Improving Search Times when Resolving External Symbols in the Timeliner System

by

Isaac Charny

S.B. EECS, M.I.T., 2008

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May, 2009

Author_____
Department of Electrical Engineering and Computer Science
May 22, 2009

Certified by_____
Dr. Robert Brown
VI-A Company Thesis Supervisor

Certified by_____
Professor Robert Berwick
M.I.T. Thesis Supervisor

Accepted by_____
Arthur C. Smith
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

[This page intentionally left blank]

Improving Search Times when Resolving External Symbols
in the Timeliner System
by
Isaac Charny

Submitted to the
Department of Electrical Engineering and Computer Science

May 22, 2009

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The Timeliner System, developed at The Charles Stark Draper Laboratory, is a tool to automate operational procedures. Using the Timeliner language, a user can easily write scripts which control complex systems. In compiling these scripts into executable data files that can then be executed by the Timeliner executor, the Timeliner compiler resolves external symbols using information stored in a target system description database (GDB). This resolution effectively binds the external symbols to commands and objects of the target system. The GDB was implemented as a group of binary trees. However, search times to resolve external symbols in the trees do not scale well as the number of symbols increases. By replacing the binary trees with hash tables, time to resolve symbols is significantly reduced.

Thesis Supervisor: Dr. Robert Brown
Title: Distinguished Member of the Technical Staff, Charles Stark Draper Laboratory, Inc.

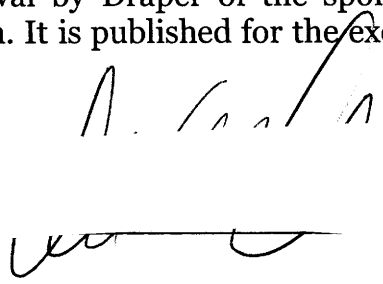
Thesis Advisor: Professor Robert Berwick
Title: Professor of Electrical Engineering & Computer Science

[This page intentionally left blank]

ACKNOWLEDGEMENT
May 22, 2009

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under NASA-JSC contract NNJ06HC37C.

Publication of this thesis does not constitute approval by Draper of the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.



[This page intentionally left blank]

Table of Contents

Acknowledgments

I would like to thank Dr. Robert Brown for helping me with this project and keeping me focused and on task, and Prof. Robert Berwick for advising me for the past four years at MIT.

Introduction/System Overview

Timeliner has been developed as a tool to automate procedural tasks. These tasks may be sequential tasks that would typically be performed by a human operator, or precisely ordered sequencing tasks that allow autonomous execution of a control process. The Timeliner system is both a specialized computer language and an execution environment. The Timeliner system includes elements for compiling and executing sequences that are defined in the Timeliner language. The Timeliner language was specifically designed to allow easy definition of scripts that provide sequencing and process control of complex systems. The execution environment provides real-time monitoring and control based on the commands and conditions defined in the Timeliner language. The Timeliner sequence control may be preprogrammed, compiled from Timeliner "scripts," or it may consist of real-time, interactive inputs from system operators.

Historically, Timeliner was created to emulate the timelines for onboard crew procedures followed by the crew of the Space Shuttle. It was used as a simulation driver in tests of the Space Shuttle system, mimicking crew actions in monitoring and controlling the spacecraft systems. This version of Timeliner has been in use since 1982.

The Timeliner simulation system was extended for use in other applications, and was tailored to provide realtime sequence execution and support interactive control commands that might be entered by the systems engineer: for example, sequence start and stop.

In 1992, Timeliner was selected by NASA as the User Interface Language (UIL) for Space Station Freedom, and later for International Space Station (ISS) to be executed on the Space Station's real-time core command and control and payload control computers. Since that time, Timeliner has evolved as a modular, extensible system that allows scripts to be developed and executed in virtually any systems environment, can be applied to control a variety of target systems, and meet a wide range of mission objectives. It is currently in use in several different instances on the Space Station for U.S. core operations, U.S. payload operations, and Japanese core and payload operations. In addition, an extended version of the real-time ISS Timeliner system is to be used throughout the U.S. by several academic institutions and NASA centers as a UNIX-based, generalpurpose, procedure executor for payload development, simulation, and test environments for payloads being developed for ISS.

In general, the Timeliner system lowers the workload in the performance of mission or process control operations. In a mission environment, scripts can be used to automate spacecraft or aircraft operations including autonomous or interactive vehicle control, performance of preflight and post-flight subsystem checkouts, or handling of failure detection and recovery. Timeliner may also be used for mission payload operations, such as stepping through pre-defined procedures of a scientific experiment. Other applications may include process control for manufacturing, materials processing, robotics, or any automated procedures that need to be clearly defined by a systems

specialists (rather than software engineers) and need to be executed reliably, insuring repeatability.

For the International Space Station, these aspects will provide significant mission cost savings and productivity gains. Since Timeliner allows multiple, parallel automated sequencing, multiple operations and experiments can occur simultaneously, utilizing the Space Station crew members for only the 'human required' activities (e.g. replacing filters, exchanging components, observing visually successful or failed events, etc.) Also, since much of the sequencing is provided autonomously, the crew and ground operations training required is greatly reduced providing a significant savings in mission cost. As the International Space Station is used after the first few years, many still yet to be defined operations will be developed. Significant cost savings is realized again for training for these newly defined activities. Also, costs for procedure development and execution are reduced by several orders of magnitude since Timeliner scripts are defined 'independent' of the core Ada software builds, providing no impact on core system software development, testing, and integration.

In addition to savings and productivity gains, the use of the Timeliner provides significant improvements to mission success, reliability, and safety since human sequencing errors are eliminated (e.g. steps executed out of order, duplicated, or omitted, incorrect command parameters, improper evaluation of system data, etc.) [1]. Figure 1 shows a system overview.

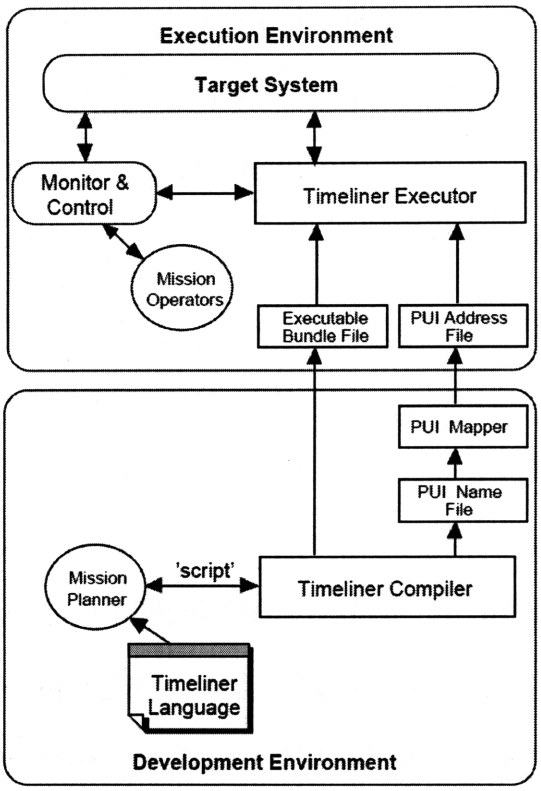


Figure 1: System overview of the Timeliner System [2].

Previous Work

The Timeliner components were developed by Draper under contract for NASA. Past theses examined ways to enhance the Timeliner product. Frank Tien-Fu Liu developed a Java version of the Timeliner compiler in 2001. Using Java, Liu re-engineered the compiler to augment its extensibility and make it easier to maintain. Liu's work modified the compiler to generate an abstract syntax tree represented in XML. Steven Stern, in 2005, developed a version of the Timeliner executor that processed the abstract syntax tree generated by Liu's version of the compiler. Maya Dobuzhskaya developed a Timeliner Integrated Development Environment based on

Eclipse for her 2005 Master of Engineering Thesis. In his 2006 thesis, Daniel Swanton modified the Timeliner language to allow specification of preconditions and postconditions and demonstrated the integration of the Timeliner system with an autonomous planner.

My Work

Much of the Timeliner system's versatility is achieved by separating the specification of external symbols from the specification of the procedural logic in Timeliner scripts. External symbols such as commands to the target system and variables representing state data of the target system are defined in a target interface database, or ground database (GDB). In compiling Timeliner scripts, the compiler resolves external symbols using information stored in the GDB. This resolution effectively binds the external symbols to commands and objects of the target system.

Under the supervision of Dr. Robert Brown, my work investigated ways to decrease the time spent searching the GDB to resolve external symbols. It began by examining and characterizing the existing algorithm used by Timeliner. Next, it surveyed other search algorithms and evaluated their implementations in the Timeliner system. It concluded with a recommendation and implementation of a search algorithm which improves on the existing algorithm and scales up to handle GDBs containing thousands of entries.

Ground Database

The GDB is a database of data definitions for Commands and Variables (data references) used by Timeliner scripts. It is defined through input files which are simple to maintain and change. The GDB provides access routines which are used during Timeliner script compilation. These routines access the database to return the information pertaining to the database objects. The goal is to generate Timeliner data files through an easy-to-maintain data definition, which allows the Timeliner compiler to be updated with new data simply by changing the data input files and recreating the GDB. Figure 2 shows an overview of the GDB system.

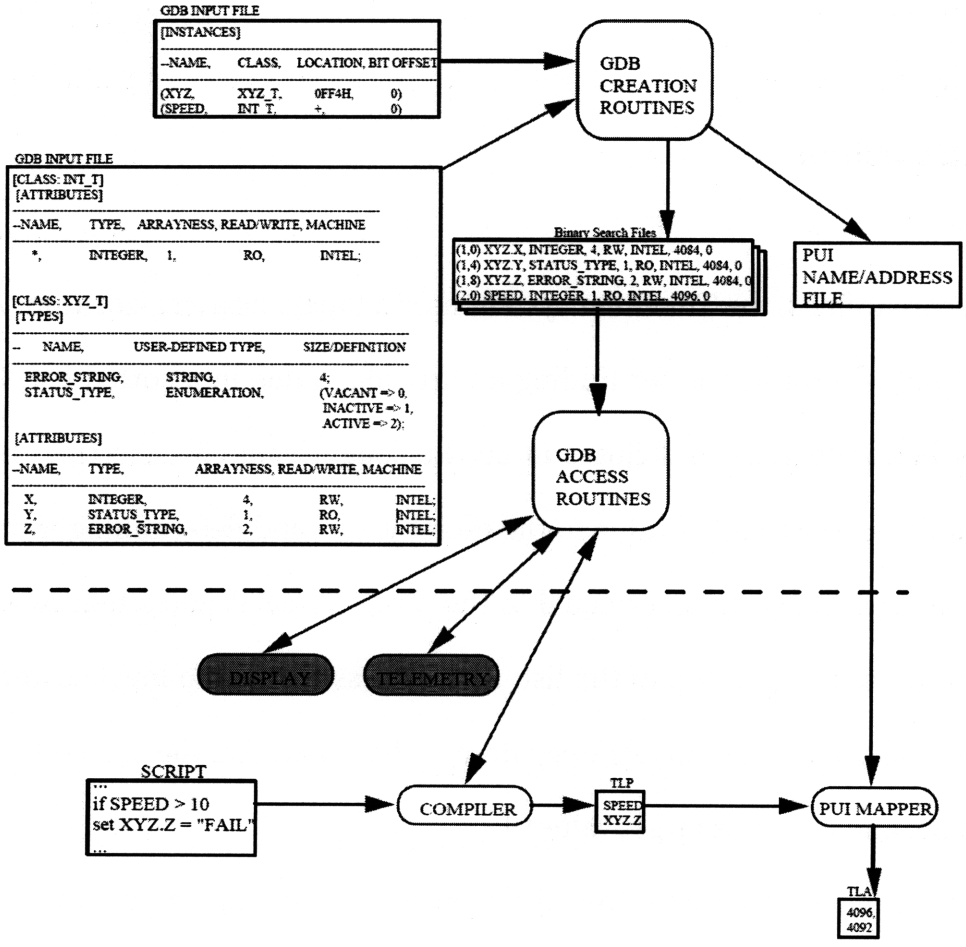


Figure 2: Design of the GDB system [3].

Variables in the GDB are defined using classes which have attributes. A class defines either a single-element or multi-element object. This allows the GDB to have single-item classes that can be referred to by only the instance name, but also multi-item classes of which there could be several instances. Commands in the GDB are defined as data objects that contain lists of parameters. In the existing implementation, the GDB consists of several binary trees: commands the system recognizes, class descriptions for data types, user data types, data type attributes, data type parameter, specific instances of data types, and specific instances of data types with alternative names. A binary tree is a tree data structure in which each node has at most two children (4).

Existing Search Algorithm

To optimize search times, the Timeliner system uses a binary search algorithm. A binary search algorithm is a technique for finding a particular value in a sorted list. It makes progressively better guesses, and closes in on the sought value by selecting the median element in a list, comparing its value to the target value, and determining if the selected value is greater than, less than, or equal to the target value. A guess that turns out to be too high becomes the new top of the list, and a guess that is too low becomes the new bottom of the list. Pursuing this strategy iteratively, it narrows the search by a factor of two each time, and finds the target value (5).

Binary search is a logarithmic algorithm. This means that finding an item in a list of N items takes up to $log(N)$ iterations of the algorithm, also known as probes. Figure 3

shows how search time (in number of probes) increases logarithmically as the number of items to be searched increases.

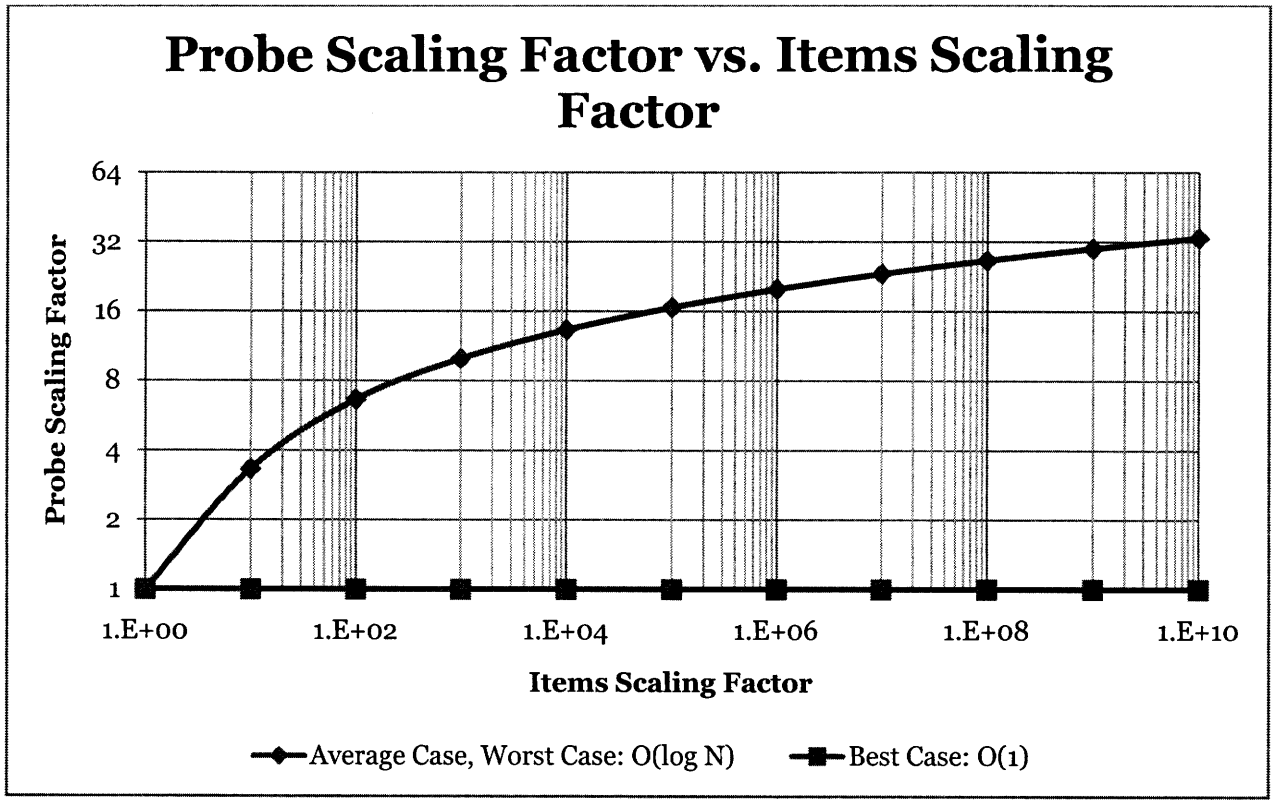


Figure 3: Best, worst, and average cases of a binary search algorithm.

Unfortunately, the performance of binary searches deteriorates when the GDB trees are sorted, which is often the case. When trees are sorted, the search scales as $O(N)$. This scaling is shown in Figure 4, and is unacceptable.

