p. 18

# FINAL REPORT

**Louis A. DeAcetis**

**Bronx Community College/CUNY**

**January 1991**

(NASA-CR-[illegible]) FINAL REPORT (Houston Univ.) 1[illegible] p CSCL 09[illegible] N91-25[illegible]7

Unclas
G3/61 0020010

**Cooperative Agreement NCC 9-16**
**Research Activity No. AI.1**

**NASA Johnson Space Center**
**Engineering Directorate**
**Tracking and Communications Division**

ricis

*Research Institute for Computing and Information Systems*
*University of Houston - Clear Lake*

*T·E·C·H·N·I·C·A·L R·E·P·O·R·T*

## *The RICIS Concept*

The University of Houston-Clear Lake established the Research Institute for Computing and Information systems in 1986 to encourage NASA Johnson Space Center and local industry to actively support research in the computing and information sciences. As part of this endeavor, UH-Clear Lake proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a three-year cooperative agreement with UH-Clear Lake beginning in May, 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The mission of RICIS is to conduct, coordinate and disseminate research on computing and information systems among researchers, sponsors and users from UH-Clear Lake, NASA/JSC, and other research organizations. Within UH-Clear Lake, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business, Education, Human Sciences and Humanities, and Natural and Applied Sciences.

Other research organizations are involved via the "gateway" concept. UH-Clear Lake establishes relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research.

A major role of RICIS is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. Working jointly with NASA/JSC, RICIS advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research, and integrates technical results into the cooperative goals of UH-Clear Lake and NASA/JSC.

# FINAL REPORT

**Louis A. DeAcetis**

**Bronx Community College/CUNY**

January 1991

Cooperative Agreement NCC 9-16
Research Activity No. AI.1

NASA Johnson Space Center
Engineering Directorate
Tracking and Communications Division

*Research Institute for Computing and Information Systems*
*University of Houston - Clear Lake*

*T·E·C·H·N·I·C·A·L R·E·P·O·R·T*

## Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Dr. Louis A. DeAcetis of Bronx Community College, City University of New York. Dr. T. F. Leibfried served as RICIS research representative.

Funding has been provided by the Engineering Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA technical monitor for this activity was Oron Schmidt, of the Communications Performance and Integration Branch, Tracking and Communications Division, Engineering Directorate, NASA/JSC.

# FINAL REPORT

Subcontract No. 078
RICIS Research Activity AI.1
(NASA Cooperative Agreement NCC 9-16)

Louis A. DeAcetis, Ph.D.
Bronx Community College/CUNY
Bronx NY 10453

January 1991

## 1. Introduction

This is a report of the activities of Dr. Louis A. DeAcetis of Bronx Community College who served as the "Key Personnel" under a sub-contract between the University of Houston/Clear Lake and Bronx Community College/CUNY. The period of this contract extended from September 1, 1990 to January 15, 1991. The principal work site was NASA/Johnson Space Center, Communications and Tracking Division (Building 44), Houston TX 77058.

The scope of the work included the areas of simulators and simulation, data acquisition and transmission, and evaluation of Ada compilers and programming environments. In addition, Dr. DeAcetis was to serve as a "tester/Test Manager" on the Software Support Environment (SSE) as needed. (The SSE was not used, and therefore no time was devoted to the latter activity).

An oral Presentation of this work was given on January 14, 1991 at JSC to both UH/CL and NASA personnel. This report also constitutes a summary/review of that Presentation.

## 2. Activity Summary

### A. Bus 1553B Software Development

As part of simulator development, it was determined that some experience with the MIL-STD-1553 bus would be useful since this will be the primary network for data acquisition from, and command transmission to remote equipment on Space Station Freedom. Interface cards for use in IBM-PC compatibles were available from ILC Data Device Corporation along with driver software written in Microsoft C. A major aspect of the work was the development of programs in Ada which would interface with the driver programs written in C that were supplied by the manufacturer. Appendix A includes the Ada specifications for a first cut at this software interface. Although the bodies of these packages have not been exhaustively tested and debugged as yet, the functions and procedures indicated should serve as a basis for software development in Ada which will interface with these cards and the 1553B bus. In addition, simulations of orbital replacement unit (ORU) equipment can use the programs to send/receive information over the 1553B bus, thereby more closely simulating conditions on the space station.

