Signals:** Receive_Complete : VAX ast

- e. Algorithms:** If interface running then copy bus data to user variables
 else signal error.

- f. Error Handling:** if CONSTRAINT_ERROR signal data format error.

- g. Used Elements:** Procedure Read_Timer : declared in Interface_1553

4.4.10 BI_1553_INTERFACE.Monitor_RT

This procedure has only one formal generic parameter. This is because both Command and Status words are returned with the data. The concept with this procedure was not to capture all the data on the bus but rather to allow the hardware to filter only the data of interest. The Form parameter in this context is to add filtering by selecting the type of transfer to record; i.e. RT to BC or BC to RT.

4.4.10.1 Monitor_RT Design Specification Constraints.

n.a.

4.4.10.2 Monitor_RT Design

- a. Inputs:** Form : MESSAGE_TYPE
 RT_Number : A_REMOTE_TERMINAL_ADDRESS
 Subaddress : A_SUBADDRESS
- b. Outputs:** Command_Word : A_DATA_COMMAND_WORD
 Data : A_MONITOR_MESSAGE_BUFFER
 Status_Word : A_STATUS_WORD
 Time : A_TIME_TAG
- c. Local Data:** Data_Pnt : ADDRESS
 Data_Word : UNSIGNED_WORD
 Words : INTEGER
 Command_Tmp : A_DATA_COMMAND_WORD
 Direction : A_TRANSFER_DIRECTION
 Time_Value : A_TIME
 Status : A_INTERFACE_ERROR
- d. Signals:** n.a.
- e. Algorithms:** If interface is running get data and time stamp
 else signal error.
- f. Error Handling:** none.
- g. Used Elements:** Procedure Copy_Block : declared in local package body.

4.5 DD CSC Monitor_Executive

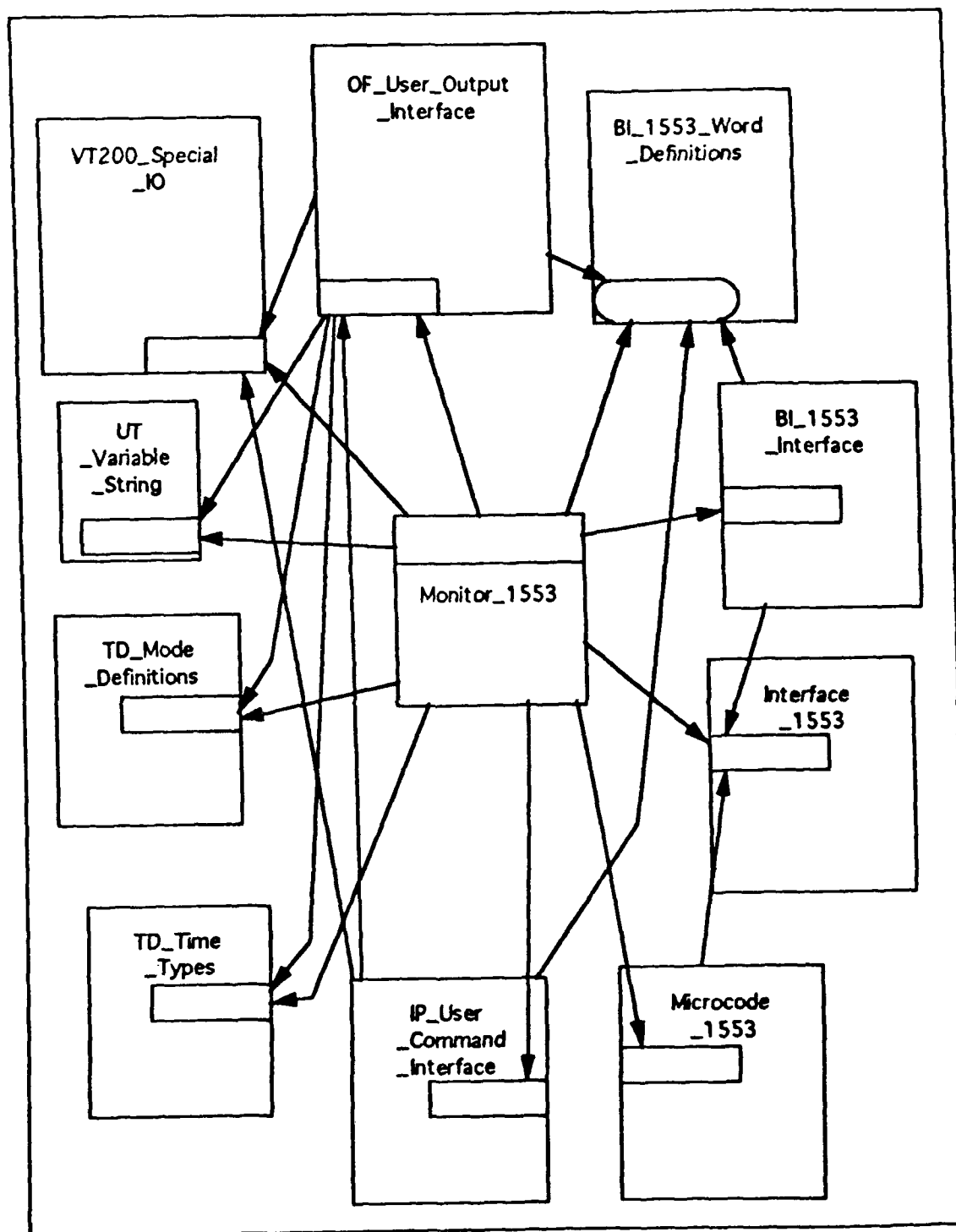


Figure 19 Monitor Executive Ada Structure Diagram

```

with VT200_SPECIAL_IO;           use VT200_SPECIAL_IO;

with UT_VARIABLE_STRING;         use UT_VARIABLE_STRING;
with TD_MODE_DEFINITIONS;        use TD_MODE_DEFINITIONS;
with TD_TIME_TYPES;              use TD_TIME_TYPES;
with INTERFACE_1553;              use INTERFACE_1553;
with MICROCODE_1553;              use MICROCODE_1553;
with BI_1553_WORD_DEFINITIONS;    use BI_1553_WORD_DEFINITIONS;
with BI_1553_INTERFACE;           use BI_1553_INTERFACE;
with OF_USER_OUTPUT_INTERFACE;    use OF_USER_OUTPUT_INTERFACE;
with IP_USER_COMMAND_INTERFACE;    use IP_USER_COMMAND_INTERFACE;

procedure MONITOR_1553 is

    type AN_RT_SELECTION_TABLE is array(A_REMOTE_TERMINAL_ADDRESS) of BOOLEAN;

    type A_SELECTION_DESCRIPTOR is
        record
            RT      : A_REMOTE_TERMINAL_ADDRESS;
            SA      : A_SUBADDRESS;
            TR      : A_TRANSFER_DIRECTION;
            BN      : INTEGER;
            UN      : INTEGER;
            Active  : BOOLEAN := FALSE;
        end record;

    subtype A_SELECTION_INDEX is INTEGER range 1..2;

    type A_SELECTION_ARRAY is array (A_SELECTION_INDEX) of A_SELECTION_DESCRIPTOR;

    type AN_INTERFACE_ARRAY is array (A_INTERFACE_UNIT'pos(A_INTERFACE_UNIT'first)..
                                         A_INTERFACE_UNIT'pos(A_INTERFACE_UNIT'last)) of BOOLEAN;

    MICROCODE_ERROR      : exception;

    Microcode_Filename : constant STRING := "1553$DISK:[ATIP.DEV.MICROCODE]BUSMON3.LIS";

    Monitor_RT          : AN_RT_SELECTION_TABLE := (others => FALSE);
    Interface_Loaded    : AN_INTERFACE_ARRAY := (others => FALSE);
    Current_Sel         : A_SELECTION_ARRAY;
    Next_Sel            : INTEGER := A_SELECTION_INDEX'first;