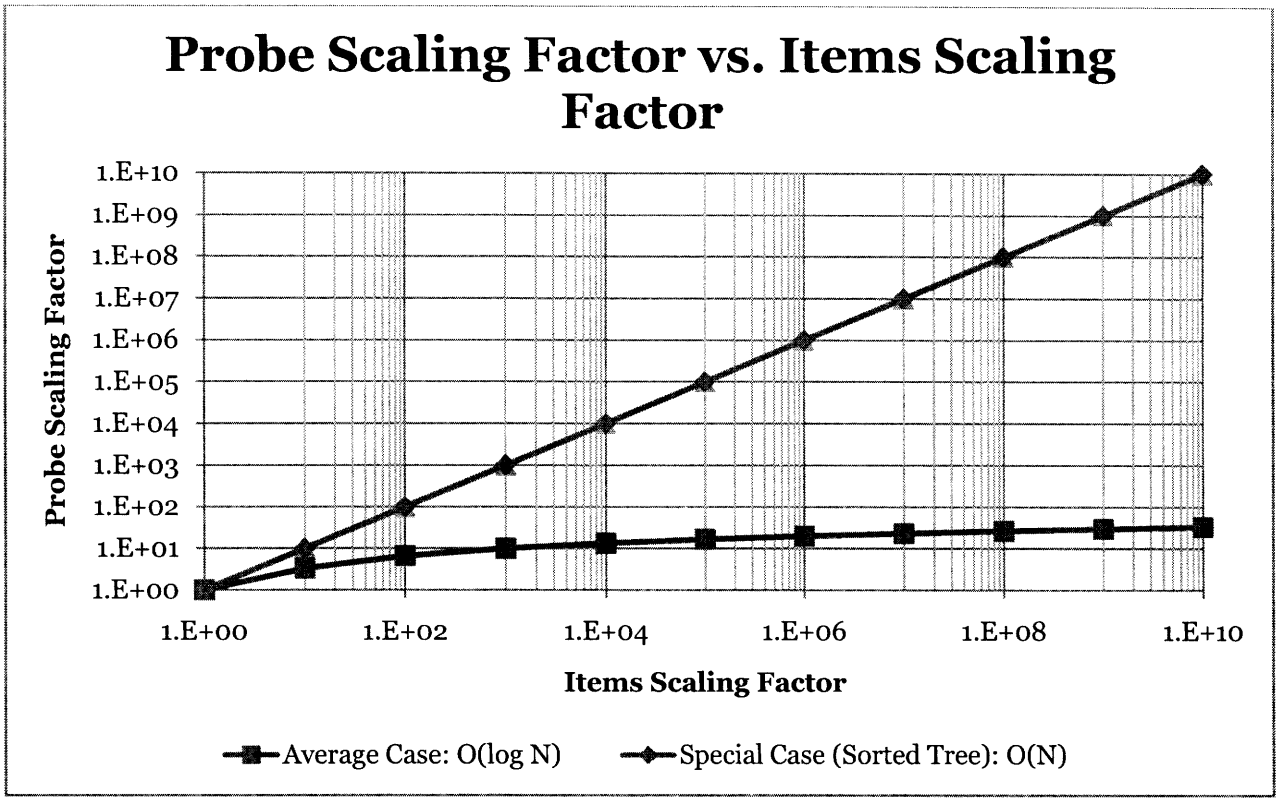


Figure 4: Special case of a sorted tree and average case of a binary search algorithm.

New Search Algorithm

A drawback of the binary search algorithm is that search time increases as the size of the list increases, often as poorly as $O(N)$ when the GDB trees are sorted. It is possible for the GDB to contain hundreds of thousands items, and so compilation can take unacceptably long. Ideally, search times should be $O(1)$.

Hash table is a structure that boasts near $O(1)$ search times, making it more suitable for the GDB. A hash function to efficiently map certain identifiers or keys to associated values. The hash function is used to transform the key into the index of an array element where the corresponding value is to be sought [6].

Figure 5 shows how search time (in number of probes) remains constant as the number of items searched increases.

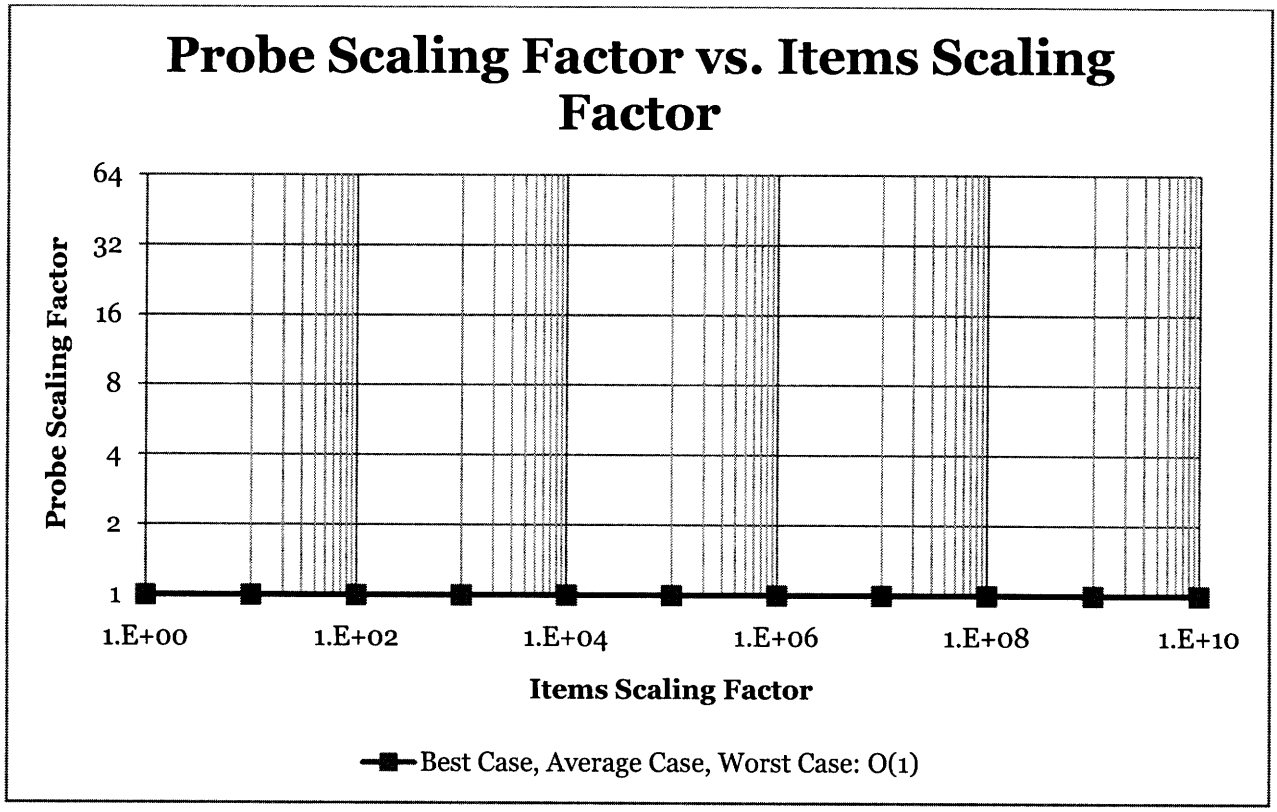


Figure 5: Best, worst, and average cases of searching a hash table.

The basic unit of the implementation of hash tables used here is a node which consists of a key (the name), an element (the datum), and a pointer to the next node. Nodes are linked to other nodes whose key hash is identical to form chains, and each such node chain is assigned a unique number (called a bucket). When inserting or searching for an element, the program uses the element's key hash to determine which bucket to use, then searches the node chain until it finds the element (for searches) or the end of the chain (for inserting). The choice of hash table size (number of buckets) strongly affects search performance: if too small, the node chains become long, and thus

searches deteriorate to $O(N)$ (essentially linearly searching a list); if too large, searches approach $O(1)$ at the expense of using a lot of memory. This implementation resizes and repopulates the table whenever the load factor—number of items in the table divided by number of buckets in the table—is above 1. This keeps the average chain at a single node, thereby minimizing search times, with only a moderate increase in required memory.

Most implementations of hash tables, including this one, have a built-in collision resolution strategy to handle situations in which more than one item have the same hash value. However, collision resolution complicates the structure, causing longer search times. A good hash function minimizes collisions, thereby minimizing search times, by mapping almost all keys to a unique bucket. The choice of a hashing function depends strongly on the nature of the input data, and their probability distribution in the intended application. The hash function used here is good for an unknown distribution of keys.

Hash tables greatly improve search times over binary trees; the difference becomes more noticeable as the number of entries increases. For this reason, hash tables are a better structure for the GDB. There are, however a few drawbacks: hash tables require much more room in memory (only an issue for enormous tables), they take more time to initially populate than binary trees, and they actually are less efficient for small datasets (the goal is to scale up well, not down, so this isn't really a problem).

Changes and Comparison

Every usage of and reference to binary trees in the GDB was replaced with hash tables.

- In gdbudt_user_data_types: user data types are stored as enumeration pair records, which consist of {name, value, internal code}. The binary tree holding all of the enumeration pairs was replaced with a hash table with a type's name as the key and an enumeration pair record as the table element.
- In gdbprm_parameters: command parameters are stored as parameter records, which consist of {name, primitive type, user data type, array size, byte offset, bit offset, offset, size, index}. The "user data type" element of a parameter record points to a specific location in a User Data Type structure. The binary tree holding all the of the parameter records was replaced with a hash table with a parameter's name as the key and a parameter record as the table element.
- In gdbatt_attributes: class attributes are stored as attribute records, which consist of {name, primitive type, user data type, array size, write flag, machine type, offset, size, index, protocol info}. The "user data type" element of a parameter record points to a specific location in a User Data Type structure. The binary tree holding all the of the attribute records was replaced with a hash table with an attribute's name as the key and an attribute record as the table element.
- In gdbins_instances: specific instances of classes are stored as instance records, which consist of {name, alternative name, location ID, bit offset, instance counter, of class, protocol information}. The "of class" element of an instance record points to a specific location in a Classes structure. The binary tree holding all the of the instance records was replaced with a hash table with an instance's name as the key and an instance record as the table element.

- In gdbcla_classes: class definitions are stored as class records, which consist of {name, protocol name, user data types, attributes, attributes start, attributes stop, last offset}. The "user data types" element of a class record is a User Data Type structure that maps user data type names to their respective values. Similarly, the "attributes" element of a class record is an Attribute structure that maps attribute names to their respective values. The binary tree holding all the of the class records was replaced with a hash table with a class' name as the key and a class record as the table element.
- In gdbcmd_commands: command definitions are stored as command records, which consist of {name, machine type, command shell, user data types, parameters, last offset, file index, number of data object parameters}. The "user data types" element of a class record is a User Data Type structure that maps user data type names to their respective values. Similarly, the "parameters" element of a class record is a Parameters structure that maps parameter names to their respective values. The binary tree holding all the of the command records was replaced with a hash table with a command's name as the key and a command record as the table element.
- In gdbalt_alternatives (defines alternative instances), gdb_initial_values (sets variable initial values), gdb_definitions (interface for storing/finding commands), gdb_protocol (handles generating protocol summary files), gdbbpt_basic_parser_interface, and gdb_parser.y (main parser files): all methods that create, add to, and search the structures were modified to interact with hash tables instead of binary trees.

Script compilation requires two steps: (1) creating a system-specific compiler from a GDB creation file and target system interface database, and (2) using the compiler. All the changed made affected step 1. Initially, it was theorized that since hash tables require more memory overhead, the GDB creation file would grow when binary trees were replaced. However, the opposite happened: using binary trees, the creation file was 2,178KB in size, as compared to 1,635KB using hash tables.

It is noted above that hash tables scale up better trees, but scale down worse. To determine at what capacity (size of target system interface database) using hash tables is preferred over binary trees, hash tables and trees of varying sizes were created, and searches for a varying number of items were timed. The results are shown in Table 1 and Table 2.

Number of Items in Binary Tree

Number of Items Retrieved	1.E+04	1.E+05	1.E+06	1.E+07
1.E+04	0	0	0	0
1.E+05		0.125	0.125	0.17
1.E+06			1.55	8.63E+6
1.E+07				8.63E+6

Table 1: Search times for binary trees with varying numbers of items

Number of Items in Hash Table

Number of Items	1.E+04	1.E+05	1.E+06	1.E+07
1.E+04	0.0156	0.0156	0.0156	0.0156

1.E+05		0.0313	0.0313	0.0781
1.E+06			0.359	75
1.E+07				75

Table 2: Search times for hash tables with varying numbers of items

These results show that searches are faster for trees than for tables for data structures with roughly 10^4 items, but that searching hash tables becomes faster than searching binary trees once the structures grow to 10^5 items and beyond.

The Next Step

The GDB is really several hash tables. When an external symbol needs to be recognized, most of the tables get searched, and each provides an element of the item being searched for. Finally, all the pieces are collected together and the symbol is resolved. The GDB is structured in this way so as to separate the data type from the data value. For example, searching for integer "A" involves searching the class table for a class description of integers, the attribute table for integer attributes, and the instance table for the value of integer "A". In essence, searching the GDB is actually searching several hash tables.

It would be possible to restructure the GDB in a way that decreases the number of hash table searches. Instead of breaking up a symbol's information into several different tables, each object type would have its own dedicated hash table. In these, searching for a symbol returns all available information regarding that symbol. Another hash table would be needed to resolve a symbol's type: the symbol would the key, which would map

to a pointer to the symbol type's dedicated table. In this way, each symbol would be fully resolved in only two $O(1)$ hash table searches. Figure 6 shows how resolving the example integer "A" would work.

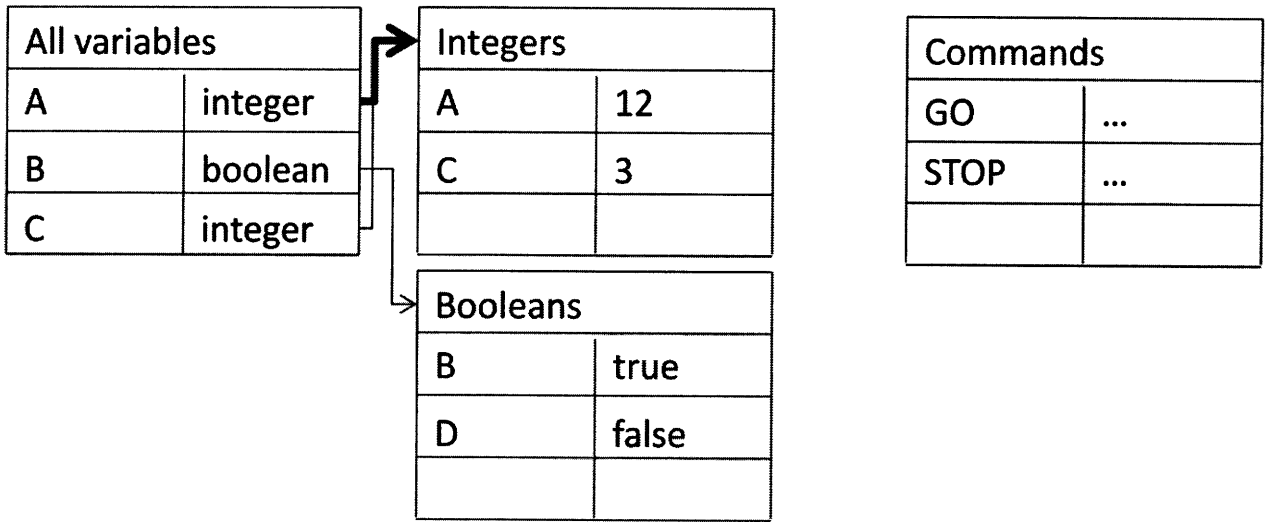


Figure 6: An example of the proposed new GDB structure. Here, the symbol to be resolved is "A"—an integer. First, "A" is searched for in the "All variables" table. This returns a pointer to the "Integer" table, which then is searched for "A". This returns all the information associated with "A"—in this case, the number 12.

In this way, the information in the GDB would be more logically grouped by type, so each table could be optimized for that particular data type. This implies that each individual table would be much smaller than the tables in the current GDB—meaning faster searches.

Bibliography

1. ISS Timeliner Ground Tools Release Notes. [Online]
http://timeliner.draper.com/docs/306644Rev51.pdf.

2. International Space Station User Interface Language User's Guide. [Online]
http://timeliner.draper.com/docs/306642r2_ug.pdf.

3. The Timeliner User Interface Language System for the International Space Station documentation. [Online]
http://timeliner.draper.com/docs/971106_ISS_TL_WRITEUP.pdf.