### B. Simulator Development

A simulation of the space-to-ground communications subsystem that has been used in previous testbed activities was revised to include generic elements. Appendix B contains the specifications and bodies of Ada packages that were developed to create both an Ada Task implementation of a simulator, and a Procedure (non-Task) version. Examples of the use of these algorithms were presented at an oral presentation given on January 14, 1991 at JSC, and in a recent publication: "Using ADA tasks to simulate operating equipment", Louis A. DeAcetis, Oron Schmidt, and Kumar Krishen, _Computers in Physics_, September/October 1990.

C. Although no new Ada compilers and programming tools were presented for evaluation during the tenure of the work, continued work with both the Alsys and Meridian compilers under DOS further confirmed the opinion that the current Alsys Ada environment is superior to the Meridian AdaVantage environment. (In particular, the space-to-ground simulator proved too large for the Meridian compiler, even after repeated attempts to make it fit.)

## Appendix A

Ada programs to interface with C-software which drives PC-based interface cards (from ILC Data Device Corporation) for the MIL-STD-1553B bus. Three Package specifications are included. Package FIFTEEN_53B has routines common to emulations of both a remote terminal (RT) and a bus controller (BC). Packages FIFTEEN_53B_RT_PROGRAMS and FIFTEEN_53B_BC_PROGRAMS have routines specific to RT emulation and BC emulation respectively.

```
-- Package  FIFTEEN_53B Specifications ---------------------------------------
------------------------------------------------------------------------------
--   Last Update: 9:19:10  1/12/1991  LAD                                   --
--                                                                          --
-- This is a package of Ada Routines that interface with C-programs         --
--   supplied with the DDC 1553B PC interface cards ( BUS-65517 )           --
--                                                                          --
-- These routines are common to both a Remote Terminal and Bus Controller  --
--   emulation.                                                             --
--                                                                          --
-- See Package FIFTEEN_53B_RT_PROGRAMS, and                                 --
--     Package FIFTEEN_53B_BC_PROGRAMS                                      --
-- for Packages that are specific to Remote Terminals and Bus Controller    --
-- respectively, and which use this package.                                --
------------------------------------------------------------------------------
--- References:                                                             --
--   1. BUS-69008 MICROSOFT C MIL-STD-1553B DRIVER
--      ----------------------------------------
--      RELEASE 1.52
--      ILC DATA DEVICE CORPORATION, July 27, 1989
--
--   2. MIL-STD-1553 Designer's Guide
--      -----------------------------
--      Second Edition, Second Printing
--      ILC DATA DEVICE CORPORATION, 1988
--
-- IBM-XT 1553B Interface card:
--
--   BUS-65517 Interface Card
--   ILC DATA DEVICE CORPORATION
--   105 Wilbur Place,
--   Bohemia NY 11716
--   (516) 567-5600
--   (516) 563-5234 [Data Bus Applications Dept.]
------------------------------------------------
-- Louis A. DeAcetis, Ph.D.                   --
-- Tracking and Communications Division, EE7  --
-- NASA/Johnson Space Center                  --
-- Houston TX 77058                           --
------------------------------------------------
```

```
with system;

package FIFTEEN_53B is
  -----
  subtype WORD_COUNT_TYPE is integer range 1..32;
  --
  subtype SHARED_MEMORY_TABLE_SIZE_TYPE is integer range 0..32;
  --                                              -- 0 => Table not in use
  subtype ID_TYPE is integer range 0..251;
  --
  subtype ADDRESS_TYPE is integer range 0..31; -- 0 not a valid address;
  --                                           --   used as a flag
  type IO_BUFFERING_TYPE is (SINGLE,DOUBLE);
  --
  type BUS_TYPE is ( BUS_A, BUS_B );
  --
  type RECEIVE_TRANSMIT_TYPE is (RECEIVE,TRANSMIT);
  --
  type DRIVER_CONFIG_TYPE is
  record
    MEMORY_SEGMENT : character;
    BCRT_CODE_FILE : string(1..9);
    MON_CODE_FILE  : string(1..8);
    IMRLIB_FILE    : string(1..11);
    IDEA_SYM_FILE  : string(1..9);
    INTERRUPT_REQU : character;
    STANDARD_1553  : character;
    CARD_VERSION   : character;
  end record;
  -----
  -----
  DEFAULT_SHARED_MEMORY_TABLE_SIZE : SHARED_MEMORY_TABLE_SIZE_TYPE := 32;
  ------------------------------------
  ------------------------------------
  function DDC_VERSIONS return string;
    ----------------------------------------------------------------------
    ----- Returns a string which includes the Versions of the DDC card, -
    -----   DDC software, and the SN of the DDC card, separated by "/". -
    ----------------------------------------------------------------------
  ------------------------------------
  function DEFINE_TABLE_SIZE( TABLE : ID_TYPE;
                              SIZE : WORD_COUNT_TYPE ) return integer;
    -----------------------------------------------------------------------
    --- Define size of Shared Memory data table. Default size is equal to -