```

```

Last_Sel      : INTEGER := A_SELECTION_INDEX'first;
New_Sel       : A_SELECTION_DESCRIPTOR;
Mode          : A_MONITOR_MODE := BUS_STATUS;
Format        : A_FORMAT_MODE;
Mode_Change   : BOOLEAN := FALSE;
Format_Change : BOOLEAN := FALSE;
Exit_Loop     : BOOLEAN := FALSE;
First_RT_Selected : BOOLEAN := TRUE;
Command_Rec   : A_VALID_COMMAND_RECORD;
Information    : A_MESSAGE_DATA_BLOCK(TEXT);
Error_Text    : A_MESSAGE_DATA_BLOCK(ERROR);
MicroCode_ID  : A_MICROCODE_ID;
Mic_Error     : A_MICROCODE_ERROR;

function Get_Selection_Data (User_Data : STRING)
    return A_SELECTION_DESCRIPTOR is separate;

procedure Rotate (Index : in out A_SELECTION_INDEX) is separate;

procedure Initialize (Active_RT      : in out AN_RT_SELECTION_TABLE;
                    Interface_Active : in out AN_INTERFACE_ARRAY;
                    Microcode_Ident  : in      A_MICROCODE_ID;
                    New_Selection    : in      A_SELECTION_DESCRIPTOR) is separate;

procedure Deactivate (Selection_List : in out A_SELECTION_ARRAY;
                    Selection        : in      A_SELECTION_DESCRIPTOR) is separate;

procedure Monitor is new BI_1553_INTERFACE.MONITOR_RT(A_MESSAGE_ARRAY);

procedure Display_Data (Selection : in      A_SELECTION_DESCRIPTOR) is separate;

begin

    Start_Display;
    Put_At(24,1,"Reading microcode: " & Microcode_Filename);
    Read_Microcode(Microcode_Filename, MicroCode_ID, Mic_Error);
    if Mic_Error /= SUCCESS then
        raise MICROCODE_ERROR;
    end if;
    Put_At(24,1,"
    Start_Input;

```

```

Assign(Information.Message, "                               Waiting For Command");
Write(Information);

loop
  Get_Next_Command(Command_Rec);
  case Command_Rec.Command is
    when NONE =>
      null;
    when SELECT_RT =>
      New_Sel := Get_Selection_Data(Command_Rec.Command_Data(1..Command_Rec.Data_Length));
      Initialize(Monitor_RT, Interface_Loaded, MicroCode_ID, New_Sel);
      if Monitor_RT(New_Sel.RT) and New_Sel.Active then
        if First_RT_Selected then
          Mode := RAW_DATA;
          Mode_Change := TRUE;
          Format := HEX;
          Format_Change := TRUE;
          First_RT_Selected := FALSE;
        end if;
        Current_Sel(New_Sel) := New_Sel;
        Last_Sel := New_Sel;
        Rotate(New_Sel);
      end if;
    when MODE_HEX =>
      Format := HEX;
      Format_Change := TRUE;
    when MODE_DECIMAL =>
      Format := DEC;
      Format_Change := TRUE;
    when MODE_BINARY =>
      Format := BIN;
      Format_Change := TRUE;
    when MODE_OCTAL =>
      Format := OCT;
      Format_Change := TRUE;
    when MODE_PREFERRED_DATA =>
      Mode := PRIORITY;
      Mode_Change := TRUE;
    when MODE_RAW_DATA =>
      Mode := RAW_DATA;
      Mode_Change := TRUE;
  end case;
end loop;

```



```

when RESET_DISP =>
    Clear_Display;
when CANCEL_SELECTION =>
    Deactivate(Current_Sel,Get_Selection_Data(
        Command_Rec.Command_Data(1..Command_Rec.Data_Lenght)));
when QUIT =>
    Exit_Loop := TRUE;
when others =>
    Put_At(23,1,ASCII.bell &
        "%MON1553-W-UNIMPL, that command is not currently supported.");
end case;
exit when Exit_Loop;
if Mode_Change then
    Set_Mode(Mode);
    Mode_Change := FALSE;
end if;
if Format_Change then
    Set_Format(Format);
    Format_Change := FALSE;
end if;
case Mode is
when RAW_DATA =>
    for Sel in A_SELECTION_INDEX loop
        Display_Data(Current_Sel(Sel));
    end loop;
when PRIORITY =>
    Display_Data(Current_Sel(Last_Sel));
when others =>
    null;
end case;
delay 0.1;
end loop;

Shutdown_Input;
Shutdown_Display;

exception

when others =>
    Shutdown_Input;
    Shutdown_Display;

```

raise;

end MONITOR_1553;

4.5.1 Ex_1553_Monitor.Rotate

This procedure is simply for implementing a binary up counter with roll over characteristic.

4.5.1.1 Rotate Design Specification Constraints.

n.a.

4.5.1.2 Rotate Design

a. Inputs: Index : A_SELECTION_INDEX

b. Outputs: Index : A_SELECTION_INDEX

c. Local Data: n.a.

d. Signals: n.a.

e. Algorithms: if Index is less than last increment Index
else set Index to first.

f. Error Handling: none.

g. Used Elements: n.a.

4.5.2 Ex_1553_Monitor.Initialize

Starts the monitoring of a specific RT.

4.5.2.1 Initialize Design Specification Constraints.

n.a.

4.5.2.2 Initialize Design

a. Inputs: Active_RT : AN_RT_SELECTION_TABLE
Interface_Active : AN_INTERFACE_ARRAY
Microcode_Ident : A_MICROCODE_ID
New_Selection : A_SELECTION_DESCRIPTOR

- b. Outputs:** Active_RT : AN_RT_SELECTION_TABLE
 Interface_Active : AN_INTERFACE_ARRAY
- c. Local Data:** Mic_Error : A_MICROCODE_ERROR
 Mon_Statis : A_BUS_STATUS
- d. Signals:** n.a.
- e. Algorithms:** If New_Selection not Active_RT then...
 If not Interface_Active then Load Microcode_Ident
 Start_Monitor(New_Selection)
 end if
- f. Error Handling:** When Mic_Error report error.
- g. Used Elements:** Procedure Start_Monitor : declared in package BI_1553_Interface.
 Procedure Write : declared in package OF_User_Output_Interface.
 Procedure Assign : declared in UT_Variable_String.
 procedure Load_Microcode : declared in Microcode_1553.

4.5.3 Ex_1553_Monitor.Get_Selection_Data

This function takes a text string containing the selection parrameters entered by a user and converts them to a selection record.

4.5.3.1 Get_Selection_Data Design Specification Constraints.

If the any parameters are out of range the a non active selection is returned and an error message is printed in the Status Area of the output screen.

4.5.3.2 Get_Selection_Data Design

- a. Inputs:** User_Data : STRING
- b. Outputs:** A_SELECTION_DESCRIPTOR
- c. Local Data:** Pos_1 : INTEGER
 Pos_2 : INTEGER

Arg : INTEGER
Int_Data : INT_ARRAY
More : BOOLEAN
Selection : A_SELECTION_DESCRIPTOR

- d. Signals: n.a.
- e. Algorithms: parse String for 5 selection parameters
- f. Error Handling: When CONSTRAINT_ERROR write error message, return inactive default record.
- g. Used Elements: n.a.

4.5.4 Ex_1553_Monitor.Deactivate

This procedure sets the selected element of a display table to inactive. This stops all data display on this selection and makes it's slot available for reuse.

4.5.4.1 Deactivate Design Specification Constraints.

n.a.

4.5.4.2 Deactivate Design

- a. Inputs: Selection_List : A_SELECTION_ARRAY
Selection : A_SELECTION_DESCRIPTOR
- b. Outputs: Selection_List : A_SELECTION_ARRAY
- c. Local Data: n.a.
- d. Signals: n.a.
- e. Algorithms: If Selection in Selection_List make inactive.
- f. Error Handling: none.
- g. Used Elements: n.a.

4.5.5 Ex_1553_Monitor.Display_Data

Read the bus buffer associated with a selection and if new data is present display it on the data output display.

4.5.5.1 Display_Data Design Specification Constraints.

n.a.

4.5.5.2 Display_Data Design

a. Inputs: Selection : A_SELECTION_DESCRIPTOR

b. Outputs: n.a.

c. Local Data: Command_Word : A_DATA_COMMAND_WORD;
Status_Word : A_STATUS_WORD
Message_Form : A_MESSAGE_TYPE
Bus_Message : A_MESSAGE_ARRAY
Time_Tag : A_TIME_TAG
Valid_Cmdnd : BOOLEAN

d. Signals: n.a.

e. Algorithms: If Selection active then get bus data.
If bus data current then write data to display.

f. Error Handling: none.

g. Used Elements: Procedure Monitor_RT : declared in package
BI_1553_INTERFACE
Procedure Write : declared in package
OF_USER_OUTPUT_INTERFACE

5 CSCI DATA

The only global data shared between packages is in the form of constant definitions. The remainder of the global data in this CSCI is local to a particular package body. The following subparagraphs outline the inter and intra package global data.