4. Wikipedia entry for Binary Tree. [Online] http://en.wikipedia.org/wiki/Binary_tree.

5. Wikipedia entry for Binary Search. [Online]
http://en.wikipedia.org/wiki/Binary_search.

6. Wikipedia entry for Hash Table. [Online] http://en.wikipedia.org/wiki/Hash_table.

Appendix A: Modified code with changes in bold

Gdb_definitions_b

```
------------------------------------------------------------------------------
--
-- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--
------------------------------------------------------------------------------
--
-- Package Name: GDB_DEFINITIONS
--
-- Description:   This module provides an interface for storing and looking up
--                definitions of associations of names to command Ids,
--                command shell byte offsets, data types, and constant values.
--
--
-- Modification History:
--
--      Date      Name      PCR   Description
--      ----      ----      ---   -----------
--      05/03/00  D.DiBiaso x200  Created
--      05/05/00  D.DiBiaso x235  Added support for defining/resolving names for
--                                constant values
--        04/17/09     I.Charny     TBD       Changed  binary  trees  to  hashtables  in
Define_Command_Id,
--                                                    Lookup_Command_Id,
Define_Data_Type, Lookup_Data_Type,
--                                                    Define_Shell_Offset,
Lookup_Shell_Offset,
--                                                    Delete_Shell_Overlays,
Define_Shell_Overlay,
--                                                    Lookup_Shell_Overlay,
Define_Constant_Value,
--                                                    and Lookup_Constant_Value
--
------------------------------------------------------------------------------

with ada.containers.hashed_maps;
with ada.strings.hash;
with CASE_CONVERSION;

package body GDB_DEFINITIONS is

    ------------------------------------------------------------
    --- Data types related to storing command Id associations ---
    ------------------------------------------------------------

    --- Declare command Id record definition
    type Command_Id_Record_T is record
        Name          : GDBBAS.Name_T := (others => ' ');
        Value         : TYPES.Long_Integer_T := 0;
    end record;
    function "<" (x,y : in Command_Id_Record_T) return boolean;

       -- Declare hash table of command ids
   package Command_Ids_T is new ada.containers.Hashed_Maps
         (Key_Type => Gdbbas.Name_t,
          Element_Type => Command_Id_Record_T,
          Hash => Ada.Strings.Hash,
          Equivalent_Keys => "=");


       Command_Ids : Command_Ids_T.map := Command_Ids_T.Empty_Map;

    ------------------------------------------------------------
    --- Data types related to storing data type  associations ---
    ------------------------------------------------------------
```

```
--- Declare data type record definition
type Data_Type_Record_T is record
    Name          : GDBBAS.Name_T := (others => ' ');
    Value         : GDBBAS.Name_T := (others => ' ');
end record;
function "<" (x,y : in Data_Type_Record_T) return boolean;

 -- Declare hash table of data types
package Data_Types_T is new ada.containers.Hashed_Maps
     (Key_Type => Gdbbas.Name_t,
      Element_Type => Data_Type_Record_T,
      Hash => Ada.Strings.Hash,
      Equivalent_Keys => "=");

 Data_Types : Data_Types_T.map := Data_Types_T.Empty_Map;

-----------------------------------------------------------------
--- Data types related to storing shell offset  associations ---
-----------------------------------------------------------------

--- Declare shell offset record definition
type Shell_Offset_Record_T is record
    Name          : GDBBAS.Name_T := (others => ' ');
    Value         : TYPES.Integer_T := 0;
end record;
function "<" (x,y : in Shell_Offset_Record_T) return boolean;

-- Declare hash table of shell offsets
    package Shell_Offsets_T is new ada.containers.Hashed_Maps
      (Key_Type => Gdbbas.Name_t,
       Element_Type => Shell_Offset_Record_T,
       Hash => Ada.Strings.Hash,
       Equivalent_Keys => "=");

    Shell_Offsets : Shell_Offsets_T.map := Shell_Offsets_T.Empty_Map;

-----------------------------------------------------------------
--- Data types related to storing constant value associations ---
-----------------------------------------------------------------

--- Declare constant value record definition
type Constant_Value_Record_T is record
    Name          : GDBBAS.Name_T := (others => ' ');
    Value         : GDB_INITIAL_VALUES.Value_Record;
end record;
function "<" (x,y : in Constant_Value_Record_T) return boolean;

-- Declare hash table of constant values
    package Constant_Values_T is new ada.containers.Hashed_Maps
      (Key_Type => Gdbbas.Name_t,
       Element_Type => Constant_Value_Record_T,
       Hash => Ada.Strings.Hash,
       Equivalent_Keys => "=");

 Constant_Values : Constant_Values_T.map := Constant_Values_T.Empty_Map;

-------------------------------------------------------------------------------
--- Data types related to storing shell overlays to byte offset associations ---
-------------------------------------------------------------------------------

--- Declare shell overlay record definition
type Shell_Overlay_Record_T is record
    Name          : GDBBAS.Name_T := (others => ' ');
    Value         : TYPES.Integer_T := 0;
end record;
function "<" (x,y : in Shell_Overlay_Record_T) return boolean;
```

```
   Last_Shell_Overlay_Command : GDBBAS.Name_T := (others => ' ');

   -- Declare hash table of shell offsets
       package Shell_Overlays_T is new ada.containers.Hashed_Maps
         (Key_Type => Gdbbas.Name_t,
          Element_Type => Shell_Overlay_Record_T,
          Hash => Ada.Strings.Hash,
          Equivalent_Keys => "=");

    Shell_Overlays : Shell_Overlays_T.map := Shell_Overlays_T.Empty_Map;

   -- Declare comparison for traversing command Ids
   function "<" (x,y : in Command_Id_Record_T) return boolean is
   begin
       return (x.Name < y.Name);
   end "<";

   -- Declare comparison for traversing data types
   function "<" (x,y : in Data_Type_Record_T) return boolean is
   begin
       return (x.Name < y.Name);
   end "<";

   -- Declare comparison for traversing shell offsets
   function "<" (x,y : in Shell_Offset_Record_T) return boolean is
   begin
       return (x.Name < y.Name);
   end "<";

   -- Declare comparison for traversing shell overlays
   function "<" (x,y : in Shell_Overlay_Record_T) return boolean is
   begin
       return (x.Name < y.Name);
   end "<";

   -- Declare comparison for traversing constant values
   function "<" (x,y : in Constant_Value_Record_T) return boolean is
   begin
       return (x.Name < y.Name);
   end "<";



   --- Procedure to store a defined command id association
   procedure Define_Command_Id(Name   : in  GDBBAS.Name_T;
                               Id     : in  TYPES.Long_Integer_T;
                               Exists : out boolean) is
       Cmd_Id : Command_Id_Record_T;
        Table : Command_Ids_T.map;
       cursor : Command_Ids_T.Cursor;
   begin
       --- Set up to add command Id
       Cmd_Id.Name  := Name;
       Cmd_Id.Value := Id;
       Case_Conversion.To_Upper(Cmd_Id.Name);

      --- Search table of TIL names for protocol
      Command_Ids_T.Insert(Container => Table,
                           Key       => Name,
                           New_Item  => Cmd_Id,
                           Position  => cursor,
                           Inserted  => Exists);
      exists := not exists;

   end Define_Command_Id;

   --- Procedure to resolve a defined command id association
   procedure Lookup_Command_Id(Name   : in  GDBBAS.Name_T;
                               Id     : out TYPES.Long_Integer_T;
```

```
                              Found  : out boolean) is
    Cmd_Id : Command_Id_Record_T;
     Table : Command_Ids_T.map;
    cursor : Command_Ids_T.Cursor;
    use Command_Ids_T;
begin

    --- Setup to search for command id name
    Cmd_Id.Name := Name;
    Case_Conversion.To_Upper(Cmd_Id.Name);

    --- Search for command id name
   cursor := Command_Ids_T.Find(Container => Table,
                                Key       => Name);
   Cmd_Id := Command_Ids_T.Element(cursor);

    --- Set return value
    Id := Cmd_Id.Value;
   Found := (cursor /= Command_Ids_T.No_Element);

end Lookup_Command_Id;

--- Procedure to store a defined data type name association
procedure Define_Data_Type(Name        : in  GDBBAS.Name_T;
                           Data_Type   : in  GDBBAS.Name_T;
                           Exists      : out boolean) is
    Dtype : Data_Type_Record_T;
   Table  : Data_Types_T.map;
   cursor : Data_Types_T.Cursor;
begin
    --- Set up to add data type
    Dtype.Name  := Name;
    Dtype.Value := Data_Type;
    Case_Conversion.To_Upper(Dtype.Name);

     --- Search table of TIL names for protocol
     Data_Types_T.Insert(Container => Table,
                         Key => Name,
                         New_Item => Dtype,
                             Position  => cursor,
                             Inserted  => Exists);

   exists := not exists;
end Define_Data_Type;

--- Procedure to resolve a defined data type name association
procedure Lookup_Data_Type(Name      : in  GDBBAS.Name_T;
                           Data_Type : out GDBBAS.Name_T;
                           Found     : out boolean) is
    Dtype : Data_Type_Record_T;
     Table  : Data_Types_T.map;
    cursor : Data_Types_T.Cursor;
    use Data_Types_T;
begin

    --- Setup to search for data type name
    Dtype.Name := Name;
    Case_Conversion.To_Upper(Dtype.Name);

    --- Search for data type
    cursor := Data_Types_T.Find(Container => Table,
                                Key       => Name);
     Dtype := Data_Types_T.Element(cursor);

    --- Set return value
   Data_Type := Dtype.Value;
   Found := (cursor /= Data_Types_T.No_Element);
```

```
end Lookup_Data_Type;


--- Procedure to store a defined command shell byte offset name association
procedure Define_Shell_Offset(Name        : in  GDBBAS.Name_T;
                              Byte_Offset : in  TYPES.Integer_T;
                              Exists      : out boolean) is
    Shell_Offset : Shell_Offset_Record_T;
     Table  : Shell_Offsets_T.map;
    cursor : shell_offsets_t.Cursor;
begin
    --- Set up to add shell offset
    Shell_Offset.Name  := Name;
    Shell_Offset.Value := Byte_Offset;
    Case_Conversion.To_Upper(Shell_Offset.Name);

    --- Attempt to add shell offset name association
   Shell_Offsets_T.Insert(Container => Table,
                          Key       => Name,
                          New_Item  => Shell_Offset,
                          Position  => cursor,
                          Inserted  => Exists);
   exists := not exists;

end Define_Shell_Offset;

--- Procedure to resolve a defined command shell byte offset name association
procedure Lookup_Shell_Offset(Name        : in  GDBBAS.Name_T;
                              Byte_Offset : out TYPES.Integer_T;
                              Found       : out boolean) is
    Shell_Offset : Shell_Offset_Record_T;
    Table : Shell_Offsets_T.map;
    cursor : shell_offsets_t.Cursor;
    use shell_offsets_t;
begin

    --- Setup to search for shell offset name
    Shell_Offset.Name := Name;
    Case_Conversion.To_Upper(Shell_Offset.Name);

    --- Search for shell offset name
    cursor := Shell_Offsets_T.Find(Container => Table,
                                   Key       => Name);
     Shell_Offset := Shell_Offsets_T.Element(cursor);

    --- Set return value
    Byte_Offset := Shell_Offset.Value;
    Found := (cursor /= Shell_Offsets_T.No_Element);

end Lookup_Shell_Offset;

procedure Delete_Shell_Overlays is
begin
    --- If so, destroy the last table and create a new one
     Shell_Overlays_T.Clear(Shell_Overlays);
     Shell_Overlays := Shell_Overlays_T.Empty_Map;
    Last_Shell_Overlay_Command := (others => ' ');
end Delete_Shell_Overlays;

--- Procedure to store a defined command shell overlay name to byte offset association
procedure Define_Shell_Overlay(Name        : in  GDBBAS.Name_T;
                               Command     : in  GDBBAS.Name_T;
                               Byte_Offset : in  TYPES.Integer_T;
                               Exists      : out boolean) is
    Shell_Overlay : Shell_Overlay_Record_T;
    Table  : Shell_Overlays_T.map;
    cursor : Shell_Overlays_T.Cursor;
begin
```

```
        --- Is this a shell overlay for a command different than the last command
        if Command /= Last_Shell_Overlay_Command then

            --- If so, destroy the last table and create a new one
             Shell_Overlays_T.Clear(Shell_Overlays);
             Shell_Overlays := Shell_Overlays_T.Empty_Map;
            Last_Shell_Overlay_Command := Command;

        end if;

        --- Set up to add shell overlay
        Shell_Overlay.Name  := Name;
        Shell_Overlay.Value := Byte_Offset;
        Case_Conversion.To_Upper(Shell_Overlay.Name);

        --- Attempt to add shell overlay name association
      Shell_Overlays_T.Insert(Container => Table,
                              Key       => Name,
                              New_Item  => Shell_Overlay,
                              Position  => cursor,
                              Inserted  => Exists);
       exists := not exists;

    end Define_Shell_Overlay;

    --- Procedure to lookup a shell overlay name association with a byte offset
    procedure Lookup_Shell_Overlay(Name        : in  GDBBAS.Name_T;
                                   Command     : in  GDBBAS.Name_T;
                                   Byte_Offset : out TYPES.Integer_T;
                                   Found       : out boolean) is
        Shell_Overlay : Shell_Overlay_Record_T;
         Table : Shell_Overlays_T.map;
    begin

        --- Is this a shell overlay for a command different than the last command
        if Command /= Last_Shell_Overlay_Command then
            Found := false;
            Byte_Offset := 0;
        else
            --- Setup to search for shell overlay name
            Shell_Overlay.Name := Name;
            Case_Conversion.To_Upper(Shell_Overlay.Name);

            --- Search for shell overlay name
             Shell_Overlay := Shell_Overlays_T.Element(Container => Table,

Key => Name);

            --- Set return value
            Byte_Offset := Shell_Overlay.Value;

        end if;

    end Lookup_Shell_Overlay;

    --- Procedure to store a defined name association with a constant value
    procedure Define_Constant_Value(Name        : in  GDBBAS.Name_T;
                                    Value       : in GDB_INITIAL_VALUES.Value_Record;
                                    Exists      : out boolean) is
        Constant_Value : Constant_Value_Record_T;
        Table  : Constant_Values_T.map;
       cursor : Constant_Values_T.Cursor;
    begin
        --- Set up to add constant value
        Constant_Value.Name  := Name;
        Constant_Value.Value := Value;
        Case_Conversion.To_Upper(Constant_Value.Name);

        --- Attempt to add constant value name association
```

```
        Constant_Values_T.Insert(Container => Table,
                                 Key       => Name,
                                 New_Item  => Constant_Value,
                                 Position  => cursor,
                                 Inserted  => exists);
        exists := not exists;

    end Define_Constant_Value;

    --- Procedure to lookup a defined name association with a constant value
    procedure Lookup_Constant_Value(Name        : in  GDBBAS.Name_T;
                                    Value       : out GDB_INITIAL_VALUES.Value_Record;
                                    Found       : out boolean) is
        Constant_Value : Constant_Value_Record_T;
        Table : Constant_Values_T.map;
        cursor : Constant_Values_T.Cursor;
        use Constant_Values_T;
    begin

        --- Setup to search for constant value name
        Constant_Value.Name := Name;
        Case_Conversion.To_Upper(Constant_Value.Name);

      --- Search for constant value name
      cursor := Constant_Values_T.Find(Container => Table,
                                       Key       => Name);
         Constant_Value := Constant_Values_T.Element(cursor);

        --- Set return value
      Value := Constant_Value.Value;
       Found := (cursor /= Constant_Values_T.No_Element);

    end Lookup_Constant_Value;

begin
    declare
        exists : boolean;
    begin
        Define_Command_Id("CUSTOM
", 0, exists);
    end;

end GDB_DEFINITIONS;
```