    ---   DEFAULT_SHARED_MEMORY_TABLE_SIZE defined in the Specifications  -
    ---   of this Package. The function DEFINE_DEFAULT_CHARACTERISTICS    -
    ---   (which must be called before DEFINE_TABLE_SIZE) sets the default-
    ---   value. If default size is OK, this function need not be called. -
    --- If this function is used, it will change the pointers to all      -
    ---   tables with higher values. The contents of TABLE and all of the -
    ---   higher tables must therefore be redefined.                      -
    -----------------------------------------------------------------------
```

```
-------------------------------------
procedure END_INITIALIZATION;
-------------------------------------
procedure HALT_ALL;
  ------------------------------------------------------------------
  ----- Place Bus Controller and/or Remote Terminal Emulations in the -
  -----   Halt state, and Bus Monitor as well.                       -
  ------------------------------------------------------------------
-------------------------------------
procedure HALT_BCRT;
  ------------------------------------------------------------------
  ----- Place Bus Controller and/or Remote Terminal Emulations in the -
  -----   Halt state                                                 -
  ------------------------------------------------------------------
-------------------------------------
procedure RESET_CARD( CARD_NUMBER : integer := 0;
                    DRIVER_CONFIG : DRIVER_CONFIG_TYPE :=
                              ( MEMORY_SEGMENT => 'D',
                                BCRT_CODE_FILE => "bcrt.bin"  &ascii.nul,
                                MON_CODE_FILE  => "mon.bin"   &ascii.nul,
                                IMRLIB_FILE    => "imrlib.bin"&ascii.nul,
                                IDEA_SYM_FILE  => "idea.sym"  &ascii.nul,
                                INTERRUPT_REQU => '2',
                                STANDARD_1553  => 'B',
                                CARD_VERSION   => '1' ) );

--NOTE-- This procedure is not fully implemented as yet using all of the
--       above parameters. Use the default form, RESET_CARD, which will
--       invoke the program "reset_idea" and accomplish the reset that way.
--       ALL of the above default parameters are invoked when this default
--       form is used.
--
--       The C-program "reset_idea" requires the files:
--          bcrt.bin, mon.bin, imrlib.bin, idea.sym.
--       These names are hardcoded into "new_reset_idea". The parameters
--       INTERRUPT_REQU and MEMORY_SEGMENT are supplied in file "idea.cfg"
--       which also must be present in the directory from which driver
--       program is run.
-------------------------------------
procedure START_INITIALIZATION( BUFFERING : IO_BUFFERING_TYPE := DOUBLE );
-------------------------------------
function WR_1553_DATA( TABLID : ID_TYPE;
                          MSG : system.address;
                       WD_CNT : WORD_COUNT_TYPE;
                          POS : WORD_COUNT_TYPE ) return integer;
  --This function is generally not accessed directly by the user.
  --  It is made visible in case it may be needed. It is used by
  --  programs in files P1553BBC.ADB and P1553BRT.ADB
  --  ( Packages FIFTEEN_53B_BC_PROGRAMS and FIFTEEN_53B_RT_PROGRAMS )
  pragma INTERFACE( C, WR_1553_DATA );
  pragma INTERFACE_NAME( WR_1553_DATA, "_new_write_data");
  --pragma INTERFACE_NAME( WR_1553_DATA, "_write_data");
-------------------------------------
-------------------------------------
End FIFTEEN_53B;
*****************************************************************************
```

```
-- Package  FIFTEEN_53B_BC_PROGRAMS  Specifications ----------------------
---------------------------------------------------------------------------
-- P1553BBC.ADB  Last Update:  9:14:51  1/15/1991     LAD                 -
--                                                                        -
-- This is a Package of Ada Routines that interface with C-programs      --
--   supplied with the DDC 1553B PC interface cards ( BUS-65517 ), and    -
--   additions thereto. C-program sources are in file 1553b_io.c          -
--                                                                       --
-- Routines common to both a Remote Terminal (RT) and Bus Controller (BC) --
--   are in Package FIFTEEN_53B (files P1553B.ADS/B).                     -
--                                                                        -
-- This particular Package contains routines for BC definition/emulation. --
---------------------------------------------------------------------------
-- References:                                                            -
--
--   1. BUS-69008 MICROSOFT C MIL-STD-1553B DRIVER
--      ---------------------------------------
--      RELEASE 1.52                                                      -
--      ILC DATA DEVICE CORPORATION, July 27, 1989
--
--   2. MIL-STD-1553 Designer's Guide                                     -
--      --------------------------
--      Second Edition, Second Printing
--      ILC DATA DEVICE CORPORATION, 1988
--                                                                        -
-- IBM-XT 1553B Interface card:
--
--   BUS-65517 Interface Card                                             -
--   ILC DATA DEVICE CORPORATION
--   105 Wilbur Place,