5.1 Intra Package Global Data

DUAL_PORT_MANAGER

DP_Memory	DUAL_PORT_MEMORY	
Global structure to map BIT-3 into		
Section_Created	BOOLEAN	FALSE
Set to true is section didn't preexist.		
Section_Mapped	BOOLEAN	FALSE
Set to true if map function was successful		
Lock_Held	BOOLEAN	FALSE
Set true if Init lock is aquired.		
Lock_Status_Block	STARLET.LOCK_STATUS_BLOCK_TYPE	
Results of request to que Init lock.		
DP_Initialized	BOOLEAN	FALSE
True after first call to Initialize_D._P.		
Next_Block_Pnt	SHORT_INTEGER	
Used to keep place in Next_Block function		
MAX_DUAL_PORT_SIZE	constant	128 * 1024
64K Words		
BYTE	constant	8
8 Bits 1 byte		
WORD	constant	16
16 Bits 1 Word		
ALLOC_TABLE_SIZE	constant	1024 * BYTE
1K Byte in bits		
ALLOC_TABLE_NAME	constant STRING	
"DP_ALLOCATION_TABLE" name of lock		
GSD_Name	constant STRING	"DUAL_PORT"
"DUAL_PORT"		
Test_Patern_1	constant UNSIGNED_BYTE	16#AA#
Test_Patern_2	constant UNSIGNED_BYTE	16#55#

AM_BACKGROUND_MONITOR

Current_RT_State_Table	AN_RT_STATE_ARRAY	
Past_Duration	SECONDS	
The_Sample_Rate	SAMPLES_PER_SECOND	
The_1553_Interface_Unit	A_Interface_Unit	

AM_TIMING_PACKAGE

CONTEXT_ADR		
CONTEXT'ADDRESS		

BI_1553_INTERFACE

Offsets	AN_RT_OFFSET_POINTER	null
Byte offsets to Subaddress Buffers		
Data	SYSTEM.ADDRESS	ADDRESS_ZERO
Begining address of dual port block		
Rec_SA	A_SUBADDRESS_POINTER_TABLE	(others =>
ADDRESS_ZERO)	Addresses of Receive SA buffers	
Trx_SA	A_SUBADDRESS_POINTER_TABLE	(others =>
ADDRESS_ZERO)	Addresses of Transmit SA buffers	
Status	A_STATUS_WORD	
Default_Status_Word	Current status response for RT	
Time	A_SUBADDRESS_TIME_TABLE	(others => 0)
The relative time of the last message		
Runing	BOOLEAN	FALSE
Flag true if Start_RT successfull		
Bus_Num	INTERFACE_1553.A_BUS_NUMBER	
RT_Num	AN_RT_DESCRIPTOR_TABLE	
RT_Table	AN_INTERFACE_TABLE	
Struture to competely describe hardware environment.		
Interface	A_INTERFACE_UNIT	
Globally visable object to hold interface number.		
Unit_Selected	BOOLEAN	FALSE
	Flag to indicate if a call to Select_Unit has been	

made.

RT_Control_Block_Base	constant	16#0800# rt
control block base		
RT_Ctrl_Blk_ESW	constant	16#0000#
emulator status word offset		
RT_Ctrl_Blk_ECW	constant	16#0001#
emulator control word offset		
RT_Ctrl_Blk_Last_Cmd	constant	16#0002#
last comand offset		
RT_Ctrl_Blk_BJW	constant	16#0003#
bit jump word offset		
RT_Ctrl_Blk_VEC	constant	16#0004#
interrupt vector offset		
RT_Ctrl_Blk_BAP	constant	16#0005#
buffer address pointer offset		
RT_Ctrl_Blk_CAP	constant	16#0006#
chain address pointer		
RT_Ctrl_Blk_ATP	constant	16#0007#
address table pointer		
RT_Ctrl_Blk_STAT	constant	16#0008#
status word offset		
RT_Ctrl_Blk_OMD	constant	16#0009#
output mode data offset		
RT_Ctrl_Blk_MCW	constant	16#000a#
memory control word offset		
RT_Ctrl_Blk_ATBA	constant	16#000b#
address table base address offset		
RT_Ctrl_Blk_IMD	constant	16#000c#
input mode data offset		
RT_Ctrl_Blk_MSGC	constant	16#000d#
message counter offset		
RT_Ctrl_Blk_MM1	constant	16#000e#
mode mask 1 offset		
RT_Ctrl_Blk_MM2	constant	16#000f#
mode mask 2 offset		
RT_Ctrl_Blk_UCBH	constant	16#0010#
UCB high offset		

RT_Ctrl_Blk_UCBL	constant	16#0011#
UCB low offset		
RT_Ctrl_Blk_DLAY	constant	16#0012#
delay offset		
RT_Ctrl_Blk_CMD2	constant	16#0014#
command word 2 offset		
RT_Ctrl_Blk_SUBA	constant	16#0015#
subaddress offset		
RT_Ctrl_Blk_MSGT	constant	16#0016#
message type offset		
RT_Ctrl_Blk_BFHI	constant	16#0018#
buffer address high offset		
RT_Ctrl_Blk_BFLO	constant	16#0019#
buffer address low offset		
RT_Ctrl_Blk_MCHI	constant	16#001a#
msg ctr high address offset		
RT_Ctrl_Blk_MCLO	constant	16#001b#
msg ctr low address offset		
RT_Ctrl_Blk_HSHK	constant	16#001d#
handshake flag offset		
RT_Ctrl_Blk_SYNC	constant	16#001e#
sync type msg flag offset		
RT_Ctrl_Blk_MCYC	constant	16#001f#
minor cycle offset		
RT_Ctrl_Blk_RBUF	constant	16#0020#
receive buffer offset		
RT_Ctrl_Blk_CARB	constant	16#0021#
circular asynch. rcv buffer offset		
RT_Ctrl_Blk_TBUF	constant	16#0022#
transmit buffer offset		
RT_Ctrl_Blk_CATB	constant	16#0023#
circular asynch xmt buffer		
RT_Ctrl_Blk_RCBH	constant	16#0024#
receive circular buffer head offset		
RT_Ctrl_Blk_RCBT	constant	16#0025#
receive circular buffer tail offset		
RT_Ctrl_Blk_TCBH	constant	16#0026#

```

transmit circular buffer head offset |
RT_Ctrl_Blk_TCBT          | constant          | 16#0027# |
transmit circular buffer tail offset |
RT_Ctrl_Blk_INF1          | constant          | 16#0028# |
info 1 offset              |
RT_Ctrl_Blk_INF2          | constant          | 16#0028# |
info 1 offset              |
RT_Ctrl_Blk_SRCV          | constant          | 16#0029# |
synch rcv count offset     |
RT_Ctrl_Blk_SXMT          | constant          | 16#002a# |
synch xmt count offset     |
RT_Ctrl_Blk_ARCV          | constant          | 16#002b# |
asynch rcv count offset    |
RT_Ctrl_Blk_AXMT          | constant          | 16#002c# |
asynch xmt count offset    |
RT_Ctrl_Blk_QV0           | constant          | 16#0030# |
queued vector 0 offset     |
RT_Ctrl_Blk_SPBH          | constant          | 16#0038# |
synch rcv buffer high offset |
RT_Ctrl_Blk_SRBL          | constant          | 16#0039# |
synch rcv buffer low offset |
RT_Ctrl_Blk_STBH          | constant          | 16#003a# |
synch xmt buffer high offset |
RT_Ctrl_Blk_STBL          | constant          | 16#003b# |
synch xmt buffer low offset |
INTERFACE_AT_BASE         | constant          | 16#1000# |
base address of offset table in interface local ram
UNIBUS_BASE_ADDRESS       | constant          | 8#000000# |
base address of dual port memory on the unibus
Default_Status_Word       | constant A_STATUS_WORD |
(0,FALSE,FALSE,FALSE,FALSE,