Gdb_initial_values_b

```
------------------------------------------------------------------------------
--
-- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--
------------------------------------------------------------------------------
--
-- Package Name: GDB_INITIAL_VALUES
--
-- Description:   This module provides an interface to
--                set and generate the initial values for object instances
--
--
-- Modification History:
--
--     Date      Name      PCR   Description
--     ----      ----      ---   -----------
--     06/25/99  D.DiBiaso 003   Created
--     04/24/00  D.DiBiaso 308   Pass string value into set_float_value,
--                               set_int_value, and set_boolean_value; output
--                               error value instead of instance name for bad
--                               instance initial value error message
--     05/01/00  D.DiBiaso 059   Extract set_initial_value procedure from
--                               add_instance procedure, to allow usage for both
```

```
--                                  shell overlay and initial instance values
--     05/02/00  D.DiBiaso 229      Add error message for too long instance initial
--                                  string values
--     05/05/00  D.DiBiaso 235      Create and use record for storing value definitions
--     05/17/00  D.DiBiaso 314      Moved MAX_OBJECT_SIZE from package body to spec
--     02/04/03  D.DiBiaso x516     Fixed bug copying string value in set_int_value proc
--        04/17/09    I.Charny    TBD      Changed binary trees to hash tables in
Add_Instance
--                                                          and Process_Attribute_Value
--
-------------------------------------------------------------------------------
with UNCHECKED_CONVERSION;
with TEXT_IO;
with SYSTEM;
with GDBERR_ERROR_REPORTING;
with TL_DATA_CONVERSION;
with SEQUENTIAL_IO;
with Gdbudt_User_Data_Types;
with ada.strings.Hash;
with ada.Containers.Hashed_Maps;

package body GDB_INITIAL_VALUES is

    package GDBERR renames GDBERR_ERROR_REPORTING;
    package Gdbudt renames Gdbudt_User_Data_Types;

    function "-"(Left, Right : GDB_TIMELINER_INTERFACE_TYPES.Long_Integer_T)
        return GDB_TIMELINER_INTERFACE_TYPES.Long_Integer_T
        renames GDB_TIMELINER_INTERFACE_TYPES."-";

    -- Filename for instance values file
    Values_Filename : constant string := "TIDB_INSTANCE_VALUES.BIN";

    -- Declare Object values buffer
    type Buffer_T is array (GDBBAS.Long_Natural_T range <>) of Types.Byte_T;
    pragma pack(Buffer_T);
    subtype Object_Buffer_Range is GDBBAS.Long_Natural_T range 0..Max_Object_Size-1;
    subtype Object_Buffer_T is Buffer_T(Object_Buffer_Range);
    Object_Buffer : Object_Buffer_T;

    -- Declare file block buffer
    File_Buffer_Size : constant := 1024;
    subtype File_Buffer_Range is GDBBAS.Long_Natural_T range  0..File_Buffer_Size-1;
    subtype File_Buffer_T is Buffer_T(File_Buffer_Range);
    File_Buffer : File_Buffer_T;
    File_Buffer_Offset : File_Buffer_Range;

    -- Declare package and file type for writing file block buffers
    package Buffer_IO is new Sequential_IO(File_Buffer_T);
    Values_File : Buffer_IO.File_Type;

    -- Declare record for storing initial value information
    type value_t is array(GDB_TYPES.Long_Integer_T range 1..256) of Types.byte_t;
    for value_t'size use 2048;
    type Value_Record_T is record
        Instance_Id   : Gdbbas.Instance_Id_T :=   Gdbbas.Null_Instance_Id;
        Value_Size    : GDB_TYPES.Long_Integer_T := 0;
        Value         : Value_T := (others=> 0);
    end record;

   function "<" (x,y : in Value_Record_T) return boolean;

   -- Declare hash table of instance values
   package Values_t is new ada.containers.hashed_maps
         (Key_Type => Gdbbas.name_t,
          Element_Type => Value_record_t,
          Hash => ada.Strings.Hash,
          Equivalent_Keys => "=");
```

```
      Values : Values_t.map := Values_t.Empty_Map;

     -- Declare comparison for traversing instance value table
    function "<" (x,y : in Value_Record_T) return boolean is
        function   "<"(Left,   Right   :   GDB_TYPES.Long_Integer_T)   return   boolean   renames
GDB_TYPES."<";
    begin
        return (x.Instance_Id < y.Instance_Id);
    end "<";

   -- Prcoedure to set a floating point type initial value for the
    -- current object instance
    procedure Set_Float_Value(val : in Types.Long_Float_T; val_str : in string) is
        len : GDB_TYPES.Long_Integer_T := val_str'length;
    begin
        Current.Float_Value := Val;
        Current.Value_Type := FLOAT;
        for i in val_str'first .. val_str'last loop
            if val_str(i) = ' ' then
                len := GDB_TYPES.Long_Integer_T(i - 1);
                exit;
            end if;
        end loop;
        Current.String_Value := (others => ' ');
        Current.String_Value(1..integer(len)) := Val_str(1..integer(len));
        Current.String_Length := Len;
    end Set_Float_Value;

    -- Procedure to set an integer type initial value for the
    -- current object instance
    procedure Set_Int_Value(val : in Types.Long_Integer_T; val_str : in string) is
        len : GDB_TYPES.Long_Integer_T := val_str'length;
    begin
        Current.Int_Value := Val;
        Current.Value_Type := INT;
        for i in val_str'first .. val_str'last loop
            if val_str(i) = ' ' then
                len := GDB_TYPES.Long_Integer_T(i - 1);
                exit;
            end if;
        end loop;
        Current.String_Value := (others => ' ');
        Current.String_Value(1..integer(len))                                                    :=
Val_str(val_str'first..val_str'first+integer(len)-1);
        Current.String_Length := Len;
   end Set_Int_Value;

    -- Procedure to set a string type initial value for the
    -- current object instance
    procedure Set_String_Value(val : in String;
                               len : in GDB_TYPES.Long_Integer_T) is
    begin
        Current.String_Value := (others => ' ');
        Current.String_Value(1..integer(len)) := Val(1..integer(len));
        Current.String_Length := Len;
        Current.Value_Type := STR;
     end Set_String_Value;

    -- Procedure to set a boolean type initial value for the
    -- current object instance
    procedure Set_Boolean_Value(Val : in Boolean; val_str : in string) is
        len : GDB_TYPES.Long_Integer_T := val_str'length;
    begin
        Current.Boolean_Value := Val;
        Current.Value_Type := Bool;
        for i in val_str'first .. val_str'last loop
            if val_str(i) = ' ' then
                len := GDB_TYPES.Long_Integer_T(i - 1);
                exit;
            end if;
        end loop;
```

```
        Current.String_Value := (others => ' ');
        Current.String_Value(1..integer(len)) := Val_str(1..integer(len));
        Current.String_Length := Len;
    end Set_Boolean_Value;

    -- Procedure to clear the last instance initial value
    procedure Clear_Last_Value is
    begin
        Current.Value_Type := NONE;
        Current.String_Value := (others=> ' ');
        Current.String_Length := 0;
    end Clear_Last_Value;

    -- Prcoedure to add an instance's initial value
    procedure Add_Instance(Inst_Info : in GDBINS.Instance_Record_T;
                           Att_Info  : in GDBATT.Attribute_Record_T) is
        -- Function renames
        function "="(Left, Right : GDBBAS.Primitives) return Boolean renames GDBBAS."=";
        function  "="(Left,  Right  :  GDB_TYPES.Basic_Machine_Types)  return  Boolean  renames
GDB_TYPES."=";
        Value : Value_Record_T;
        Value_OK : boolean := false;
         Value_Table : Values_T.map;
        Exists : boolean;
        Machine_Type : TL_DATA_CONVERSION.machine_type_t;
        function ctob is new unchecked_conversion(character, Types.byte_t);
        Quote : String(1..1) := (others => '"');
        Udt_Data :   Gdbudt.User_Data_Type_Record_T;
       cursor : Values_T.Cursor;
       use Values_T;
    begin
        -- Is this a starred instance?
        if Att_Info.Name(1) = ' ' then

            -- Is a value specified for this instance
            if Current.Value_Type /= NONE then

                -- Determine Local Machine Type
                case Att_Info.machine_type is
                    when GDB_TYPES.sun =>
                        machine_type := tl_data_conversion.sun;
                    when GDB_TYPES.intel =>
                        machine_type := tl_data_conversion.intel;
                    when GDB_TYPES.vax =>
                        machine_type := tl_data_conversion.vax;
                end case;

                Set_Initial_Value(Att_Info.Primitive_Type,  Machine_Type,  Value.Value(1)'address,
Value.Value_Size, Value_OK);

                -- If Value processing has been successfull, add value to table
                if Value_Ok then
                    -- Check for string length too long
                    if ((Att_Info.Primitive_Type = GDBBAS.Of_String) or (Att_Info.Primitive_Type
= GDBBAS.Undefined)) and
                       (Current.Value_Type = STR) then
                        Udt_Data := Gdbudt.Gdbuti.element(Att_Info.User_Data_Type);
                       if integer(Current.String_Length) > integer(Udt_Data.String_Length) then
                           GDBERR.Report_Error(GDBERR.INSTANCE_STRING_TOO_LONG,      Quote      &
Current.String_Value(1..integer(Current.String_Length)) & Quote );
                       end if;
                    end if;

                    -- Add value to table
                Value.Instance_Id := Inst_Info.Location_Id;
                Values_T.Insert(Container => Values,
                                Key                                                          =>
gdbbas.Instance_Id_T'Image(value.Instance_Id),
                                New_Item  => Value,
                                Position  => cursor,
```

```
                                Inserted  => exists);
                else
                    if Current.Value_Type = Str then
                        GDBERR.Report_Error(GDBERR.BAD_INSTANCE_VALUE,          Quote          &
Current.String_Value(1..integer(Current.String_Length)) & Quote );
                    else
                        GDBERR.Report_Error(GDBERR.BAD_INSTANCE_VALUE,
Current.String_Value(1..integer(Current.String_Length)));
                    end if;
                end if;

            end if;

        -- If this is not a starred instance and a value has been provided,
        -- then error
        elsif Current.Value_Type /= NONE then
            GDBERR.Report_Error(GDBERR.VALUE_FOR_NON_STARRED, Inst_Info.Name);
        end if;

    exception
        when others=>
                    if Current.Value_Type = Str then
                        GDBERR.Report_Error(GDBERR.BAD_INSTANCE_VALUE,          Quote          &
Current.String_Value(1..integer(Current.String_Length)) & Quote );
                    else
                        GDBERR.Report_Error(GDBERR.BAD_INSTANCE_VALUE,
Current.String_Value(1..integer(Current.String_Length)));
                    end if;

    end Add_Instance;

    procedure Set_Initial_Value(Data_Type    : in GDBBAS.Primitives;
                                Machine_Type : in TL_DATA_CONVERSION.machine_type_t;
                                Value_Addr   : in System.address;
                                Value_Size   : out GDB_TYPES.Long_Integer_T;
                                Value_OK     : out boolean) is
        function "="(Left, Right : GDBBAS.Primitives) return Boolean renames GDBBAS."=";
        Local_Value_OK : Boolean := false;
        type bytes_2 is array (1..2) of TYPES.byte_t;
        type bytes_4 is array (1..4) of TYPES.byte_t;
        type bytes_8 is array (1..8) of TYPES.byte_t;
     begin

        -- Case based upon attribute type
        case Data_Type is
            when GDBBAS.Of_Boolean =>
                declare
                    val : Types.Byte_T;
                    for val use at Value_Addr;
                begin
                    Value_Size := 1;
                    if Current.Value_Type = Bool then
                        if Current.Boolean_Value then
                            Val := 1;
                        else
                            Val := 0;
                        end if;
                        Local_Value_Ok := true;
                    end if;
                end;
            when GDBBAS.Of_Byte =>
                Value_Size := 1;
                declare
                    val : Types.Byte_T;
                    for val use at Value_Addr;
                begin
                    if Current.Value_Type = Float then
                        Val := TYPES.Byte_T(Current.Float_Value);
                        Local_Value_Ok := true;
                    elsif Current.Value_Type = Int then
                        Val := TYPES.Byte_T(Current.Int_Value);
```

```
                Local_Value_Ok := true;
            end if;
        end;
    when GDBBAS.Of_Short_Integer =>
        Value_Size := 1;
        declare
            val : Types.Short_Integer_T;
            for val use at Value_Addr;
        begin
            if Current.Value_Type = Float then
                Val := Types.Short_Integer_T(Current.Float_Value);
                Local_Value_Ok := true;
            elsif Current.Value_Type = Int then
                Val := Types.Short_Integer_T(Current.Int_Value);
                Local_Value_Ok := true;
            end if;
        end;
    when GDBBAS.Of_Natural =>
        Value_Size := 2;
        declare
            val : Types.Word_T;
            val1 : bytes_2;
            for val1 use at val'address;
            val2 : bytes_2;
            for val2 use at Value_Addr;
        begin
            if Current.Value_Type = Float then
                Val := Types.Word_T(Current.Float_Value);
                Local_Value_Ok := true;
            elsif Current.Value_Type = Int then
                Val := Types.Word_T(Current.Int_Value);
                Local_Value_Ok := true;
            end if;
            if Local_Value_Ok then
                val2(1) := val1(1); val2(2) := val1(2);
                tl_data_conversion.convert_short_int(addr=>Value_Addr,
                                                     to_type=>machine_type);
            end if;
        end;
    when GDBBAS.Of_Integer =>
        Value_Size := 2;
        declare
            val : Types.Integer_T;
            val1 : bytes_2;
            for val1 use at val'address;
            val2 : bytes_2;
            for val2 use at Value_Addr;
        begin
            if Current.Value_Type = Float then
                Val := Types.Integer_T(Current.Float_Value);
                Local_Value_Ok := true;
            elsif Current.Value_Type = Int then
                Val := Types.Integer_T(Current.Int_Value);
                Local_Value_Ok := true;
            end if;
            if Local_Value_Ok then
                val2(1) := val1(1); val2(2) := val1(2);
                tl_data_conversion.convert_short_int(addr=>Value_Addr,
                                                     to_type=>machine_type);
            end if;
        end;
    when GDBBAS.Of_Long_Natural =>
        Value_Size := 4;
        declare
            use types;
            val : Types.Long_Integer_T;
            val1 : bytes_4;
            for val1 use at val'address;
            val2 : bytes_4;
            for val2 use at Value_Addr;
        begin
```

```
                    if Current.Value_Type = Float then
                        if  Current.Float_Value  >  Types.long_float_t(Types.long_integer_t'last)
then
                            val       :=       Types.long_integer_t(Current.Float_Value       -
Types.long_float_t(Types.long_integer_t'last)- 1.0);
                            val := val - Types.long_integer_t'last - 1;
                            Local_Value_Ok := true;
                        elsif Current.Float_Value >= 0.0 then
                            Val := Types.Long_Integer_T(Current.Float_Value);
                            Local_Value_Ok := true;
                        end if;
                    elsif Current.Value_Type = Int then
                        if Current.Int_Value >= 0 then
                            Val := Types.Long_Integer_T(Current.Int_Value);
                            Local_Value_Ok := true;
                        end if;
                    end if;
                    if Local_Value_Ok then
                        val2(1) := val1(1); val2(2) := val1(2);
                        val2(3) := val1(3); val2(4) := val1(4);
                        tl_data_conversion.convert_Long_int(addr=>Value_Addr,
                                                            to_type=>machine_type);
                    end if;
                end;
            when GDBBAS.Of_Long_Integer =>
                Value_Size := 4;
                declare
                    val : Types.Long_Integer_T;
                    val1 : bytes_4;
                    for val1 use at val'address;
                    val2 : bytes_4;
                    for val2 use at Value_Addr;
                begin
                    if Current.Value_Type = Float then
                        Val := Types.Long_Integer_T(Current.Float_Value);
                        Local_Value_Ok := true;
                    elsif Current.Value_Type = Int then
                        Val := Types.Long_Integer_T(Current.Int_Value);
                        Local_Value_Ok := true;
                    end if;
                    if Local_Value_Ok then
                        val2(1) := val1(1); val2(2) := val1(2);
                        val2(3) := val1(3); val2(4) := val1(4);
                        tl_data_conversion.convert_Long_int(addr=>Value_Addr,
                                                            to_type=>machine_type);
                    end if;
                end;
            when GDBBAS.Of_Float =>
                Value_Size := 4;
                declare
                    val : Types.Float_T;
                    val1 : bytes_4;
                    for val1 use at val'address;
                    val2 : bytes_4;
                    for val2 use at Value_Addr;
                begin
                    if Current.Value_Type = Float then
                        Val := Types.Float_T(Current.Float_Value);
                        Local_Value_Ok := true;
                    elsif Current.Value_Type = Int then
                        Val := Types.Float_T(Current.Int_Value);
                        Local_Value_Ok := true;
                    end if;
                    if Local_Value_Ok then
                        val2(1) := val1(1); val2(2) := val1(2);
                        val2(3) := val1(3); val2(4) := val1(4);
                        tl_data_conversion.convert_Short_Float(addr=>Value_Addr,
                                                               to_type=>machine_type);
                    end if;
                end;
            when GDBBAS.Of_Long_Float =>
```

```
                Value_Size := 8;
                declare
                    val : Types.Long_Float_T;
                    val1 : bytes_8;
                    for val1 use at val'address;
                    val2 : bytes_8;
                    for val2 use at Value_Addr;
                begin
                    if Current.Value_Type = Float then
                        Val := Types.Long_Float_T(Current.Float_Value);
                        Local_Value_Ok := true;
                    elsif Current.Value_Type = Int then
                        Val := Types.Long_Float_T(Current.Int_Value);
                        Local_Value_Ok := true;
                    end if;
                    if Local_Value_Ok then
                        val2(1) := val1(1); val2(2) := val1(2);
                        val2(3) := val1(3); val2(4) := val1(4);
                        val2(5) := val1(5); val2(6) := val1(6);
                        val2(7) := val1(7); val2(8) := val1(8);
                        tl_data_conversion.convert_Long_Float(addr=>Value_Addr,
                                                              to_type=>machine_type);
                    end if;
                end;
            when GDBBAS.Of_String | GDBBAS.Undefined =>
                declare
                    val : String(1..integer(Current.String_Length));
                    for Val use at Value_Addr;
                begin
                    Value_Size := Current.String_Length;
                    if Current.Value_Type = Str then
                        Val := Current.String_Value(1..integer(Current.String_Length));
                        Local_Value_Ok := true;
                    end if;
                end;
            when others =>
                null;
        end case;

        value_Ok := local_value_Ok;
    end Set_Initial_Value;

    -- Procedure to open the instance values file
    procedure Open_Instance_Values_File(Tidb_Path : in String) is
    begin

        -- Create instance values file
        Buffer_IO.Create(Values_File, Buffer_Io.Out_File, Tidb_Path & Values_Filename);

        -- Initialize file block buffer offset
        File_Buffer_Offset := 0;

    -- Report Error if instance values file could not be created
    exception
        when others =>
            GDBERR.Report_Error(GDBERR.ERROR_CREATING_INST_VALS, Tidb_Path & Values_Filename);

    end Open_Instance_Values_File;

    -- Procedure to write an object's worth of bytes through the buffer to the file
    procedure Write_Object_Values(Object_Size : GDBBAS.Long_Natural_T) is
        function   ">"(Left,   Right   :   GDB_TYPES.Long_Integer_T)   return   Boolean   renames
GDB_TYPES.">";
        function   ">="(Left,   Right   :   GDB_TYPES.Long_Integer_T)   return   Boolean   renames
GDB_TYPES.">=";
        function  "+"(Left,  Right  :  GDB_TYPES.Long_Integer_T)  return  GDB_TYPES.Long_Integer_T
renames GDB_TYPES."+";
        function  "-"(Left,  Right  :  GDB_TYPES.Long_Integer_T)  return  GDB_TYPES.Long_Integer_T
renames GDB_TYPES."-";
        Object_Offset : GDBBAS.Long_Natural_T := Object_Buffer'First;
    begin
```

```
        -- If size of object is larger than room left in current file buffer block
        if Object_Size >= File_Buffer_Size - File_Buffer_Offset then

            -- Fill and write current block
            File_Buffer(File_Buffer_Offset..File_Buffer'last) :=
                Object_Buffer(Object_Offset..File_Buffer'last - File_Buffer_Offset);
            Object_Offset := Object_Offset +
                             Object_Buffer_Range(File_Buffer_Size) -
                             Object_Buffer_Range(File_Buffer_Offset);
            Buffer_IO.Write(Values_File, File_Buffer);
            File_Buffer_Offset := File_Buffer'first;

            -- Fill and write full blocks of data
            while (Object_Size - Object_Offset) >= File_Buffer_Size loop
                File_Buffer                                                    :=
Object_Buffer(Object_Offset..Object_Offset+Object_Buffer_Range(File_Buffer_Size)-1);
                Object_Offset := Object_Offset + File_Buffer_Size;
                Buffer_IO.Write(Values_File, File_Buffer);
            end loop;

            -- Fill last partial block
            if Object_Size > Object_Offset then
                File_Buffer(File_Buffer'first..Object_Size - (Object_Offset+1)) :=
                            Object_Buffer(Object_Offset..Object_Size-1);
                File_Buffer_Offset := Object_Size - Object_Offset;
            end if;
        else
            -- Object size is <= room left if file buffer block,
            -- Copy bytes to buffer and be done
            File_Buffer(File_Buffer_Offset..File_Buffer_Offset+Object_Size-1) :=
              Object_Buffer(Object_Offset..Object_Size-1);
            File_Buffer_Offset := File_Buffer_Offset + Object_Size;

        end if;

    end Write_Object_Values;

    -- Procedure to close the instance values file
    procedure Close_Instance_Values_File is
        function   ">"(Left,   Right   :   GDB_TYPES.Long_Integer_T)   return   Boolean   renames
GDB_TYPES.">";
    begin
        -- Write out the last buffer, if necessary
        if File_Buffer_Offset > File_Buffer'first then

            -- Fill the remainder of the buffer with 0's
            File_Buffer(File_Buffer_Offset..File_Buffer'last) := (others => 0);

            -- Write the last buffer to file
            Buffer_IO.Write(Values_File, File_Buffer);

        end if;
        -- Close the file
        Buffer_IO.Close(Values_File);

    end Close_Instance_Values_File;

    -- Procedure to start creating object values for new object
    procedure Start_Object_Values(Object_Size : in GDBBAS.Attribute_Offset_T) is
        function  "+"(Left,  Right  :  GDB_TYPES.Long_Integer_T)  return  GDB_TYPES.Long_Integer_T
renames GDB_TYPES."+";
        function  "-"(Left,  Right  :  GDB_TYPES.Long_Integer_T)  return  GDB_TYPES.Long_Integer_T
renames GDB_TYPES."-";
    begin
        -- Initialize object values to 0
        Object_Buffer(Object_Buffer'First..Object_Buffer'First+Object_Size-1) := (others => 0);
    end Start_Object_Values;

    -- Procedure to process Object attribute value
    procedure Process_Attribute_Value(Instance_Id   : in GDBBAS.Instance_Id_T;
                                      Attr_Name     : in GDBBAS.Name_T;
```

```
                                    Data_Type      : in GDBBAS.Primitives;
                                    Arrayness      : in GDBBAS.User_Array_Range_T;
                                    Offset         : in GDBBAS.Attribute_Offset_T;
                                    String_Length    :    in    GDBBAS.User_String_Range_T    :=
GDBBAS.User_String_Range_T'first) is
        function  "+"(Left,  Right  :  GDB_TYPES.Long_Integer_T)  return  GDB_TYPES.Long_Integer_T
renames GDB_TYPES."+";
        function  "-"(Left,  Right  :  GDB_TYPES.Long_Integer_T)  return  GDB_TYPES.Long_Integer_T
renames GDB_TYPES."-";
        function  "*"(Left,  Right  :  GDB_TYPES.Long_Integer_T)  return  GDB_TYPES.Long_Integer_T
renames GDB_TYPES."*";
        function "="(Left, Right: GDBBAS.Primitives) return boolean renames GDBBAS."=";
        function ">"(Left, Right: GDB_TYPES.Long_Integer_T) return boolean renames GDB_TYPES.">";
        function ctob is new unchecked_conversion(character, Types.byte_t);

        -- Declare local string length of data type for calculations
        Str_Len : GDB_TYPES.Long_Integer_T := GDB_TYPES.Long_Integer_T(String_Length);

        -- Declare Value record to be used for lookup
        Value_Rec  : Value_Record_T;
         Value_Table : Values_t.map;
        Exists     : boolean;
        Data_Size  : GDB_TYPES.Long_Integer_T;
       cursor : values_t.Cursor;
       use values_t;
    begin
        -- If this is a string primitive type...
        if Data_Type = GDBBAS.Of_String then

            -- Set attribute initial value to all spaces
            Object_Buffer(Offset..Offset+(Arrayness*Str_Len)-1) := (others => ctob(' '));

        end if;

        -- If this is a starred attribute
        if Attr_Name(1) = ' '  then

            -- Lookup instance Id in initial values table
            Value_Rec.Instance_Id := Instance_Id;
               cursor := Values_T.find(Container => Values,
                                       Key => gdbbas.Instance_Id_T'image(Instance_Id));
                      -- If instance value was found ...
       exists := (cursor /= Values_T.No_Element);
       if exists then
              Value_Rec := Values_T.element(Position => cursor);

             -- Adjust data size for string types
             if Data_Type = GDBBAS.Of_String then
                 Data_Size := Str_Len;
                 if Value_Rec.Value_Size > Str_Len then
                     Value_Rec.Value_Size := Str_Len;
                 end if;
             else
                 Data_Size := Value_Rec.Value_Size;
             end if;

             -- Loop for each element in array ...
             for i in 1..Arrayness loop

                 -- Set each element to initial value
                 for j in 1..Value_Rec.Value_Size loop
                     Object_Buffer(Offset+(Data_Size*(i-1))+(j-1)) := Value_Rec.Value(j);
                 end loop;

             end loop;
          end if;

        end if;

    end Process_Attribute_Value;
```

```
    -- Procedure to finish creating object values for object
    procedure Finish_Object_Values(Object_Size : in GDBBAS.Attribute_Offset_T) is
    begin
        -- Write out object values to file
        Write_Object_Values(Object_Size);
    end Finish_Object_Values;

end GDB_INITIAL_VALUES;
```