--   Bohemia NY 11716
--   (516) 567-5600                                                       -
--   (516) 563-5234 [Data Bus Applications Dept.]
--
-------------------------------------------------------------------------  -
-----------------------------------------------
-- Louis A. DeAcetis, Ph.D.                  --
-- Tracking and Communications Division, EE7 --                           -
-- NASA/Johnson Space Center                 --
-- Houston TX 77058                          --
-----------------------------------------------
```

```
with FIFTEEN_53B;

generic

  type WORD_TYPE is (<>);       -- Type of data transmitted over 1553B bus;
                               --   Limited to discrete for now;
package FIFTEEN_53B_BC_PROGRAMS is
  ------------------------------------
  ---
  BC_RECEIVE_TABLE_NO  : FIFTEEN_53B.ID_TYPE := 1;

  BC_TRANSMIT_TABLE_NO : FIFTEEN_53B.ID_TYPE := 1;
  ---
  type IO_DATA_BUFFER_TYPE is array( FIFTEEN_53B.WORD_COUNT_TYPE range <>)
                                                            of WORD_TYPE;
  ------------------------------------
  procedure DEFINE_BC_DEFAULT_CHARACTERISTICS(
                  BC_ADDRESS : FIFTEEN_53B.ADDRESS_TYPE;
                    TABLE_ID : FIFTEEN_53B.ID_TYPE := 1;
                  TABLE_SIZE : FIFTEEN_53B.SHARED_MEMORY_TABLE_SIZE_TYPE :=
                               FIFTEEN_53B.DEFAULT_SHARED_MEMORY_TABLE_SIZE;
           MINOR_FRAME_TIME : long_integer := 16#1000# );
  ------------------------------------
  procedure DEFINE_MESSAGE(
                  MESSAGE_ID : integer                 := 1; -- 1..255; See P.30
                   COMM_TYPE : FIFTEEN_53B.RECEIVE_TRANSMIT_TYPE--Not complete
                               := FIFTEEN_53B.TRANSMIT; -- set of choices yet
              DATA_TABLE_NO : FIFTEEN_53B.ID_TYPE      := 1;
                        BUS : FIFTEEN_53B.BUS_TYPE     := FIFTEEN_53B.BUS_A;
                      TADDR : FIFTEEN_53B.ADDRESS_TYPE := 1;
                    SUBADDR : FIFTEEN_53B.ADDRESS_TYPE := 1;
                     TR_RCV : FIFTEEN_53B.RECEIVE_TRANSMIT_TYPE
                              := FIFTEEN_53B.TRANSMIT;
                 WORD_COUNT : FIFTEEN_53B.WORD_COUNT_TYPE; -- := 1; default?
        TIME_TO_NEXT_MESSAGE : long_integer := 16#100#; --Was integer 1/4/91
              INJECT_ERRORS : integer := 0 );
  ------------------------------------
  function INPUT_DATA_ARRAY_FROM(
                              RT_ADDRESS : FIFTEEN_53B.ADDRESS_TYPE;
                              SUBADDRESS : FIFTEEN_53B.ADDRESS_TYPE;
                         NUMBER_OF_WORDS : FIFTEEN_53B.WORD_COUNT_TYPE
                                           := FIFTEEN_53B.WORD_COUNT_TYPE'last;
                                TABLE_ID : FIFTEEN_53B.ID_TYPE
                                           := BC_RECEIVE_TABLE_NO;
                          FRAME_POSITION : integer      := 1;
                         NUMBER_OF_TIMES : long_integer := 1;
                              MESSAGE_ID : integer      := 1 )
                                                return IO_DATA_BUFFER_TYPE;
          -----------------------------------------------------------------
          --NOTE: Procedure below returns STATUS; this Function does not--
          -----------------------------------------------------------------
```

```
      -------------------------------------
      procedure INPUT_DATA_ARRAY_FROM(
                              RT_ADDRESS : in FIFTEEN_53B.ADDRESS_TYPE;
                              SUBADDRESS : in FIFTEEN_53B.ADDRESS_TYPE;
                         NUMBER_OF_WORDS : in FIFTEEN_53B.WORD_COUNT_TYPE :=
                                              FIFTEEN_53B.WORD_COUNT_TYPE'last;
                            INPUT_BUFFER : in out IO_DATA_BUFFER_TYPE;
                                TABLE_ID : in FIFTEEN_53B.ID_TYPE
                                              := BC_RECEIVE_TABLE_NO;
                          FRAME_POSITION : in integer      := 1;
                         NUMBER_OF_TIMES : in long_integer := 1;
                              MESSAGE_ID : in integer      := 1;
                                  STATUS : in out integer );
              --------------------------------------------------------------
              --NOTE: This Procedure returns STATUS; Function above does not--
              --------------------------------------------------------------
      -------------------------------------
      function RUN_BC( MESSAGE_POSITION : integer      := 1;
                        NUMBER_OF_TIMES : long_integer := 1 ) return integer;
      -------------------------------------
      function TRANSMIT_DATA_ARRAY_TO(
                            RT_ADDRESS : FIFTEEN_53B.ADDRESS_TYPE;
                            SUBADDRESS : FIFTEEN_53B.ADDRESS_TYPE;
                               MESSAGE : IO_DATA_BUFFER_TYPE;
                       NUMBER_OF_WORDS : FIFTEEN_53B.WORD_COUNT_TYPE;
                       FROM_DATA_TABLE : FIFTEEN_53B.ID_TYPE
                                         := BC_TRANSMIT_TABLE_NO;
                   DATA_TABLE_POSITION : FIFTEEN_53B.WORD_COUNT_TYPE := 1)
                                                         return boolean;
                                    -- Use DATA_TABLE_POSITION := 1 for now
      -------------------------------------