```

EX_1553_MONITOR

```

Monitor_RT                | AN_RT_SELECTION_TABLE | (others =>
FALSE)
Interface_Loaded           | AN_INTERFACE_ARRAY    | (others =>

```

```

FALSE)
Current_Sel          | A_SELECTION_ARRAY      |
Next_Sel             | INTEGER                |
A_SELECTION_INDEX'first
Last_Sel             | INTEGER                |
A_SELECTION_INDEX'first
New_Sel              | A_SELECTION_DESCRIPTOR |
Mode                 | A_MONITOR_MODE         | BUS_STATUS
Format               | A_FORMAT_MODE          |
Mode_Change          | BOOLEAN                | FALSE
Format_Change        | BOOLEAN                | FALSE
Exit_Loop            | BOOLEAN                | FALSE
First_RT_Selected    | BOOLEAN                | TRUE
Command_Rec          | A_VALID_COMMAND_RECORD |
Information           | A_MESSAGE_DATA_BLOCK(TEXT)
Error_Text           | A_MESSAGE_DATA_BLOCK(ERROR)
MicroCode_ID         | A_MICROCODE_ID         |
Mic_Error            | A_MICROCODE_ERROR      |
Microcode_Filename   | constant STRING        |
"1553$DISK|[ATIP.DEV.MICROCODE]BUSMON3.LIS"

```

INTERFACE_1553

```

Registers            | A_Register_Map         |
Interface_Mapped      | Boolean                | FALSE
Setup_File_Read       | Boolean                | FALSE
Unit_Status           | A_UNIT_STATUS          |
(NOT_INSTALLED,
Base_Address          | A_BASE_ADDRESS         |
Base_CSR              | A_CSR_ADDRESS          |
Unit_Num              | A_INTERFACE_UNIT       |
Interface_Error       | A_INTERFACE_ERROR      |
Setup_File_Name       | constant string        | "PORTS_1553"
SOURCE_BASE_REG_ADDR  | constant A_TBUS_ADDRESS | 16#2007#
DESTINATION_BASE_REG_ADDR | constant A_TBUS_ADDRESS | 16#200F#
IO_BASE_ADDRESS       | constant A_TBUS_VALUE  | 16#2000#
BUS_REGISTER_PRIME    | constant A_TBUS_ADDRESS | 16#2003#

```

BUS_REGISTER_ALTER	constant A_TBUS_ADDRESS	16#200B#
HOST_DATA	constant A_TBUS_ADDRESS	16#0013#
HOST_COMMAND	constant A_TBUS_ADDRESS	16#0010#
HOST_REQUEST	constant A_TBUS_VALUE	16#0001#
TIMER_ADDRESS	constant A_TBUS_ADDRESS	16#2020#

IP_USER_COMMAND_INTERFACE

Command_Queue	A_FIFO_QUEUE	
---------------	--------------	--

MICROCODE_1553

New_Id	A_MICROCODE_ID	0
Microcode_List	A_MICROCODE_LIST	

OF_USER_OUTPUT_INTERFACE

Main_Display	A_BUFFER_ID	
Message_Screen	A_BUFFER_ID	
Error_Screen	A_BUFFER_ID	
Raw_Data_Screens	A_RAW_DATA_SCREEN_TABLE	
Priority_Screens	A_PRIORITY_DATA_SCREEN_TABLE	
RT_Activity	A_BUFFER_ID	
Formatted_Screen	A_BUFFER_ID	
Logging_Screen	A_BUFFER_ID	
Bus_Stat_Screen	A_BUFFER_ID	
Data_Screen	A_BUFFER_ID	
Display_Pages	A_DISPLAY_TABLE	
RATE_CHANGE	BOOLEAN	TRUE

SCREEN_MANAGER

Screen_Buffer	A_SCREEN_MANAGEMENT_BUFFER	
Next_Buffer	A_BUFFER_ID	
A_BUFFER_ID'first		
More_Buffers	BOOLEAN	TRUE
CSI	constant STRING	ASCII.esc &

"["		
POSLEN	constant INTEGER	CSI'length +
2 CSI length + length + f length		
ATRCHR	constant AN_ATTRIBUTE_TABLE	
('0','1','4','5','7')		

UNBOUNDED_FIFO_QUEUE

Master_Key	AN_ENTRY_KEY	
AN_ENTRY_KEY'first		

VT200_SPECIAL_IO

ESC	constant CHARACTER	ascii.esc
CTL	constant STRING(1..2)	ESC & '['
Escape_Sequence	constant SCREEN_CODES	(INVERSE
=> (6,4, CTL & "7m "),		
Attribute_On	constant ATTRIBUTE_CODES	(INVERSE_TXT
=> (5,4, CTL & "7m "),		
Attribute_Off	constant ATTRIBUTE_CODES	(INVERSE_TXT
=> (5,5, CTL & "27m"),		
All_Off	constant STRING	CTL &
"022242527m"		
Turn_Cursor_On	constant STRING	CTL & "?25h"
Turn_Cursor_Off	constant STRING	CTL & "?25l"

5.2 Inter Package Global Definitions

OF_LOCAL_SPECS

MAX_NUMBER_RT_DISPLAYS	constant	2
MAIN_HEADER_ROW1	constant NATURAL	1
MAIN_HEADER_COL1	constant NATURAL	1
SEC_HEADER_ROW1	constant NATURAL	2
SEC_HEADER_COL1	constant NATURAL	1
SEC_HEADER_COL2	constant NATURAL	20
SEC_HEADER_COL3	constant NATURAL	32

SEC_HEADER_COL4	constant NATURAL	51
SEC_HEADER_COL5	constant NATURAL	71
THRID_HEADER_ROW1	constant NATURAL	3
THRID_HEADER_COL1	constant NATURAL	1
THRID_HEADER_ROW2	constant NATURAL	4
WORK_COMMAND_SEP_ROW	constant NATURAL	22
WORK_COMMAND_SEP_COL	constant NATURAL	1
DISPLAY_START_ROW	constant NATURAL	4
DISPLAY_STOP_ROW	constant NATURAL	21
DISPLAY_START_COL	constant NATURAL	
MAIN_HEADER_COL1		
ACT_MON_ROW	constant NATURAL	2
ACT_MON_COL	constant NATURAL	10
TEXT_HEADER_ROW	constant NATURAL	8
TEXT_HEADER_COL	constant NATURAL	30
ERROR_HEADER_ROW	constant NATURAL	8
ERROR_HEADER_COL	constant NATURAL	30
TEXT_ROW	constant NATURAL	10
TEXT_COL	constant NATURAL	1
ERROR_ROW	constant NATURAL	10
ERROR_COL	constant NATURAL	1
RT_ACT_ROW1	constant NATURAL	
THRID_HEADER_ROW1		
RT_ACT_ROW2	constant NATURAL	
THRID_HEADER_ROW2		
RT_ACT_COL	constant NATURAL	
THRID_HEADER_COL1 + 6		
RT_OFFSET_ROW1	constant NATURAL	4
RT_OFFSET_ROW2	constant NATURAL	3
RT_SA_BN_COL	constant NATURAL	6
RT_HEADER_ROW	constant NATURAL	6
SA_HEADER_ROW	constant NATURAL	7
BN_HEADER_ROW	constant NATURAL	8
RT_RT_HEADER_OFFSET	constant NATURAL	35
WORK_AREA_OFFSET	constant NATURAL	14
BIN_WORK_AREA_OFFSET	constant NATURAL	30
NUMBER_OFFSET	constant NATURAL	8