Gdb_protocol_b

```
------------------------------------------------------------------------------
--
-- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
-- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
-- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
-- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--
------------------------------------------------------------------------------
--
-- Package Name: GDB_PROTOCOL
--
-- Description:   This module provides an interface for generating the
--                protocol specific object summary mapping files
--
--
-- Modification History:
--
--     Date      Name      PCR   Description
--     ----      ----      ---   -----------
--     01/11/00  D.DiBiaso x203  Created
--     01/11/00  G.Cunha   x207  Added protocol record for the File Input
--                               capability.
--     04/05/00  D.DiBiaso x208  Added protocol records for LDAP and JDBC Tils
--     04/24/00  D.DiBiaso x304  Always create protocol object mapping file with
--                               header, even if no objects are defined.
--     05/03/00  D.DiBiaso x200  Allow list of valid TIL names to specified via
--                               TIL name GDB input definitions
--     01/06/03  D.DiBiaso x516  Added Found "out" parameter to lookup_protocol procedure
--        04/17/09    I.Charny    TBD      Changed binary trees to hash tables in
Add_Protocol,
--                                                    Lookup_Protocol,
Write_Protocol_Attributes,
--                                                    Write_Protocol_Mapping_File, and
Write_Protocol_Mapping_Files
------------------------------------------------------------------------------
with UNCHECKED_CONVERSION;
with TEXT_IO;
with SYSTEM;
with ada.containers.hashed_maps;
use ada.containers;
with Ada.Strings.hash;
use ada.strings;
with GDBINS_INSTANCES;
with GDB_FILE_SYSTEM_INTERFACE;
with GDBSES_INPUT_PROCESSOR_SESSION;
with GDBCLA_CLASSES;
with CASE_CONVERSION;
with GDBATT_ATTRIBUTES;
with GDBUDT_USER_DATA_TYPES;
with GDBERR_ERROR_REPORTING;

package body GDB_PROTOCOL is

    package GDBINS renames GDBINS_INSTANCES;
    package GDBFSI renames GDB_FILE_SYSTEM_INTERFACE;
    package GDBSES renames GDBSES_INPUT_PROCESSOR_SESSION;
    package GDBCLA renames GDBCLA_CLASSES;
    package GDBATT renames GDBATT_ATTRIBUTES;
```

```
package GDBUDT renames GDBUDT_USER_DATA_TYPES;
package GDBERR renames GDBERR_ERROR_REPORTING;

--- Declare protocol record definition
type Protocol_Record_T is record
    Name          : GDBBAS.Name_T := (others => ' ');
    Used          : Boolean := False;
end record;

function "<" (x,y : in Protocol_Record_T) return boolean;

-- Declare hash table of protocols
package Protocols_t is new ada.containers.Hashed_Maps
     (Key_Type => Gdbbas.Name_t,
      Element_Type => Protocol_record_t,
      Hash => Ada.Strings.Hash,
      Equivalent_Keys => "=");

Protocols : Protocols_t.map := Protocols_t.Empty_Map;


-- Declare comparison for traversing protocol
function "<" (x,y : in Protocol_Record_T) return boolean is
begin
    return (x.Name < y.Name);
end "<";

   --- TRIM STRING FUNCTION: TRIMS FRONT & BACK OF STRING
   function Trim(s : in string) return string is

       --- FIRST AND LAST NON-BLANK CHARACTERS
       c0 : integer := s'first;
       c1 : integer := s'last;

   begin

   --- IF STRING IS NULL OR ALL BLANKS...
       if s = (c0..c1 => ' ') then
       --- RETURN NULL STRING
           return "";

   --- OTHERWISE...
       else
       --- CHECK FORWARD FROM BEGINNING
           for i in c0..c1 loop
               if s(i) = ' ' or s(i) = ascii.ht or s(i) = ascii.cr then
                   c0 := c0 + 1;
               else
                   exit;
               end if;
           end loop;
       --- CHECK BACKWARD FROM END
           for i in reverse c0..c1 loop
               if s(i) = ' ' or s(i) = ascii.ht or s(i) = ascii.cr then
                   c1 := c1 - 1;
               else
                   exit;
               end if;
           end loop;
       --- RETURN TRIMMED STRING
           return s(c0..c1);
       end if;

   end Trim;

 --- Procedure to add a protocol to the table of protocols
 procedure Add_Protocol(Name : in GDBBAS.Name_T) is
     p : Protocol_Record_T;
      cursor : Protocols_T.Cursor;
      success : boolean;
```

```
    begin
        --- Set up to add TIL protocol name
        p.Name := Name;
        p.Used := false;
        Case_Conversion.To_Upper(p.Name);

        --- Search table of TIL names for protocol
        Protocols_T.Insert(Protocols, Name, p, cursor, success);

        --- If it is, then error
        if success = false then
            Gdberr.Report_Error(gdberr.DUP_TIL_NAME, Name);
        end if;

    end Add_Protocol;

    --- Procedure to lookup a protocol in the table of protocols
    procedure Lookup_Protocol(Name : in GDBBAS.Name_T;
                              Found  : out Boolean) is
        p : Protocol_Record_T;
         cursor : Protocols_T.Cursor;
       use Protocols_T;
    begin
        --- Set up to search for TIL protocol name
        p.Name := Name;
        Case_Conversion.To_Upper(p.Name);

        --- Search for TIL protocol name
        cursor := Protocols_T.Find(Protocols, p.Name);

        --- If protocol name is found, make sure used flag is set
        if (cursor /= Protocols_T.No_Element) then
            if not p.Used then
                p.Used := true;
             Protocols_T.Replace_Element(Container => Protocols,
                                         Position => cursor,
                                         New_Item => p);
                           Found := true;
            end if;
        else
            --- Else, error
            Gdberr.Report_Error(gdberr.BAD_TIL_NAME, Name);
                      Found := false;
        end if;
    end Lookup_Protocol;


    --- Procedure to write the attribute records
    procedure Write_Protocol_Attributes(Mapping_File : in Text_IO.File_Type;
                                        Class_Rec    : in Gdbcla.Class_Record_T) is
        use GDBBAS;
       use Gdbatt.gdbati;

        Att_Rec  :   Gdbatt.Attribute_Record_T;
       Cursor         :   Gdbatt.gdbati.Cursor;
        Udt_Data :   Gdbudt.User_Data_Type_Record_T;
        Str_Len  :   Integer;
    begin

        --- Iterate through each attribute for this class
        Cursor := Gdbatt.gdbati.First(Class_Rec.Attributes);
              while (Cursor /= Gdbatt.gdbati.No_Element) loop --get all attributes one
at a time

             --- Retrieve next attribute record
             Att_Rec := Gdbatt.Gdbati.Element(Cursor);

            --- Output attribute name
            Text_IO.Put(Mapping_File, trim(Att_Rec.Name) & ",");
```

```
            --- Output byte offset
            Text_IO.Put(Mapping_File, Gdbbas.Attribute_Offset_T'Image(Att_Rec.Offset) & ", ");

            --- Output data type
            if (Att_Rec.Primitive_Type = Gdbbas.Undefined) then

                --- it is a user-defined type (currently only a string or enumeration
               Udt_Data := Gdbudt.Gdbuti.Element(Att_Rec.User_Data_Type);
                Str_Len := Integer(Udt_Data.String_Length);
                declare
                    data_type            :            constant            string            :=
Gdbbas.Primitives'Image(Udt_Data.Primitive_Type);
                begin
                    Text_IO.Put(Mapping_File, data_type(4..data_type'last) & ",");
                end;

            else
                --- it is a primitive type
                Str_Len := 0;
                declare
                    data_type            :            constant            string            :=
Gdbbas.Primitives'Image(Att_Rec.Primitive_Type);
                begin
                    Text_IO.Put(Mapping_File, data_type(4..data_type'last) & ",");
                end;
            end if;

            --- Output Array size
            Text_IO.Put(Mapping_File, Gdbbas.User_Array_Range_T'Image(Att_Rec.Array_Size) & ",");

            --- Output string length
            Text_IO.Put(Mapping_File, Integer'Image(Str_Len) & ", ");

            --- Output protocol Info
            if Att_Rec.Protocol_Info(1) = '{' then
                Text_Io.Put_Line(Mapping_File,
                                 Trim(Att_Rec.Protocol_Info));
            else
                Text_Io.Put_Line(Mapping_File, "{}");
            end if;
               Gdbatt.Gdbati.Next(Cursor);
        end loop;

    end Write_Protocol_Attributes;


    --- Procedure to generate the protocol object mapping file for a protocol
    procedure Write_Protocol_Mapping_File(Protocol_Name   : in GDBBAS.Name_T;
                                          Objects_Defined : in Boolean := true) is

        use gdbbas;

        Mapping_File : Text_IO.File_Type;

        --- Construct the mapping filename
        Mapping_Filename : constant string := Trim(Protocol_Name) & "_OBJECT_MAP.TXT";

        --- Top of Instance Table
         Object_Table        : Gdbins.Gdbini.map := Gdbses.Global_Name_Tables.Instance;
        --- Object Record
        Object_Record     : Gdbins.Instance_Record_T;
        --- Class Record
      Class_Record      : Gdbcla.Class_Record_T;

      Cursor : Gdbins.Gdbini.Cursor := Gdbins.Gdbini.First(Object_Table);
       use Gdbins.Gdbini;

    begin
        --- Open the mapping file to be written
```

```
        Text_IO.Create (Mapping_File, Text_Io.Out_File,
                        GDBFSI.Config_Path( 1 .. GDBFSI.Config_Path_Length ) & Mapping_Filename
);

        --- Write format header to file
        Text_IO.Put_Line(Mapping_File, "-- Filename: " & Mapping_Filename);
        Text_IO.Put_Line(Mapping_File, "-- Format:");
        Text_IO.Put_Line(Mapping_File,  "--  <object  name>,  <object  id>,  <object  size>,  <num
attributes>, <til info>");
        Text_IO.Put_Line(Mapping_File, "-- <attribute name>, <byte offset>, <data type>, <array
size>, <string len>, <til info>");

        --- Iterate for each instance
        if objects_Defined then

          while (Cursor /= Gdbins.Gdbini.No_Element) loop

            --- Get next record
             Object_Record := Gdbins.Gdbini.Element(Cursor);
            --- Now, grab the class record of which this PUI is an object.
             Class_Record := Gdbcla.Gdbcli.Element(Object_Record.Of_Class);

            --- If this record of the protocol type?
            if Protocol_Name = Class_Record.Protocol then

                --- Write out the name to file
                Text_IO.Put(Mapping_File, trim(Object_Record.Name) & ",");
                --- Write PUI ID to file
                Text_Io.Put(Mapping_File,
                            Gdbbas.Long_Natural_T'Image(Object_Record.Instance_Ctr) & ",");
                --- Write size of object from class record (in bytes) to file
                Text_Io.Put(Mapping_File,
                            Gdbbas.Attribute_Offset_T'Image(Class_Record.Last_Offset) & ",");
                --- Write number of attributes in class to file
                Text_Io.Put(Mapping_File,
                            Gdbbas.File_Index_T'Image(Class_Record.Attributes_Stop + 1 -
                                                      Class_Record.Attributes_Start) & ", ");
                --- Write instance protocol info
                if Object_Record.Protocol_Info(1) = '{' then
                    Text_Io.Put_Line(Mapping_File,
                                     Trim(Object_Record.Protocol_Info));
                else
                    Text_Io.Put_Line(Mapping_File, "{}");
                end if;

                --- Write the instance attributes
                Write_Protocol_Attributes(Mapping_File, Class_Record);

                --- Add some whitespace
                Text_Io.New_Line(Mapping_File);

            end if;
              Gdbins.Gdbini.Next(Cursor);
          end loop;

        end if;

        --- Close the mapping file
        Text_IO.Close (Mapping_File);

    exception
        when others=>
          Gdberr.Report_Error(gdberr.OBJ_SUMMARY_ERROR,
                              GDBFSI.Config_Path(    1    ..    GDBFSI.Config_Path_Length    )    &
Mapping_Filename, false, false);

    end Write_Protocol_Mapping_File;

    --- Procedure to generate the protocol object mapping file for each defined protocol
    procedure Write_Protocol_Mapping_Files is
```

```
      --- Create Protocol Iterator
      Cursor : Protocols_T.Cursor := Protocols_T.First(Protocols);
      Protocol_Record     : Protocol_Record_T;
       use Protocols_T;
    begin

      --- Traverse the protocol table
        while (Cursor /= Protocols_T.No_Element) loop

            --- Get next record
            Protocol_Record := Protocols_T.Element(Cursor);

            --- Write the protocol mapping file for this protocol
            Write_Protocol_Mapping_File(Protocol_Record.Name,
                                        Protocol_Record.Used);

                      Protocols_T.Next(Cursor);
        end loop;

    end Write_Protocol_Mapping_Files;

begin

    Add_Protocol("GENERAL
");

end GDB_PROTOCOL;
```