      procedure WRITE_1553B_DATA( TABLEID : FIFTEEN_53B.ID_TYPE;
                                  MESSAGE : IO_DATA_BUFFER_TYPE;
                               WORD_COUNT : FIFTEEN_53B.WORD_COUNT_TYPE;
                                 POSITION : FIFTEEN_53B.WORD_COUNT_TYPE );
     -------------------------------------

     end FIFTEEN_53B_BC_PROGRAMS;
     *************************************************************************
```

```
-- Package  FIFTEEN_53B_RT_PROGRAMS  Specifications ------------------------
----------------------------------------------------------------------------
-- P1553BRT.ADB  Last Update:   9:20:03  1/12/1991    LAD                --
--                                                                       --
-- This is a package of Ada Routines that interface with C-programs      --
--   supplied with the DDC 1553B PC interface cards ( BUS-65517 )        --
--                                                                        --
-- Routines common to both a Remote Terminal and Bus Controller are in   --
--   Package FIFTEEN_53B (files P1553B.ADS/B).                           --
--                                                                       --
-- This particular Package contains routines for RT definition/emulation. --
----------------------------------------------------------------------------
-- References:
--
--   1. BUS-69008 MICROSOFT C MIL-STD-1553B DRIVER
--      ------------------------------------------
--      RELEASE 1.52
--      ILC DATA DEVICE CORPORATION, July 27, 1989
--
--   2. MIL-STD-1553 Designer's Guide
--      ----------------------------
--      Second Edition, Second Printing
--      ILC DATA DEVICE CORPORATION, 1988
--
-- IBM-XT 1553B Interface card:
--
--   BUS-65517 Interface Card
--   ILC DATA DEVICE CORPORATION
--   105 Wilbur Place,
--   Bohemia NY 11716
--   (516) 567-5600
--   (516) 563-5234 [Data Bus Applications Dept.]
--
------------------------------------------------
-- Louis A. DeAcetis, Ph.D.                   --
-- Tracking and Communications Division, EE7  --
-- NASA/Johnson Space Center                  --
-- Houston TX 77058                           --
------------------------------------------------
```

```
with FIFTEEN_53B;
generic
  type WORD_TYPE is (<>);       -- Type of data transmitted over 1553B bus;
                                --   Limited to discrete for now.
package FIFTEEN_53B_RT_PROGRAMS is
  ---
  type ON_OFF_TYPE is ( ON, OFF );
  ---
  type IO_DATA_BUFFER_TYPE is array( FIFTEEN_53B.WORD_COUNT_TYPE range <> )
                                                          of WORD_TYPE;
  ----------------------------------------------------------------------
  procedure CLEAR_AND_RUN_RTs( RT_ADDRESS: FIFTEEN_53B.ADDRESS_TYPE := 0 );
  ------------------------------------
  function DEFINE_RT_DATA_TABLE(
                          RT_ADDRESS  : FIFTEEN_53B.ADDRESS_TYPE;
                          SUBADDRESS  : FIFTEEN_53B.ADDRESS_TYPE;
                    RT_COM_DIRECTION  : FIFTEEN_53B.RECEIVE_TRANSMIT_TYPE;
                            TABLE_ID  : FIFTEEN_53B.ID_TYPE := 0 )
                                         return FIFTEEN_53B.ID_TYPE;

  ------------------------------------
  procedure DEFINE_RT_DEFAULT_CHARACTERISTICS;
  ------------------------------------
  function RECEIVE_DATA_FROM_BC(
                            USING_TableId : FIFTEEN_53B.ID_TYPE;
                          NUMBER_OF_WORDS : FIFTEEN_53B.WORD_COUNT_TYPE )
                                         return IO_DATA_BUFFER_TYPE;

  ------------------------------------
  function RECEIVE_DATA_FROM_BC( AT_RT_ADDRESS  : FIFTEEN_53B.ADDRESS_TYPE;
                                    SUBADDRESS  : FIFTEEN_53B.ADDRESS_TYPE;
                               NUMBER_OF_WORDS  : FIFTEEN_53B.WORD_COUNT_TYPE
                                         return IO_DATA_BUFFER_TYPE;

  ------------------------------------
  function RUN_RTs return integer;
  ------------------------------------
  procedure SET_BUSY_STATE( RT_ADDRESS  : FIFTEEN_53B.ADDRESS_TYPE;
                                ON_OFF  : ON_OFF_TYPE := ON );
  ------------------------------------
  function TRANSMIT_FROM( RT_ADDRESS  : FIFTEEN_53B.ADDRESS_TYPE;
                          SUBADDRESS  : FIFTEEN_53B.ADDRESS_TYPE;
                          MESSAGE     : IO_DATA_BUFFER_TYPE;
                          COUNT       : FIFTEEN_53B.WORD_COUNT_TYPE;
                          POSITION    : FIFTEEN_53B.WORD_COUNT_TYPE := 1 )
                                                      return boolean;

  ------------------------------------
  function TRANSMIT_FROM( ID  : FIFTEEN_53B.ID_TYPE;
                     MESSAGE  : IO_DATA_BUFFER_TYPE;
                       COUNT  : FIFTEEN_53B.WORD_COUNT_TYPE;
                    POSITION  : FIFTEEN_53B.WORD_COUNT_TYPE := 1 )
                                                      return boolean;

  ------------------------------------
  procedure WRITE_1553B_DATA( TABLEID  : FIFTEEN_53B.ID_TYPE;
                              MESSAGE  : IO_DATA_BUFFER_TYPE;
                           WORD_COUNT  : FIFTEEN_53B.WORD_COUNT_TYPE;
                             POSITION  : FIFTEEN_53B.WORD_COUNT_TYPE );
------------------------------------
end FIFTEEN_53B_RT_PROGRAMS;
*************************************************************************
```