START_AREA_WORK_ROW	constant NATURAL	RT_HEADER_ROW
DATA_OFFSET	constant NATURAL	11
WORK_AREA_COL1	constant NATURAL	RT_SA_BN_COL
+ DATA_OFFSET		
WORK_AREA_COL2	constant NATURAL	RT_SA_BN_COL
+ DATA_OFFSET + BIN_WORK_AREA_OFFSET		
RT_HEADER_COL	constant NATURAL	3
HEADER_AREA_OFFSET	constant NATURAL	20
RAW_DATA_BIN_OFFSET	constant NATURAL	23
RAW_DATA_DEF_OFFSET	constant NATURAL	14
PRIORITY_COL	constant NATURAL	2
PRIORITY_ROW	constant NATURAL	10
PRIORITY_HEADER_ROW	constant NATURAL	6
PRIORITY_HEADER_COL	constant NATURAL	30
PRIORITY_CMD_COL	constant NATURAL	PRIORITY_COL
+ 25		
PRIORITY_CMD_ROW	constant NATURAL	PRIORITY_ROW
+ 9		
PRIORITY_STS_ROW	constant NATURAL	
PRIORITY_CMD_ROW + 1		
PRIORITY_CMD_OFFSET	constant NATURAL	10
LOGGING_ROW	constant NATURAL	11
LOGGING_COL	constant NATURAL	30
BLANK_LINE	constant STRING(1..80)	(others => '
')		
NORMAL_TEXT	constant A_SCREEN_ATTRIBUTE	Attr(NORMAL)
UNDERLINE_TEXT	constant A_SCREEN_ATTRIBUTE	
Attr(UNDERLINE)		
BOLD_TEXT	constant A_SCREEN_ATTRIBUTE	Attr(BOLD)
BLINKING_TEXT	constant A_SCREEN_ATTRIBUTE	Attr(BLINKING)
INVERSE_TEXT	constant A_SCREEN_ATTRIBUTE	Attr(INVERSE)
BLINKING_INVERSE_TEXT	constant A_SCREEN_ATTRIBUTE	BLINKING and
INVERSE		
BOLD_UNDERLINE_TEXT	constant A_SCREEN_ATTRIBUTE	BOLD and
UNDERLINE		
BLINKING_UNDERLINE_TEXT	constant A_SCREEN_ATTRIBUTE	BLINKING and
UNDERLINE		

INVERSE_UNDERLINE_TEXT	constant A_SCREEN_ATTRIBUTE	INVERSE and
UNDERLINE		
BOLD_INVERSE_TEXT	constant A_SCREEN_ATTRIBUTE	BOLD and
INVERSE		
BOLD_INVERSE_UNDERLINE_TEXT	constant A_SCREEN_ATTRIBUTE	BOLD and
INVERSE and UNDERLINE		
DATA_POSITION	constant A_DISPLAY_FORMAT_TABLE	
PRIORITY_POS	constant A_PRIORITY_FORMAT_TABLE	

BI_1553_WORD_DEFINITIONS

Broadcast_Address	constant A_REMOTE_TERMINAL_ADDRESS	2#11111#;
Mode_Command_1	constant A_SUBADDRESS	2#11111#;
Mode_Command_2	constant A_SUBADDRESS	2#00000#;

6 CSCI DATA FILES

6.1 Data File to CSC-CSU Cross Reference

Data file PORTS_1553 used by INTERFACE_1553

6.1.1 PORTS_1553.

This file contains information on the type of CPU, number of UNIBUS channels, number of interfaces, and IO register addresses for the target platform. The file is order dependent, the data may have any number of comments between it but each item must be on a separate line, and must come in the specified order. The semicolon (;) is the comment character.

IO_BASE_PAGE_ADDRESS - this is the VAX address of the io page buffer in hex

Range: <implementation dependant>

Units: UNSIGNED

TOTAL_NUMBER_MMBI_INSTALLED - the number of interface adaptor units on this CPU

Range : 1..4

Units: UNSIGNED

UNIT_ASSIGNMENTS - one definitin per MMBI installed

UNIT # CSR - octal representation of the control status register address for the interface

Range: 766000₈ .. 766100₈

Units: UNSIGNED

8 NOTES

8.1 Definition Of Terms and Abbreviations

AN/AYK	-	(Navy standard embedded mission processor)
ATIP	-	Ada Technology Insertion Program
BC	-	Bus Controller
CSCI	-	Computer Software Configuration Item.
CSU	-	Computer Software Unit.
CTSD	-	Computer Technology and Simulation Department.
DCL	-	Digital Command Language
DEC	-	Digital Equipment Corporation
GADBTk	-	Generic Avionics Data Bus Tool Kit
RT	-	Remote Terminal
UART	-	Universal Asynchronous Receiver / Transmitter
VAX	-	Virtual Address Extension (DEC mini computer architecture)
VMS	-	Virtual Memory System (VAX operating system)

APPENDIX A

10 Requirements Model Dictionary

10.1 Data Flows

Bus_Activity =

BNF: {RT_Status}

Bus_Data =

BNF: [RT_Status | Bus_Message]

Bus_Message =

BNF: {Bus_Word}

Captured_Data =

BNF: {Bus_Word}

Composed_Command =

BNF: Current_Command + User_Data

Configuration_Info =

BNF: Default_Bus_Number + Interface_Number + Interface_Address
+ Machine_Type

Data_Received =

BNF: {Bus_Word}

Data_Scale_Definition =

BNF: {Message_Id + Word_Num + LSB + MSB + Scale_Factor}

Data_Selection =

BNF: RT_Num + (SA_Num) + (Tran_Rec) + (Bus_Num) + (Watch_Mask)

Data_To_Transmit =

BNF: {Bus_Word}

Device_Parameters =

BNF: Num_Lines + Num_Cols + Esc_Clear + Esc_Pos + Esc_Line + Esc_Bold
+ Esc_Hilite + Esc_Blink

Message_Data =

BNF: Message_ID_Block + Message_Content

Message_ID_Block =

BNF: Message_Type + (Window_Id)

Next_Command =

BNF: Current_Command

Raw_Bus_Data =

BNF: {Bus_Word}

Scaled_Data =

BNF: {Bus_Word}

Screen_Layout =

BNF: Elements + {Title + Line + Column + Units + Width}

Text_Data =

BNF: {Bus_Word_String}

Transmit_Data =

BNF: {Bus_Word}

User_Data =

BNF: [RT_Num | SA_Num | Tran_Rec | Bus_Num | Bus_Message | Out_File
| Update_Rate | Current_Mode | Watch_Mask]

Verified_Data =

BNF: {Bus_Word}

10.2 Data Stores

Command_History =
BNF: {Composed_Command}

Command_Queue =
BNF: {Current_Command}

Data_Format_Table =
BNF: {Valid_Input_Format}

Data_Scaling_Table =
BNF: {Data_Scale_Definition}

Device_Parameter_Table =
BNF: {Device_Name + Device_Parameters}

Output_Format_Table =
BNF: {Screen_Layout}

Valid_Command_Table =
BNF: {Valid_Command}

10.2 Terminators

1553_Bus =

Bus_Configuration_File =

History_File =

User =

10.2 Processes

Activity_Monitor =

As_Text =

Bus_Interface =

Bus_Monitor =

Bus_Receiver =

Bus_Transmitter =

Compare_Commands =

Data_Manager =

Disable_Com_Firmware =

Dispatch_Command =

Format_Data =

Get_Command =

Get_User_Data =

Initialize_Bus_Hardware =

Initiate_Com_Firmware =

Input_Parser =

Interpreter =

Mode_Control =

Monitor_EXEC =

Output_Formater =

Output_Mode =

Scale_Data =

Screen_Interface =

Update_Clock =

Update_Counter =

Validate_Command =

Verify_Data =

10.2 Control Flows

Get_Data =

Init_Interface =

Read_Data =

Resume_Command_Entry =

Start_Emu =

Start_Log =

Stop_Emu =

Stop_Log =

Update_Screen =

Write_Data =

10.3 Primitives

Bus_Num =

Data Type: Integer

Data Representation: numeric

Data Size: 4 bytes

Data Range: (1, 8)

Default Value: 1

Bus_Word =

Data Type: Unsigned_Word

Data Representation: numeric

Data Size: 2 bytes

Data Range: (0, 65536)

Bits: 16

Bus_Word_String =

BNF: {Character}

Data Type: String

Data Representation: alpha-numeric

Column =

Data Type: Integer

Data Representation: numeric

Data Range: (0, 132)

Default Value: 0

Command =

BNF: {Character}

Comments: Command input from user is an unformatted string that must match one of the valid commands.