Gdbalt_alternatives_s

```
-- ------------------------------------------------------------------------
--
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--
-- ------------------------------------------------------------------------
--
--  Package Name : GDBALT_ALTERNATIVES
--
--
--  Modification History :
--  Date      Name          PCR       Description
--  ----      ----          --------  -----------
--  3/24/97   D.Bulpett     PCR 469   Sort by alternative name
--  7/16/97   K.Adams       PCR 469   Add instance counter to alternative record
--  6/14/99   D.DiBiaso     PCR X025  Change instance counter from short to long natural
--  03/20/09  R.Brown          TBD       Replace binary tree with hash table (Ada2005)
--
================================================================================================
=========

with Gdbbas_Basic_Types;
use Gdbbas_Basic_Types;
with Gdbcla_Classes;
with ada.containers.hashed_maps;
with Ada.Strings.hash;

package Gdbalt_Alternatives is

   package Gdbbas   renames Gdbbas_Basic_Types;
   package Gdbcla   renames Gdbcla_Classes;

   type Alternative_Record_T is
      record
         Alt_Name      :   Gdbbas.Name_t := (others=>' ');
         Location_Id   :   Gdbbas.Instance_Id_T
                              := Gdbbas.Null_Instance_Id;
         Bit_Offset    :   Gdbbas.Bit_Offset_T
                              :=   Gdbbas.Default_Bit_Offset;
         Instance_Ctr  :   Gdbbas.Long_Natural_T
```

```
                              := Gdbbas.Long_Natural_T'first;
      Of_Class       :    Gdbcla.Gdbcli.Map;
    end record;


  package Gdbalt_Alternative_Table is new ada.containers.Hashed_Maps
       (Key_Type => Gdbbas.Name_t,
        Element_Type => Alternative_Record_T,
        Hash => Ada.Strings.Hash,
        Equivalent_Keys => "=");

  package gdbaltt renames Gdbalt_Alternative_Table;

  procedure Write_Table_To_File ( The_table :in gdbaltt.map );

end Gdbalt_Alternatives;
```

Gdbalt_alternatives_b

```
-------------------------------------------------------------------------------
--
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
-------------------------------------------------------------------------------
-- D.Bulpett     03/24/97  PCR 469, Sort by alternative name
-- F.Kreimendahl 04/08/97  PCR 469 - Add attribute info to Alternative_File_Record_T
-- K. Adams      07/17/97  PCR 469 - Add instance counter to alternative record
-- D.DiBiaso     06/14/99  PCR X025- Change instance counter from short to long natural
-- I.Charny           04/17/09  TBD              Changed binary trees to hash tables in
Write_Table_To_File
-------------------------------------------------------------------------------

with Gdbft_Ground_Data_File_Types;

package body Gdbalt_Alternatives is

   package gdbft renames Gdbft_Ground_Data_File_Types;
   use gdbft;
   use gdbbas;
   use gdbcla;

-----------------------
-- internal subprograms
-----------------------

   procedure Write_Alternative_Node(the_node :in Alternative_Record_T) is

      The_record : Alternative_File_Record_T
          := (Alt_Name    => the_node.Alt_name,
              Bit_Offset  => the_node.bit_offset,
              Instance_Ctr=> the_node.Instance_Ctr,
              Location_Id => the_node.location_id,
              Class_name  => (others => ' '),
              others      => gdbbas.null_index);

      Class_info : gdbcla.Class_Record_t
        := gdbcla.gdbcli.Element(the_node.Of_Class, the_node.Alt_Name);

   begin

      The_record.Class_name       := Class_info.Name;
      The_record.Attributes_start := Class_info.Attributes_start;
      The_record.Attributes_stop  := Class_info.Attributes_stop;

      Diralt.Write(File => Alternative_File,
                   Item => The_record);

   end Write_Alternative_Node;
```

```
----------------------
-- visible subprograms
----------------------

   function "<" (x,y: in Alternative_Record_T) return boolean is
   begin
      return (x.Alt_Name < y.Alt_Name);
   end;

      procedure Store_Table_Element ( Position: Gdbaltt.Cursor )
   is
      node : Alternative_Record_T;
      use Gdbaltt; -- for operator
   begin
      if Position /= Gdbaltt.No_Element then
         node := Gdbaltt.Element(Position => Position);
         Write_Alternative_Node (node);
      end if;
   end;


   procedure Write_Table_To_File ( The_table : in Gdbaltt.map )
   is
   begin
      Gdbaltt.Iterate(The_Table, Store_Table_Element'Access);
   end;

end Gdbalt_Alternatives;
```

Gdbatt_attributes_s

```
------------------------------------------------------------------------------
--
-- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--
------------------------------------------------------------------------------
--
-- Package Name: GDBATT_ATTRIBUTES
--
--
-- Modification History:
--
--     Date      Name      PCR   Description
--     ----      ----      ---   -----------
--     01/11/00  D.DiBiaso x203  Added protocol info field to attribute info
--                               record
--
--     03/20/09 R. Brown   TBD   Replace binary tree with hash maps (Ada2005)
------------------------------------------------------------------------------

with gdbbas_basic_types;
use Gdbbas_Basic_Types;
with gdbudt_user_data_types;
with Ada.Containers.Hashed_maps;
with Ada.Strings.Hash;


package Gdbatt_Attributes is

  package gdbbas renames gdbbas_basic_types;
  package gdbudt renames gdbudt_user_data_types;

    type Attribute_Record_T is
      record
         Name                :   Gdbbas.Name_t := (others=>' ');
         Primitive_Type      :   Gdbbas.Primitives
                                 := Gdbbas.Primitives'( Gdbbas.Undefined );
         User_Data_Type      :   gdbudt.Gdbuti.Cursor := Gdbudt.Gdbuti.No_element;
```

```
         Array_Size          :   Gdbbas.User_Array_Range_T
                                 :=   0;
         Write_Flag          :   Gdbbas.Write_Flag_T
                                 := Gdbbas.Default_Write_Flag;
         Machine_Type        :   Gdbbas.Machine_Type_T
                                 := Gdbbas.Default_Machine_Type;
         Offset              :   Gdbbas.Attribute_Offset_T
                                 := Gdbbas.Null_Attribute_Offset;
         Size                :   Gdbbas.Attribute_Offset_T
                                 := Gdbbas.Null_Attribute_Offset;
         Index               :   gdbbas.File_Index_T:= gdbbas.Null_Index;
         Protocol_Info       :   Gdbbas.Protocol_Info_t := (others=>' ');
      end record;


      package gdbati_attribute_table is new Ada.Containers.Hashed_Maps
         (Key_Type => Gdbbas.Name_t,
          Element_Type => Attribute_Record_T,
          Hash => Ada.Strings.Hash,
          Equivalent_Keys => "=");

       package gdbati renames gdbati_attribute_table;


      impossible_condition : exception;

       procedure Write_Table_To_File ( The_table :in gdbati.map );

end gdbatt_attributes;
```

Gdbatt_attributes_b

```
-------------------------------------------------------------------------------
--
-- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--
-------------------------------------------------------------------------------
--
-- Package Name: GDBATT_ATTRIBUTES
--
--
-- Modification History:
--
--      Date       Name       PCR   Description
--      ----       ----       ---   -----------
--      01/11/00   D.DiBiaso  x203  Added protocol info field to attribute info
--                                  record
--
--      03/20/09 R. Brown     TBD   Replace binary tree with hash maps (Ada2005)
-------------------------------------------------------------------------------

with Gdbft_Ground_Data_File_Types;
package body Gdbatt_Attributes is

      package gdbft renames Gdbft_Ground_Data_File_Types;

      use gdbft;
      use gdbbas;   -- for name_t and file_index ops

  -- internal procedures
      procedure write_attribute_node ( node : in out attribute_record_t )
      is
            The_Record    :   Attribute_File_Record_T;
            UDT_Data      :   gdbudt.User_Data_type_record_t;

      begin

            -- User_Data_Type_Index is set only if Primitive_Type
            -- is user-defined.  Currently, string and enumeration
```

```
            -- are the only user-defined types.
      The_Record := (
            Name => Node.Name,
            Primitive_type => Node.Primitive_Type,
            User_Data_Type_Index => gdbbas.null_index,
            Array_Size => Node.Array_Size,
            Write_Flag => Node.Write_Flag,
            Machine_Type => Node.Machine_Type,
            Attribute_Offset => Node.Offset,
            Attribute_Size => Node.Size);

        if (node.primitive_type = gdbbas.undefined ) then
          udt_data := gdbudt.gdbuti.Element(node.user_data_type);
          the_record.user_data_type_index := udt_data.index;
          the_record.primitive_type := udt_data.primitive_type;
        end if;

         -- The attribute offset is written to the file, since the attributes
         -- are written out in alphabetical order (to support search).
         -- There is no other indication of input order, and it also
         -- saves recalculating the offset.


         -- Fill in the file index for this Attribute, so it can be used later
         -- for Tier1 list processing

      Node.Index :=
         gdbbas.File_Index_T (Dirati.Index( File => Attribute_File ));

      Dirati.Write(
            File   =>   Attribute_File,
            Item   =>   The_Record );


      end write_attribute_node;

-- visible subprograms

      function "<" (x,y : in attribute_record_t) return boolean
      is
      begin
          return (x.name < y.name);
      end;

   procedure Store_Table_Element ( Position: gdbati.Cursor )
   is
      node : attribute_record_t;
      use gdbati; -- for operator
   begin
      if Position /= gdbati.No_Element then
         node := gdbati.Element(Position => Position);
         write_attribute_node (node);
      end if;
   end;


   procedure Write_Table_To_File ( The_table : in gdbati.map )
   is
   begin
      gdbati.Iterate(The_Table, Store_Table_Element'Access);
   end;

end gdbatt_attributes;
```

Gdbcla_classes_s

```
------------------------------------------------------------------------------
--
-- COPYRIGHT (C) C.S.Draper Lab, Inc. 2009.  All Rights Reserved.
```

```
--
-------------------------------------------------------------------------------
--
-- Package Name: GDBCLA_CLASSES
--
-- Modification History:
--
--     Date      Name      PCR   Description
--     ----      ----      ---   -----------
--     01/17/00  D.DiBiaso x203  Added protocol field to class info record
--
--     03/20/09  R.Brown    TBD  replace binary tree with hash table (ada2005)
-------------------------------------------------------------------------------

with Gdbbas_Basic_Types;
with gdbudt_user_data_types;
with gdbatt_attributes;
with ada.containers.hashed_maps;
with Ada.Strings.hash;

package gdbcla_classes is

   package Gdbbas  renames Gdbbas_Basic_Types;
   package gdbudt  renames gdbudt_user_data_types;
   package gdbatt  renames gdbatt_attributes;

  --internal representation
   type Class_Record_T is
      record
         Name              :   Gdbbas.Name_T             := ( others => ' ' );
         Protocol          :   Gdbbas.Name_T             := ( others => ' ' );
         User_Data_Types   :   Gdbudt.Gdbuti.map;
          Attributes       :   Gdbatt.Gdbati.map;
         Attributes_Start  :   Gdbbas.File_Index_t       := gdbbas.null_index;
         Attributes_stop   :   Gdbbas.File_Index_t       := gdbbas.null_index;
         Last_Offset       :   Gdbbas.Attribute_Offset_T := Gdbbas.Null_Attribute_Offset;
      end record;


   package gdbcli_class_Table is new ada.containers.hashed_maps
         (Key_Type => Gdbbas.Name_t,
          Element_Type => Class_Record_T,
          Hash => Ada.Strings.Hash,
          Equivalent_Keys => "=");


   package gdbcli renames gdbcli_class_Table;


   impossible_condition : exception;

   procedure Write_Table_To_File ( The_table: in gdbcli.map );

end gdbcla_classes;
```

Gdbcla_classes_b

```
-------------------------------------------------------------------------------
--
-- COPYRIGHT (C) C.S.Draper Lab, Inc. 2009.  All Rights Reserved.
--
-------------------------------------------------------------------------------
--
-- Package Name: GDBCLA_CLASSES
--
-- Modification History:
--
--     Date      Name      PCR   Description
```

```
--      ----       ----       ---    -----------
--      01/17/00   D.DiBiaso x203   Added protocol field to class info record
--
--      03/20/09   R.Brown     TBD   replace binary tree with hash table (ada2005)
-------------------------------------------------------------------------------

with Gdbft_Ground_Data_File_Types;
package body gdbcla_classes is

    package gdbft renames Gdbft_Ground_Data_File_Types;
    use gdbft;
    use gdbbas;

   --internal procedures
      procedure write_class_node ( the_node : in out Class_record_t )
      is
        Att_start, Att_stop         : gdbbas.file_index_t;
        String_start, String_stop   : gdbbas.file_index_t;
        Enum_start, enum_stop       : gdbbas.file_index_t;
        the_file_record             : class_file_record_t :=
                                        ( name => the_node.name,
                                          others=>gdbbas.null_index );

      begin
         the_file_record.user_strings_start
           := gdbbas.file_index_t(gdbft.dirusi.index (gdbft.User_strings_file));
         the_file_record.user_enums_start
           := gdbbas.file_index_t(gdbft.diruei.index (gdbft.User_enumerations_file));

         gdbudt.Write_Table_To_File(udt_table => the_node.User_data_types );

         the_file_record.user_strings_stop
            := gdbbas.file_index_t(gdbft.dirusi.index (gdbft.User_strings_file))-1;
         the_file_record.user_enums_stop
            := gdbbas.file_index_t(gdbft.diruei.index (gdbft.User_enumerations_file))-1;

         the_file_record.attributes_start
           := gdbbas.file_index_t ( gdbft.dirati.index ( gdbft.attribute_file ));
          gdbatt.write_table_to_file (the_node.attributes);
         the_file_record.attributes_stop
           := gdbbas.file_index_t ( gdbft.dirati.index ( gdbft.attribute_file ))-1;

         the_node.attributes_start := the_file_record.attributes_start;
         the_node.attributes_stop  := the_file_record.attributes_stop;

         Dircli.Write(
            File   =>   Class_File,
            Item   =>   the_file_record);
      end write_class_node;

   --visible procedures

   procedure Store_Table_Element ( Position: gdbcli.Cursor )
   is
      node : Class_record_t;
      --exists : boolean;
      use gdbcli; -- for operator
   begin
      if Position /= gdbcli.No_Element then
         node := gdbcli.Element(Position);
         write_class_node (node);
         --exists := gdbcli.contains(Container => gdbcli.Map,
         --                          Key => node.name);
        --if exists = false then
        --    raise impossible_condition;
        --end if;
      end if;
   end;
```

```
   procedure Write_Table_To_File ( The_table : in gdbcli.map )
   is
   begin
      gdbcli.Iterate(The_table, Store_Table_Element'Access);
   end;

end gdbcla_classes;
```