Generic Ada Packages for use in simulation development. Two versions are included, one to implement a simulator with equipment running as independent tasks, and a second one for equipment running as called procedures.

TASK Version Package:

```
--Package Generic_Equipment
--  TASK Version
--  Last update: 11-02-89   LAD
----------
--This package has Ada procedures as formal parameters and therefore
--must be instantiated with procedures which implement the transfer
--functions of the actual equipment used.
--In Particular:
--  Procedure Set_OFF_Values        : Parameter values for equip. OFF;
--  Procedure Set_INITIAL_ON_Values:  " Values for equipment just turned ON;
--  Procedure Set_Running_Values    :  " Values for equipment ON & running
---------------------------------------------------------------------------
generic
      with procedure Set_OFF_Values;
      with procedure Set_INITIAL_ON_Values;
      with procedure Set_Running_Values;

package GENERIC_EQUIPMENT is
  ---
  task SWITCH_CONTROL is
    entry CLOSE_SWITCH;
    entry OPEN_SWITCH;
  end SWITCH_CONTROL;
  ---
  procedure DESTROY;
  ---
end GENERIC_EQUIPMENT;
```

```
-------------------------------------
package body GENERIC_EQUIPMENT is
  --
  SWITCH_IS_OPEN : boolean := true; -- Switch starts open (equip. is OFF)
  ----------
  task body SWITCH_CONTROL is
   begin
     SWITCH_OPEN_OFF:
     loop
       SWITCH_CLOSED_ON:
       loop
         SWITCH_CONTROL:  --*** Switch Control Loop ***--
         loop
           if SWITCH_IS_OPEN      --Select when SWITCH_IS_OPEN:
            then                  -- Wait for rendezvous to close it
              select
                accept OPEN_SWITCH;      --Accept and ignore OPEN requests
               or
                accept CLOSE_SWITCH;     --Where Switch is actually Closed
                SWITCH_IS_OPEN := false;
                Set_INITIAL_ON_Values; --
                exit SWITCH_CONTROL;
              end select;
            else
              select
                accept CLOSE_SWITCH;     --Accept and ignore CLOSE requests
                exit SWITCH_CONTROL;
               or
                accept OPEN_SWITCH;      --Where Switch is actually Opened
                SWITCH_IS_OPEN := true;
                exit SWITCH_CLOSED_ON;
               else
                exit SWITCH_CONTROL;
              end select;
           end if;
         end loop SWITCH_CONTROL; --*** End Switch Control Loop ***--
       --Should only get here if Switch is Closed/ON:
         Set_Running_Values; --
         delay 0.001;  --delay/Reschedule;
         --
       end loop SWITCH_CLOSED_ON;
     --
     --Should only get here if Switch was just opened:
       Set_OFF_Values; --
       delay 0.001;  --delay/Reschedule;
       --
     end loop SWITCH_OPEN_OFF;
   end SWITCH_CONTROL;
  ---------
  procedure DESTROY is    --Command to abort task (for orderly shutdown)
    begin
      abort SWITCH_CONTROL;
    end DESTROY;
  ---------
end GENERIC_EQUIPMENT;
*************************************************************************
```