Data Type: String

Data Representation: alpha_numeric

Data Size: 80 bytes

Commanded_Mode =

BNF: [Hex + Decimal + Octal + Binary + Formated + Activity + Prefered_Data + Idle
+ Logging]

Data Type: Mode_Type

Current_Command =

BNF: [Select_RT + Select_SA + Select_TR + Select_Bus + Set_Display_Mode
+ Set_Log_Mode + Start_Loging]

Data Type: Command_Type

Data Representation: Enumeration

Current_Mode =

BNF: [Hex + Decimal + Octal + Binary + Formated + Activity + Prefered_Data + Idle
+ Logging]

Data Type: Mode_Type

Default_Bus_Number =

Comments: MFS data bus selection

Data Type: Bus_Num

Data Representation: enumeration

Device_Name =

BNF: [VT100 + VT200 + UNKNOWN]

Data Type: Terminal_Device_Type

Data Representation: enumeration

Default Value: VT200

Elements =

Comments: Number of Entries on the particular screen

Data Type: Natural

Data Representation: numeric

Data Range: (0, 3168)

Esc_Blink =

BNF: {Character}

Comments: Output device escape sequence to set blinking atributre on chracters.

Data Type: String

Data Representation: alpha-numeric

Esc_Bold =

BNF: {Character}

Comments: Escape sequence to set bold attribute on output device.

Data Type: String

Data Representation: alpha-numeric

Esc_Clear =

BNF: {Character}

Comments: Escape sequence to clear the screen on the output device

Data Type: String

Data Representation: alpha-numeric

Esc_Hilite =

BNF: {Character}

Comments: Escape sequence to set the hilite attribute for the output device.

Data Type: String

Data Representation: alpha-numeric

Esc_Line =

BNF: {Character}

Comments: Escape sequence to cause a line to be drawn on the output device.

Data Type: String

Data Representation: alpha-numeric

Esc_Pos =

BNF: {Character}

Comments: Escape sequence to position the cursor on the output device.

Data Type: String

Data Representation: alpha-numeric

Input_Command =

BNF: {Character}

Comments: Formated text string containing a user command

Data Type: String

Data Representation: alpha-numeric

Input_Data =

BNF: {Character}

Data Type: String

Data Representation: alpha-numeric

Interface_Address =

Comments: CSR address of VAX 1553 interface card

Data Type: Address

Data Representation: numeric

Interface_Number =

Comments: MFS unibus interface number

Data Type: Integer

Data Representation: numeric

Data Range: (1, 4)

Default Value: 1

Line =

Data Type: Integer

Data Representation: numeric

Data Range: (0, 24)

Default Value: 0

Log_File_Name =

BNF: {Character}

Data Type: String

Data Representation: alpha-numeric

Default Value: "BUS_LOG.OUT"

LSB =

Comments: Least Significant Bit of data

Data Type: Integer

Data Representation: numeric

Machine_Type =

BNF: [VS3200 + MVII + V780 + V6X + V8X + AYK]

Comments: Type of computer the software is exucuting on

Data Type: A_Machine_Type

Data Representation: enumeration

Message_Content =

BNF: [{Character} | {Bus_Word}]

Comments: This field contains the text of the message to be displayed

Data Type: String

Data Representation: alpha-numeric

Message_Id =

Comments: The message number associated with the descriptor

Data Type: Integer

Data Representation: numeric

Message_Type =

BNF: [Info + Error + Bus_Data + Prompt]

Data Type: Output_Message_Type

Data Representation: enumeration

MSB =

Comments: Most Significant Bit of the message data

Data Type: Integer

Data Representation: numeric

Numeric_Data =

BNF: {Integer}

Data Type: Numeric_Display_Data

Num_Cols =

Comments: Maximum number of colomns on the output device.

Data Type: Positive

Data Representation: numeric

Data Range: (1, 1024)

Default Value: 80

Num_Lines =

Comments: Number of lines on output device.

Data Type: Positive

Data Representation: numeric

Data Range: (1, 768)

Default Value: 24

Out_File =

BNF: {Character}

Comments: File name of output stream

Data Type: String

Data Representation: alpha-numeric

Receive_Buffer_Address =

Comments: Virtual address of physical memory buffer

Data Type: Address

Data Representation: numeric

RT_Num =

Comments: Remote Terminal address

Data Type: RT_Address_Type

Data Representation: numeric

Data Range: (0, 31)

Bits: 5

RT_Status =

BNF: [Active + Inactive + Unknown]

Comments: Most recently observed status of a given RT

Data Type: An_RT_Status

Data Representation: enumeration

SA_Num =

Comments: Subaddress of selected RT

Data Type: RT_Subaddress_Type

Data Representation: numeric

Data Range: (1, 30)

Scale_Factor =

Comments: The multiplier to convert the raw data number to the internal representation

Data Type: Float

Data Representation: numeric

Screen_Selection =

Comments: Integer to specify screen display format to use with current data

Data Type: Integer

Data Representation: numeric-coded

Startup_Vector =

Comments: Microcode starting location

Data Type: Address

Data Representation: numeric

Status =

BNF: [Normal + Bus_Error + User_Error + Config_Error]

Status_Message =

BNF: {Character}

Comments: Errors, warnings, and information associated with the software not the actual bus traffic.

Data Type: String

Data Representation: alphabetic

Title =

BNF: {Character}

Comments: Text to associate with the screen field

Data Type: String

Data Representation: alpha-numeric

Transmit_Buffer_Address =

Comments: Virtual memory address of physical buffer

Data Type: Address
Data Representation: numeric

Tran_Rec =

BNF: [Transmit + Receive]
Comments: Direction of data transfer
Data Type: Data_Direction_Type
Data Representation: enumeration

Units =

BNF: [Seconds + Minutes + Hours + Inches + Feet + Miles + CentiMeters + Meters
+ KiloMeters + Degrees + Semicircles + In_Per_Sec + Ft_Per_Sec + Mi_Per_H
+ M_Per_Sec + Ft_Per_Sec2 + M_Per_Sec2]
Comments: Units to display screen field in.
Data Type: Avionics_Units
Data Representation: enumeration

Update_Rate =

BNF: Seconds
Comments: This message to the output formater expresses how often to refresh the
screen.
Data Type: Integer
Data Representation: numeric
Data Range: (0, 60)
Default Value: 3
Units of Measure: Seconds
Accuracy: 1

Valid_Command =

BNF: [Select_RT + Select_SA + Select_TR + Select_Bus + Set_Display_Mode
+ Set_Log_Mode + Set_Watch + Start_Loing]
Data Type: Valid_Command_Type
Data Representation: Enumeration

Valid_Data =

BNF: [Integer | String | Enumeration]

Comments: This flow contains data that has been verified to be of the format requested
the values have not been tested

Valid_Input_Format =

Comments: Template of the input expected to check against actual input.

Watch_Mask =

BNF: {Bits}

Comments: Bit mask to use as logging criteria. (i.e. when pattern is seen on bus log the
message)

Data Type: Bit_Array

Data Representation: boolean

Width =

Comments: The number of spaces to use on the output for representing the field.

Data Type: Natural

Data Representation: numeric

Data Range: (0, 132)

Windows =

Comments: The number of data windows to show on the screen at one time.

Data Type: Integer

Data Representation: numeric

Data Range: (1, 4)

Default Value: 1

Window_Id =

Data Type: Integer

Data Representation: numeric

Data Size: 1 byte

Data Range: (1, 4)