Gdbcom_commands_s

```
-- ------------------------------------------------------------------------
--
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--
-- ------------------------------------------------------------------------
--
--  Package Name : GDBCOM_COMMANDS
--
--
--  Modification History :
--  Date      Name          PCR       Description
--  ----      ----          --------  -----------
--  01/15/03  D.DiBiaso     X517      Added Num_Data_Object_Params to command record
--
--   03/20/09  R. Brown       TBD      Replace binary tree with hash table (Ada2005)
-- ------------------------------------------------------------------------

with Gdbbas_Basic_types;
with gdbudt_user_data_types;
with gdbprm_parameters;
with ada.containers.hashed_maps;
with Ada.Strings.Hash;

package Gdbcom_commands is

    package gdbbas  renames  Gdbbas_Basic_types;
    package gdbudt  renames  gdbudt_user_data_types;
    package gdbprm  renames  gdbprm_parameters;

    type Command_Record_T is
        record
         Name                  :   gdbbas.name_t := (others=>' ');
         Machine_Type          :   gdbbas.Machine_Type_T
                                   := gdbbas.Default_Machine_Type;
         Command_Shell         :   gdbbas.Command_Shell_T
                                   := gdbbas.Default_Command_Shell;
         User_Data_Types       :   Gdbudt.Gdbuti.map;
          Parameters           :   gdbprm.Gdbpri.map;
         Last_Offset           :   Gdbbas.Parameter_Offset_T
                                   := Gdbbas.Null_Parameter_Offset;
         File_Index            :   gdbbas.File_Index_T
                                   :=   gdbbas.Null_Index;
         Num_Data_Object_Params :  gdbbas.Short_Natural_T := 0;
      end record;

    package gdbcmi_commands_table is new ada.Containers.hashed_maps
         (Key_Type => Gdbbas.Name_t,
          Element_Type => Command_Record_T,
          Hash => Ada.Strings.Hash,
          Equivalent_Keys => "=");


      package gdbcmi renames gdbcmi_commands_table;

   procedure Write_Table_To_File ( The_table : in gdbcmi.map );

end Gdbcom_commands;
```

Gdbcom_commands_b

```
-- ------------------------------------------------------------------------
--
--    COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.   All Rights
Reserved.
--
-- ------------------------------------------------------------------------
--
--  Package Name : GDBCOM_COMMANDS
--
--
--  Modification History :
--  Date      Name          PCR       Description
--  ----      ----          --------  -----------
--  01/15/03  D.DiBiaso     X517      Added Num_Data_Object_Params to command record
--
--  03/20/09  R. Brown       TBD      Replace binary tree with hash table (Ada2005)
-- ------------------------------------------------------------------------

with Gdbft_Ground_Data_File_Types;
package body gdbcom_commands is


      package gdbft renames Gdbft_Ground_Data_File_Types;



      use gdbft;

      use gdbbas;          -- for name_t ops


-- internal subprograms

      procedure write_command_node (node : in command_record_t)

      is

       The_Record : Command_File_Record_T := (Name => Node.name,

                                              Machine_Type => Node.Machine_Type,
                                              Command_Shell =>
                                                Node.Command_Shell,
                                              others => gdbbas.null_index);

   begin

      the_record.user_strings_start

           := gdbbas.file_index_t(gdbft.dirusi.index(gdbft.User_strings_file));

      the_record.user_enums_start

           := gdbbas.file_index_t(gdbft.diruei.index(
                                              gdbft.User_enumerations_file));


       Gdbudt.Write_Table_to_file ( Udt_table => Node.User_Data_Types);


      the_record.user_strings_stop

            := gdbbas.file_index_t(gdbft.dirusi.index (gdbft.User_strings_file))-1;
```

```
      the_record.user_enums_stop

            := gdbbas.file_index_t(gdbft.diruei.index (gdbft.User_enumerations_file))-1;




      The_Record.Parameters_start :=

            gdbbas.File_Index_T(

               gdbft.Dirpri.Index( File => gdbft.Parameter_File ));


      Gdbprm.Write_Table_To_File( Node.parameters );


      The_Record.Parameters_Stop :=

            gdbbas.File_Index_T(

               gdbft.Dirpri.Index( File => gdbft.Parameter_File ))-1;


      Dircmi.Write(

            File   =>   Command_File,

            Item   =>   The_Record );


   end Write_Command_Node;

   procedure Store_Table_Element ( Position: Gdbcmi.Cursor )
   is
      use Gdbcmi; -- for operator
   begin
      if Position /= Gdbcmi.No_Element then
         Write_Command_Node (Gdbcmi.Element(Position));
      end if;
   end;


   procedure Write_Table_To_File ( The_table : in gdbcmi.map )
   is
   begin
      Gdbcmi.Iterate(The_Table, Store_Table_Element'Access);
   end;

end Gdbcom_commands;
```

Gdbins_instances_s

```
-- ------------------------------------------------------------------------
--  COPYRIGHT (C) C.S.Draper Lab, Inc., 2009.  All Rights Reserved.
--
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--
-- ------------------------------------------------------------------------
-- F.Kreimendahl 3/05/97  PCR 469,  Add Alt_Name to Instance_Record_T
-- D.DiBiaso     6/15/99  PCR X025  Change instance counter from short to long natural
-- D.DiBiaso     01/17/00 PCR X203  Added protocol info field to instance info record
-- R. Brown       03/20/09  TBD      Replace binary tree with hash table (ada2005)
```

```
------------------------------------------------------------------------


with Gdbbas_Basic_Types;
with Gdbcla_Classes;
with ada.containers.hashed_maps;
with Ada.Strings.hash;

package Gdbins_Instances is

   package Gdbbas   renames Gdbbas_Basic_Types;
   package Gdbcla   renames Gdbcla_Classes;

   type Instance_Record_T is
      record
         Name          :   Gdbbas.Name_t := (others=>' ');
         Alt_Name      :   Gdbbas.Name_t := (others=>' ');
         Location_Id   :   Gdbbas.Instance_Id_T
                              :=   Gdbbas.Null_Instance_Id;
         Bit_Offset    :   Gdbbas.Bit_Offset_T
                              :=   Gdbbas.Default_Bit_Offset;
         Instance_Ctr  :   Gdbbas.Long_Natural_T
                              := Gdbbas.Long_Natural_T'first;
         Of_Class      :   Gdbcla.Gdbcli.Cursor := Gdbcla.Gdbcli.No_Element;
         Protocol_Info :   Gdbbas.Protocol_Info_t := (others=>' ');
      end record;

   package gdbini_Instance_Table is new ada.containers.hashed_maps
         (Key_Type => Gdbbas.Name_t,
          Element_Type => Instance_Record_T,
          Hash => Ada.Strings.Hash,
          Equivalent_Keys => "=");

   package gdbini renames gdbini_Instance_Table;

   procedure Write_Table_To_File ( The_table :in gdbini.map );

end Gdbins_Instances;
```

Gdbins_instances_b

```
-- F.Kreimendahl 3/05/97    PCR 469 - Add Alt_Name to The_record
-- I.Charny          04/17/09   TBD              Changed binary trees to hash tables
in Write_Table_To_File

with Gdbft_Ground_Data_File_Types;

package body Gdbins_Instances is

   package gdbft renames Gdbft_Ground_Data_File_Types;
   use gdbft;
   use gdbbas;

-- internal subprograms

   procedure write_instance_node (the_node :in instance_record_t)
   is
      The_record : Instance_File_Record_t :=
          (Name => the_node.name,
           Alt_Name => the_node.alt_name,
           Location_Id => the_node.location_id,
          Bit_Offset => the_node.bit_offset,
          Instance_Ctr => the_node.instance_ctr,
           Class_name => (others => ' '),
           others => gdbbas.null_index);
      Class_info : gdbcla.Class_Record_t
           := gdbcla.gdbcli.Element(the_node.Of_Class);
   begin
      The_record.class_name := Class_info.name;
```

```
      The_record.attributes_start := Class_info.attributes_start;
      The_record.attributes_stop  := Class_info.attributes_stop;

      Dirini.Write(
            File       =>   Instance_File,
            Item       =>   The_record );
   end Write_Instance_Node;



--visible subprograms

  procedure Store_Table_Element ( Position: gdbini.Cursor )
   is
      use gdbini; -- for operator
   begin
      if Position /= gdbini.No_Element then
         write_instance_node (gdbini.Element(Position));
      end if;
   end;


   procedure Write_Table_To_File ( The_table : in gdbini.map )
   is
   begin
      gdbini.Iterate(The_Table, Store_Table_Element'Access);
   end;

end Gdbins_Instances;
```

Gdbprm_parameters_s

```
-- ------------------------------------------------------------------------
--
--  COPYRIGHT (C) C.S.Draper Lab, Inc., 2009.  All Rights Reserved.
--
-- ------------------------------------------------------------------------
--
--  Package Name : GDBPRM_PARAMETERS
--
--
--  Modification History :
--  Date      Name          PCR       Description
--  ----      ----          --------  -----------
--
--  03/20/09  R. Brown       TBD      Replace binary tree with hash table (Ada2005)
-- ------------------------------------------------------------------------

with Gdbbas_Basic_types;
with gdbudt_user_data_types;
with ada.containers.hashed_maps;
with Ada.Strings.Hash;

package Gdbprm_parameters is
   package gdbbas renames gdbbas_basic_types;
   package gdbudt renames gdbudt_user_data_types;

   type Parameter_Record_T is
      record
         Name                :   Gdbbas.Name_t := (others=>' ');
         Primitive_Type      :   Gdbbas.Primitives
                                    := Gdbbas.Primitives'( gdbbas.Undefined );
          User_Data_Type     :    gdbudt.Gdbuti.Cursor := gdbudt.Gdbuti.No_element;
         Array_Size          :   Gdbbas.User_Array_Range_T
                                    :=   0;
         Byte_Offset         :   Gdbbas.Byte_Offset_T
                                    := Gdbbas.Default_Byte_Offset;
```

```
         Bit_Offset          :   Gdbbas.Bit_Offset_T
                                    := Gdbbas.Default_Bit_Offset;
         Offset              :   Gdbbas.Parameter_Offset_T
                                    := Gdbbas.Null_Parameter_Offset;
         Size                :   Gdbbas.Parameter_offset_t;
         Index               :   Gdbbas.File_Index_T
                                    := Gdbbas.Null_Index;
      end record;

   package gdbpri_parameters_table is new ada.containers.hashed_maps
         (Key_Type => Gdbbas.Name_t,
          Element_Type => Parameter_Record_T,
          Hash => Ada.Strings.Hash,
          Equivalent_Keys => "=");

      package gdbpri renames gdbpri_parameters_table;


   procedure Write_Table_To_File ( The_table : in gdbpri.map );

end gdbprm_parameters;
```

Gdbprm_parameters_b

```
-- ------------------------------------------------------------------------
--
--  COPYRIGHT (C) C.S.Draper Lab, Inc., 2009.  All Rights Reserved.
--
-- ------------------------------------------------------------------------
--
--  Package Name : GDBPRM_PARAMETERS
--
--
--  Modification History :
--  Date      Name          PCR       Description
--  ----      ----          --------  -----------
--
--  03/20/09  R. Brown       TBD      Replace binary tree with hash table (Ada2005)
-- ------------------------------------------------------------------------

with Gdbft_Ground_Data_File_Types;
package body Gdbprm_parameters is

      package gdbft renames Gdbft_Ground_Data_File_Types;
      use gdbft;
      use gdbbas;


      procedure write_parameter_node (the_node : in parameter_record_t)
      is
         UDT_info : gdbudt.user_data_type_record_t;
         The_record : parameter_file_record_t :=
            (Name => the_node.name,
             Primitive_Type => the_node.primitive_type,
             User_Data_Type_Index => gdbbas.null_index,
             Array_Size => the_node.array_size,
            Byte_Location => the_node.byte_offset,
            Bit_Location => the_node.bit_offset,
             Parameter_Offset => the_node.offset,
             Parameter_Size => the_node.size);
      begin
        if (the_node.primitive_type = gdbbas.undefined ) then
          udt_info := gdbudt.gdbuti.Element(the_node.user_data_type);
          the_record.user_data_type_index := udt_info.index;
          the_record.primitive_type := udt_info.primitive_type;
        end if;
```

```
        dirpri.write ( file=>parameter_file, item => the_record );
      end;

-- visible procedures

   procedure Store_Table_Element ( Position: gdbpri.Cursor )
   is
      use gdbpri; -- for operator
   begin
      if Position /= gdbpri.No_Element then
         write_parameter_node (gdbpri.Element(Position));
      end if;
   end;


   procedure Write_Table_To_File ( The_table : in gdbpri.map )
   is
   begin
      gdbpri.Iterate(The_Table, Store_Table_Element'Access);
   end;

end gdbprm_parameters;
```

Gdbudt_user_data_types_s

```
-- ------------------------------------------------------------------------
--
--  COPYRIGHT (C) C.S.Draper Lab, Inc., 2009.  All Rights Reserved.
--
-- ------------------------------------------------------------------------
--
--  Package Name : GDBudt_User_data_types
--
--
--  Modification History :
--  Date      Name          PCR       Description
--  ----      ----          --------  -----------
--
--  05/06/09  I.Charny       TBD      Replaced binary tree with hash table (Ada2005)
-- ------------------------------------------------------------------------
with Gdbbas_Basic_Types;
with ada.containers.hashed_maps;
with Ada.Strings.Hash;

Package Gdbudt_User_data_types is

   -- Internal representation (Hash Table)
   package Gdbbas  renames Gdbbas_Basic_Types;

   type Enumeration_Pair_Record_T is
      record
         Name           :   Gdbbas.Name_T                        :=   ( others => ' ' );
         Value                                  :         Gdbbas.User_Enumeration_Position_T      :=
Gdbbas.Null_Enumeration_Position;
         Internal_Code  :   Gdbbas.User_Internal_Code_T         := Gdbbas.Null_Internal_Code;
      end record;

   package Gdbeni_Enumerations_Table is new ada.containers.hashed_maps
      (Key_Type => Gdbbas.Name_t,
       Element_Type => Enumeration_Pair_Record_T,
       Hash => Ada.Strings.Hash,
       Equivalent_Keys =>"=");

   package Gdbeni renames Gdbeni_enumerations_Table;

   type User_Data_Type_Record_T (Primitive_Type:Gdbbas.User_Data_Type_Category
```

```
                                := Gdbbas.User_Data_Type_Category'( Gdbbas.Of_String ))
   is
      record
         Name: Gdbbas.Name_T       := ( others => ' ' );
         Index:Gdbbas.File_Index_T := Gdbbas.Null_Index;
         case Primitive_Type is
            when Gdbbas.User_Data_Type_Category'( Gdbbas.Of_String ) =>
               String_Length: Gdbbas.User_String_Range_T := Gdbbas.User_String_Range_T'first;

            when Gdbbas.User_Data_Type_Category'( Gdbbas.Of_Enumeration ) =>
               Enumeration_Pairs  :   Gdbeni.map ;
               Last_Value                        :        Gdbbas.User_Enumeration_Position_T   :=
Gdbbas.Null_Enumeration_Position;
         end case;
      end record;

   package Gdbuti_User_Data_Type_Table is new ada.containers.hashed_maps
      (Key_Type => Gdbbas.Name_t,
       Element_Type => User_Data_Type_Record_t,
       Hash => Ada.Strings.Hash,
       Equivalent_Keys =>"=");

   package gdbuti renames Gdbuti_User_Data_Type_Table;

   impossible_condition : exception;

   procedure Write_Table_To_File ( UDT_table : in gdbuti.map );

end GDBUDT_User_data_types;
```