PROCEDURE Version Package:

```
-- Package Generic_Equipment
--   Procedure Version
--   Last Update: 9:28:05  10/16/1990  LAD
------------
-- This Package has Ada Procedures as formal parameters and therefore
--   must be instantiated with procedures which implement the transfer
--   functions of the actual equipment used. In particular:
--    procedure Set_OFF_Values        : Parameter values for equipment OFF
--    procedure Set_INITIAL_ON_Values:  " values for equipment just turned ON
--    procedure Set_RUNNING_Values    :  " values for equipment ON & running
---------------------------------------------------------------------------
generic
  with procedure Set_OFF_Values;
  with procedure Set_INITIAL_ON_Values;
  with procedure Set_Running_Values;

package Generic_Equipment is
   ---
  type ON_OFF_RUN_TYPE is (ON, OFF, RUN);
  procedure SWITCH(CONTROL : in ON_OFF_RUN_TYPE);
  ---
end Generic_Equipment;
----------------------------------------
package body GENERIC_EQUIPMENT is
 --
 SWITCH_IS_OPEN : boolean := TRUE;  -- Switch starts OPEN (equip. is OFF)
 --
 procedure SWITCH(CONTROL : in ON_OFF_RUN_TYPE) is
 begin
    case CONTROL is
      when ON  =>
        if SWITCH_IS_OPEN = true then   --Should only get here if switch
          SWITCH_IS_OPEN := false;      -- was just turned ON (closed)
          -------------------------     --
          Set_INITIAL_ON_Values; --     --
          -------------------------     --
          return;                       --Return after setting initial values
        end if;
      when OFF => SWITCH_IS_OPEN := true;
      when RUN => null;
    end case;
    if not SWITCH_IS_OPEN then  --Should only get here if switch is Closed/ON
      ----------------------
      Set_RUNNING_Values; --
      ----------------------
    else                        --Should only get here if Switch is Open/OFF
      ------------------
      Set_OFF_Values; --
      ------------------
    end if;
 end SWITCH;
 -----
 end Generic_Equipment;
```