Default Value: 1

Word_Num =

Comments: Subaddress word offset

Data Type: An_RT_Word_Number

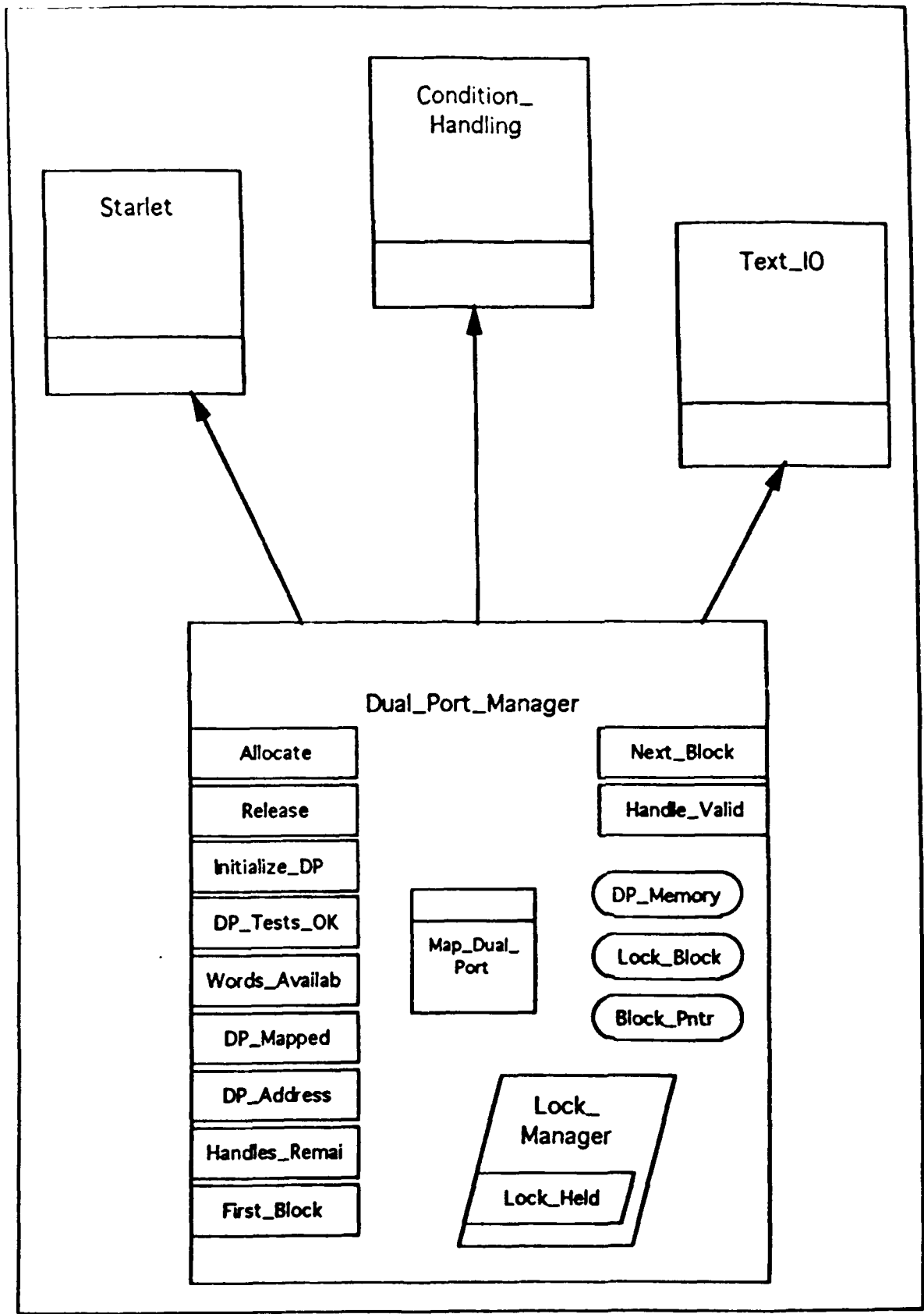
Data Representation: numeric

Data Range: (0, 31)

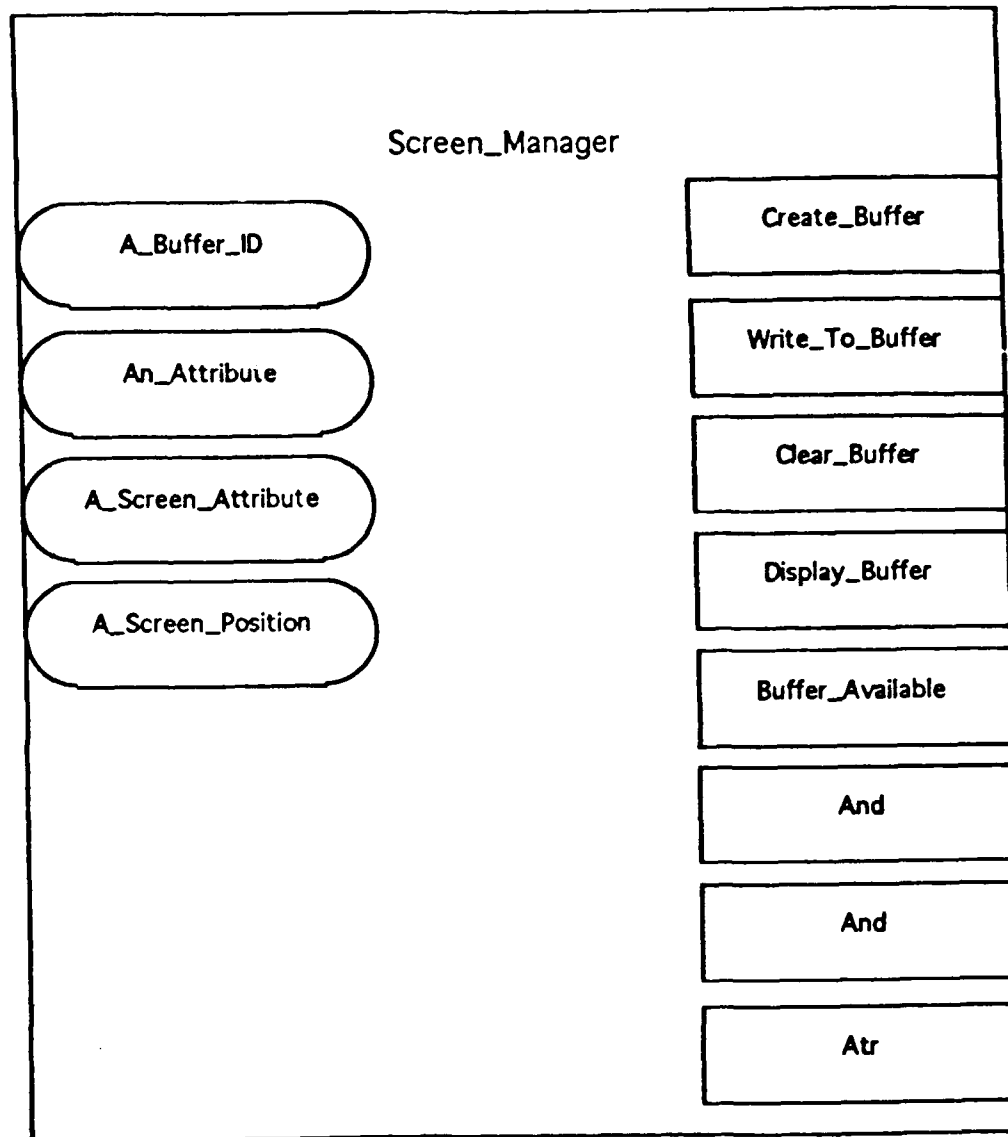
APPENDIX B

10 ADDITIONAL ADA STRUCTURE CHARTS FOR CSC's

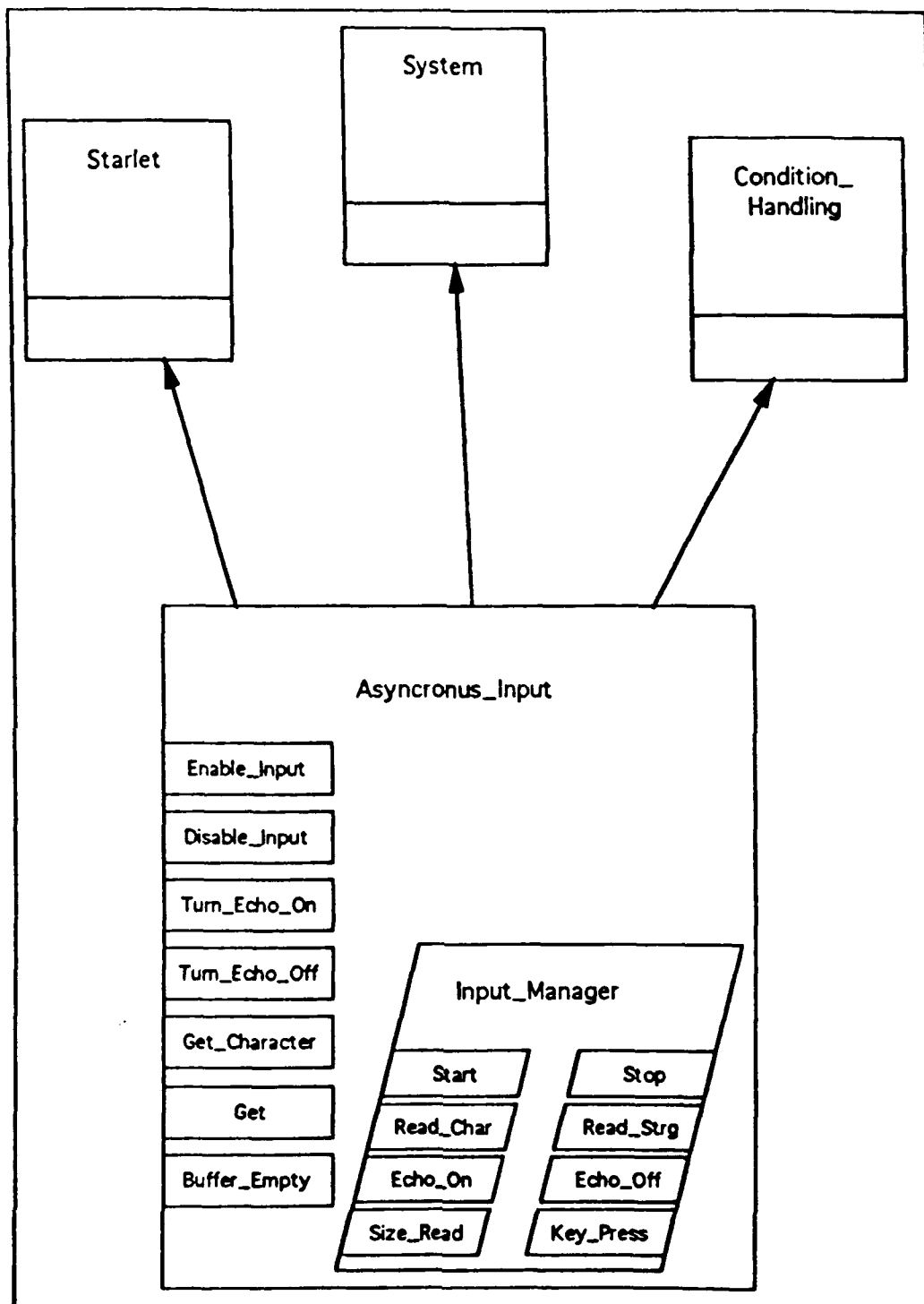
10.1 Daul Port Manager



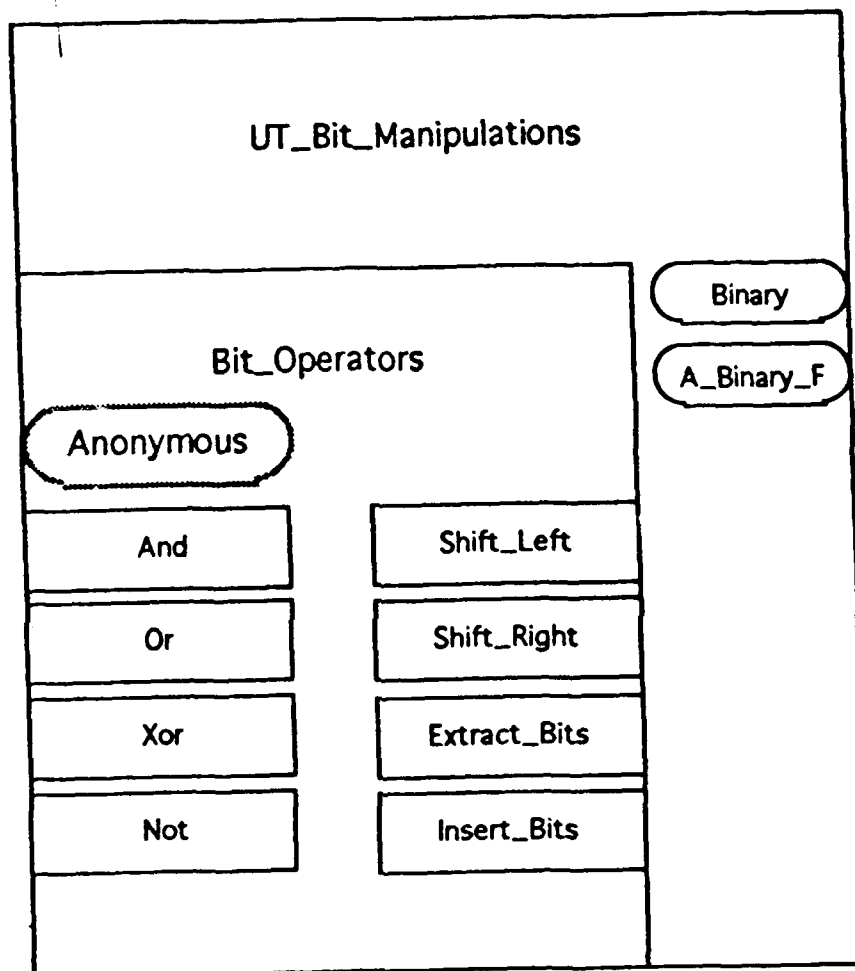
10.2 Screen Manager



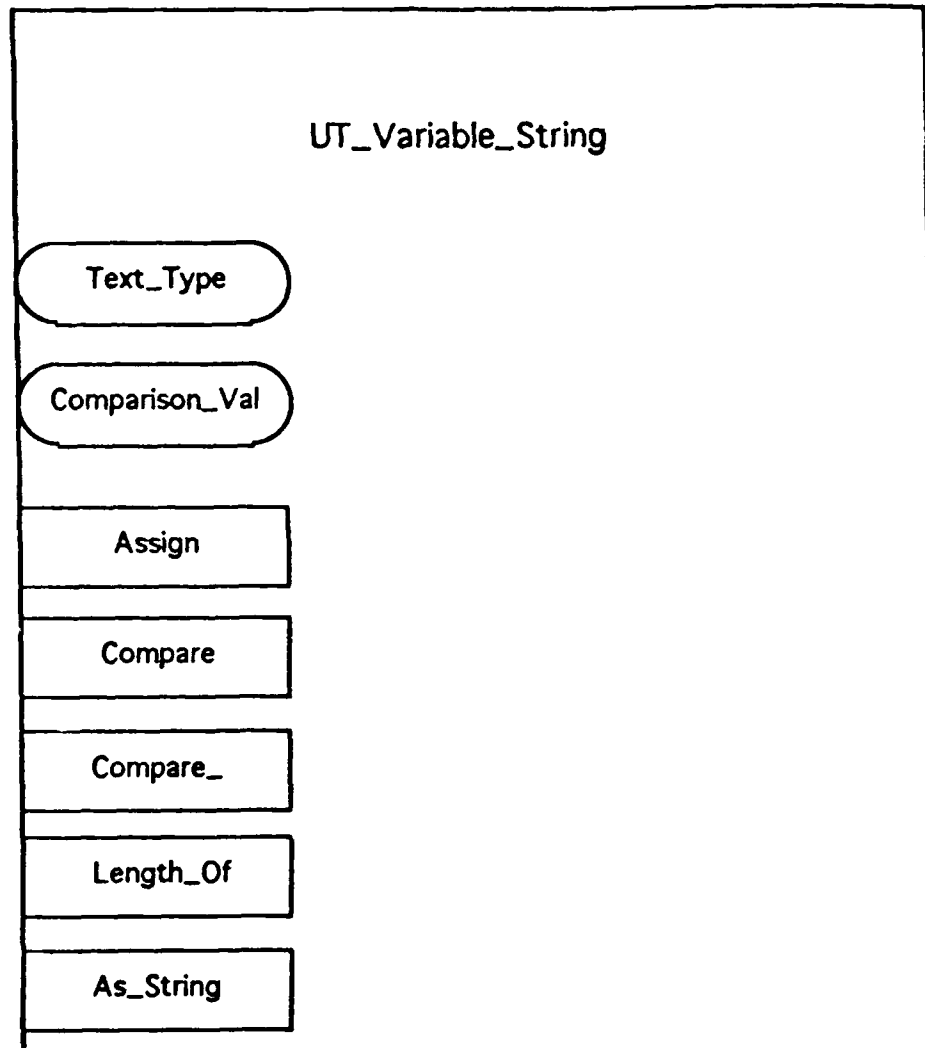
10.3 Asynchronous Input



10.4 UT Bit Manipulations



10.5 UT Variable String



APPENDIX C

10 PROGRAM DEPENDANCY CHART FOR MONITOR APPLICATION