Gdbudt_user_data_types_b

```
-- ------------------------------------------------------------------------
--
--  COPYRIGHT (C) C.S.Draper Lab, Inc., 2009.  All Rights Reserved.
--
-- ------------------------------------------------------------------------
--
--  Package Name : GDBudt_User_data_types
--
--
--  Modification History :
--  Date      Name          PCR       Description
--  ----      ----          --------  -----------
--
--   05/06/09   I.Charny          TBD        Replaced binary tree with hash table in
Write_Enumeration_Pair_Table,
--                                                          write_udt_node,       and
Store_Table_Element
-- ------------------------------------------------------------------------

with Gdbft_Ground_Data_File_Types;
package body Gdbudt_User_data_types is

    package gdbft renames Gdbft_Ground_Data_File_Types;

     use gdbft;
     use gdbbas_basic_types;    -- for name_t'"<"


  -- internal procedures

     procedure Write_Enumeration_Pair_Node (Node : in Enumeration_Pair_Record_t) is
        The_Record : Enumeration_Pair_File_Record_T;
     begin

      The_Record := ( Name => Node.Name,
```

```
                Position => Node.Value,
                Internal_Code => Node.Internal_Code );

 Direni.Write(
       File   =>   Enumeration_Pair_File,
       Item   =>   The_Record );

end Write_Enumeration_Pair_Node;

  procedure Write_Enumeration_Pair_Table ( The_table : in gdbeni.map )
  is
        Cursor : gdbeni.Cursor := gdbeni.First(The_table);
        Node : Enumeration_Pair_Record_t;
        use gdbeni;
  begin
        while (Cursor /= gdbeni.No_Element) loop
              node := gdbeni.Element(Cursor);
              Write_Enumeration_Pair_Node ( node );
              gdbeni.Next(Cursor);
        end loop;
  end Write_Enumeration_Pair_Table;

procedure write_udt_node (the_node: in out user_data_type_record_t)
is
 begin
    -- This procedure copies data from the a user data type
    -- node into a file record, writes the record, and then
    -- updates the node with the file index to what was written.

   case The_Node.Primitive_Type is

   when Gdbbas.User_Data_Type_Category'( Gdbbas.Of_String ) =>
   declare
    The_record : User_String_Type_File_Record_T;
   begin
    The_Record := (
       Primitive_Type  =>Gdbbas.User_Data_Type_Category'(Gdbbas.Of_String ),
       Name  =>  The_Node.Name,
       String_Length  =>  The_Node.String_Length );
       The_Node.Index :=
         Gdbbas.File_Index_T(
             Dirusi.Index( File => User_Strings_File ));

       Dirusi.Write(File   => User_Strings_file,
                    Item   =>   The_Record );
    end;

 when Gdbbas.User_Data_Type_Category'( Gdbbas.Of_Enumeration ) =>
   declare
     The_record : User_Enumeration_Type_File_Record_T;
     Starting_index : gdbbas.file_index_t;
   begin
    Starting_Index  :=
         Gdbbas.File_Index_T(
             Direni.Index( File => Enumeration_Pair_File ));

    Write_Enumeration_Pair_Table (The_table => The_Node.Enumeration_Pairs );

    The_Record := (
       Primitive_Type  =>
         Gdbbas.User_Data_Type_Category'(Gdbbas.Of_Enumeration ),
       Name  =>  The_Node.Name,
       Enumeration_Type_Start  =>  Starting_Index,
       Enumeration_Type_Stop  =>
         Gdbbas.File_Index_T(Direni.Index( File => Enumeration_Pair_File )) - 1 );

       -- update udt node with file index for later use by attribute write
       --
       The_Node.Index :=
```

```
            Gdbbas.File_Index_T(Diruei.Index( File => User_Enumerations_File ));

          Diruei.Write(File   => User_Enumerations_file,
                       Item   =>   The_Record );
        end;
     end case;

    end write_udt_node;


-- visible procedures

   procedure Store_Table_Element ( Position: gdbuti.Cursor )
   is
      node : user_data_type_record_t;
      --exists : boolean;
      use gdbuti; -- for operator
   begin
      if Position /= gdbuti.No_Element then
         node :=  gdbuti.Element(Position);
         write_udt_node (node);
         --exists := gdbuti..contains(Container => UDT_table,
         --                           Key => node.name);
        --if exists = false then
        --    raise impossible_condition;
        --end if;
      end if;
   end;


   procedure Write_Table_To_File ( UDT_table : in gdbuti.map )
   is
   begin
      gdbuti.Iterate(UDT_table, Store_Table_Element'Access);
   end;

end gdbudt_user_data_types;
```

Gdbbpt_basic_parser_interface_s

```
-- ------------------------------------------------------------------------
--
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--
-- ------------------------------------------------------------------------
--
--  Package Name : GDBBPT_BASIC_PARSER_TO_TRANSFORM_INTERFACE
--
--
--  Modification History :
--  Date      Name          PCR       Description
--  ----      ----          --------  -----------
--  03/05/97  F.Kreimendahl PCR 469   Add Alternate_Name to Add_Instance interface
--  11/19/98  D.DiBiaso     PCR 064   Make function Lookup_Primitive_Type available in spec
--  06/01/99  D.DiBiaso     PCR X008  Add String_OK parameter to Lookup_Primitive_Type
--                                    procedure to allow string types for overlays
--  06/03/99  D.DiBiaso     PCR X019  Add optional byte offset field to attribute
--                                    definitions
--  06/14/99  D.DiBiaso     PCR X025  Allow instance IDs to be in range 1..2147483647
--  01/17/00  D.DiBiaso     PCR x203  Added Add_Protocol procedure for storing TIL name
--                                    associated with a class definition
--  05/02/00  D.DiBiaso     PCR X229  Added Lookup_String_Type procedure
--  05/03/00  D.DiBiaso     PCR X200  Return next byte offset from add_parameter proc
--  01/22/03  D.DiBiaso     PCR X516  Added initialize procedure
```

```
--  04/17/09  I.Charny          TBD          Changed binary trees to hash tables
in Compute_Offset,
--                                                          Add_Class, Add_Protocol,
Add_String_Type, Add_Enumeration_Type,
--                                                    Add_Enumeration_Pair,
Lookup_String_Type, Add_Attribute,
--                                                      Add_Instance,
Add_Command, Add_Parameter, and Initialize
--
-- ---------------------------------------------------------------------------

with Gdbbas_Basic_Types;
with Gdbcla_Classes;
with Gdbudt_User_Data_Types;
with Gdbcom_Commands;
with Primitive_Data_Types;

package Gdbbpt_Basic_Parser_To_Transform_Interface is
   package Gdbbas   renames Gdbbas_Basic_Types;
   package Gdbcli   renames Gdbcla_Classes.gdbcli;
   package Gdbuti   renames Gdbudt_User_Data_Types.gdbuti;
   package Gdbcmi   renames Gdbcom_Commands.gdbcmi;

   -- This package contains procedures called by the parse actions
   -- during input processing.

   function Lookup_Primitive_Type (ArgType :in gdbbas.Name_T; String_OK : in boolean := False)
            return gdbbas.primitives;

   procedure Lookup_String_Type(Type_Name  : in  gdbbas.Name_T;
                                Str_Length : out gdbbas.User_String_Range_T;
                                Valid_Type : out Boolean);

   procedure Add_Class (
         Name           :   in      Gdbbas.Name_T;
         Class_Link     :   out     Gdbcli.map );

   procedure Add_Protocol ( Name       : in gdbbas.Name_T;
                            Class_Link : in gdbcli.map);

   procedure Add_String_Type (
         Name           :   in      Gdbbas.Name_T;
         Length         :   in      Gdbbas.User_String_Range_T;
          Class_Link     :   in      Gdbcli.map );

   procedure Add_String_Type (
         Name           :   in      Gdbbas.Name_T;
         Length         :   in      Gdbbas.User_String_Range_T;
         Command_Link   :   in      Gdbcmi.map );

   procedure Add_Enumeration_Type (
         Name           :   in      Gdbbas.Name_T;
          Class_Link     :   in      Gdbcli.map;
          Type_Link      :   out     Gdbuti.map);

   procedure Add_Enumeration_Type (
         Name           :   in      Gdbbas.Name_T;
          Command_Link   :   in      Gdbcmi.map;
          Type_Link      :   out     Gdbuti.map);

   procedure Add_Enumeration_Pair (
         Enum_Identifier :   in      Gdbbas.Name_T;
                             -- The above is not UIL, but the Input
                             -- Processor spec said it would
                             -- have the same constraints as a
                             -- UIL name.
         Internal_Code   :   in      Gdbbas.User_Internal_Code_T;
          Type_Link      :   in      Gdbuti.map );
```

```
   procedure Add_Attribute (
         Name              :   in      Gdbbas.Name_T;
         AttType           :   in      Gdbbas.Name_T;
         Array_Size        :   in      Gdbbas.User_Array_Range_T;
         Write_Flag        :   in      Gdbbas.Write_Flag_T;
         Machine_Type      :   in      Gdbbas.Machine_Type_T;
         Class_Link        :   in      Gdbcli.map;
         Byte_Offset_Valid :   in      Boolean;
         Byte_Offset       :   in      Gdbbas.Attribute_Offset_t;
         Protocol_Info     :   in      Gdbbas.Protocol_Info_t);

   procedure Add_Instance (
         Instance_Name   :   in      Gdbbas.Name_T;
         Alternate_Name  :   in      Gdbbas.Name_T;
         Class_Name      :   in      Gdbbas.Name_T;
         Location_Id     :   in      Gdbbas.Instance_Id_T;
         Bit_Offset      :   in      Gdbbas.Bit_Offset_T;
         Instance_Ctr    :   in      Gdbbas.long_natural_T;
         Protocol_Info   :   in      Gdbbas.Protocol_Info_T;
         Offset          :   out     Gdbbas.Attribute_Offset_T );

   procedure Add_Command (
         Name              :   in      Gdbbas.Name_T;
         Machine_Type      :   in      Gdbbas.Machine_Type_T;
         Command_Shell     :   in      Gdbbas.Command_Shell_T;
          Command_Link     :   out     Gdbcmi.map);

   procedure Add_Parameter (
         Name              :   in      Gdbbas.Name_T;
         PrmType           :   in      Gdbbas.Name_T;
         Array_Size        :   in      Gdbbas.User_Array_Range_T;
         Byte_Offset       :   in      Gdbbas.Byte_Offset_T;
         Bit_Offset        :   in      Gdbbas.Bit_Offset_T;
          Command_Link     :   in      Gdbcmi.map;
         Next_Offset       :       out Primitive_Data_Types.Integer_T);

   procedure initialize;

end Gdbbpt_Basic_Parser_To_Transform_Interface;
```

Gdbbpt_basic_parser_interface_b

```
-- ------------------------------------------------------------------------
--
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--
-- ------------------------------------------------------------------------
--
--  Package Name : GDBBPT_BASIC_PARSER_TO_TRANSFORM_INTERFACE
--
--
--  Modification History :
--  Date      Name          PCR       Description
--  ----      ----          --------  -----------
--  08/22/96  JKM3347                 Fixed string offset
--  03/05/97  F.Kreimendahl PCR 469   Add Alternate_Name to Add_Instance interface
--  06/26/97  kdd1559       PCR 478   update code to allow multiple instances to be defined
--                                    for the same CVT address.
--  08/17/98  DMD2273       PCR 020   Do not allow bit types
--  11/03/98  DMD2273       PCR 046   Add check for parameters extending beyond end of
--                                    command byte shell
--  11/19/98  DMD2273       PCR 064   Allow user to specify instance IDs
--  06/01/99  D.DiBiaso     PCR X008  Add String_OK parameter to Lookup_Primitive_Type
--                                    procedure to allow string types for overlays
--  06/03/99  D.DiBiaso     PCR X019  Add optional byte offset field to attribute
--                                    definitions
```

```
--  06/09/99  D.DiBiaso     PCR X067  Clean up error reporting messages
--  06/14/99  D.DiBiaso     PCR X025  Allow instance IDs to be in range 1..2147483647
--  06/25/99  D.DiBiaso     PCR X003  Modify add_instance to support initial values for
--                                    starred instances
--                                    Update calculation of Class_Record.Last_Offset to be
--                                    end of last attribute in object
--  01/17/00  D.DiBiaso     PCR x203  Added Add_Protocol procedure for storing TIL name
--                                    associated with a class definition
--  02/04/00  D.DiBiaso     PCR X202  Allow command parameters in entire 320 byte packet
--  05/02/00  D.DiBiaso     PCR X229  Added support for STRING### data types
--  05/03/00  D.DiBiaso     PCR X200  Added support for defined data type names for attributes
--                                    and command parameters; return next byte offset from
--                                    add_parameter procedure
--  05/17/00  D.DiBiaso     PCR X314  Added maximum data-type based arrayness checks for
--                                    attributes and parameters; max object size check for
--                                    attribute offsets
--  01/15/03  D.DiBiaso     PCR X517  Added data_object primitive data type
--  01/22/03  D.DiBiaso     PCR X516  Added initialize procedure
--  04/17/09  I.Charny          TBD          Changed binary trees to hash tables
in Compute_Offset,
--                                                          Add_Class, Add_Protocol,
Add_String_Type, Add_Enumeration_Type,
--                                                       Add_Enumeration_Pair,
Lookup_String_Type, Add_Attribute,
--                                                         Add_Instance,
Add_Command, Add_Parameter, and Initialize
--
-- ------------------------------------------------------------------------

with GDB_Timeliner_Interface_Types;
with CDH_Command_Types;
with Gdbcla_Classes;
with Gdbudt_User_Data_Types;
with Gdbatt_Attributes;
with Gdbins_Instances;
with Gdbcom_Commands;
with Gdbprm_Parameters;
with Gdberr_Error_Reporting;
with Gdbses_Input_Processor_Session;
with Gdbivi_Instance_Id_Intervals;
with Case_conversion; use case_conversion;
with Gdb_Options;
with Gdb_Initial_Values;
with Gdb_Definitions;

package body Gdbbpt_Basic_Parser_To_Transform_Interface is

   package Gdbcla   renames Gdbcla_Classes;
   package Gdbudt   renames Gdbudt_User_Data_Types;
   package Gdbatt   renames Gdbatt_Attributes;
   package Gdbins   renames Gdbins_Instances;
   package Gdbcom   renames Gdbcom_Commands;
   package Gdbprm   renames Gdbprm_Parameters;
   package Gdberr   renames Gdberr_Error_Reporting;
   package Gdbses   renames Gdbses_Input_Processor_Session;
   package Gdbivi   renames Gdbivi_Instance_Id_Intervals;

   --must have the following use clauses to include operators
   use gdbbas;
   use GDB_Timeliner_Interface_Types;
   use CDH_Command_Types;

   Argument_Offset_Increments   :   constant array ( gdbbas.Primitives )
         of gdbbas.Attribute_Offset_T := (
         gdbbas.Of_Byte  => 1,
         gdbbas.Of_Short_Integer  => 1,
         gdbbas.Of_Integer  => 2,
         gdbbas.Of_Natural  => 2,
         gdbbas.Of_Long_Integer  => 4,
         gdbbas.Of_Long_Natural  => 4,
```

```
         gdbbas.Of_Float  => 4,
         gdbbas.Of_Long_Float => 8,
            -- Values for string and enumeration user-defined types are
            -- looked up.  The 0 is used for the case in which the input
            -- incorrectly has "string" or "enumeration" as an attribute type.
            -- It keeps the Input Processor from blowing up.
         gdbbas.Of_String  => 0,
         gdbbas.Of_Enumeration  => 0,
         gdbbas.Of_Boolean  => 1,
         gdbbas.Of_Signed_Bit => 1,
         gdbbas.Of_Unsigned_Bit => 1,
         gdbbas.Of_Data_Object => 0,
         gdbbas.Undefined  => 0 );

   Lookup_Table : constant array (gdbbas.primitives) of gdbbas.Name_T :=
      (('B','Y','T','E', others=>' '),
       ('S','H','O','R','T','_','I','N','T','E','G','E','R', others=>' '),
       ('I','N','T','E','G','E','R', others=>' '),
       ('N','A','T','U','R','A','L', others=>' '),
       ('L','O','N','G','_','I','N','T','E','G','E','R', others=>' '),
       ('L','O','N','G','_','N','A','T','U','R','A','L', others=>' '),
       ('F','L','O','A','T', others=> ' '),
       ('L','O','N','G','_','F','L','O','A','T', others=> ' '),
       ('S','T','R','I','N','G', others => ' '),
       ('E','N','U','M','E','R','A','T','I','O','N', others => ' '),
       ('B','O','O','L','E','A','N', others => ' '),
       ('S','I','G','N','E','D','_','B','I','T', others => ' '),
       ('U','N','S','I','G','N','E','D','_','B','I','T', others => ' '),
       ('D','A','T','A','_','O','B','J','E','C','T', others => ' '),
       (others => 'Z'));

    Impossible_Condition : exception;

-------------------------------------------------------------------------
--                          Local Subprograms
-------------------------------------------------------------------------


   function Lookup_Primitive_Type (ArgType :in gdbbas.Name_T; String_OK : in boolean := False)
            return gdbbas.primitives
   is

          Type_Name : gdbbas.Name_T := Argtype;
          Type_Code : gdbbas.primitives := gdbbas.Undefined;
          Found     : Boolean;

   begin
        Found := false;
        for i in Lookup_Table'range loop
          Found :=  ( Lookup_Table(i) = Type_Name );
          if Found then
            Type_Code := i;
            exit;
          end if;
        end loop;

        if Found then
          case Type_Code is
            when gdbbas.of_String =>
               if not String_OK then
                     Type_Code := gdbbas.Undefined;
               end if;
            when gdbbas.of_Enumeration =>
                   Type_Code := gdbbas.Undefined;
            when gdbbas.of_signed_bit | gdbbas.of_unsigned_bit =>
               if not gdb_options.Supports_Bit_Type then
                 Type_Code := gdbbas.Undefined;
               end if;
            when others =>
              null;
            end case;
```

```
        else
          Type_Code := gdbbas.Undefined;
        end if;
        return Type_Code;
    end Lookup_Primitive_Type;


    procedure Compute_Offset (
                              Class_Record      : in out  Gdbcla.Class_Record_T;
                              Attribute_Record  : in out  Gdbatt.Attribute_Record_T;
                              Byte_Offset_Valid :   in      Boolean;
                              Byte_Offset       :   in      Gdbbas.Attribute_Offset_t) is

        Attribute_Size      :   gdbbas.Attribute_Offset_T;
        att_type_size_in_bytes    : gdbbas.Long_Positive_T;
        Udt_Info            :   gdbudt.User_Data_Type_Record_T;
    begin

        if Attribute_Record.Primitive_Type =
          gdbbas.Primitives'(gdbbas.Undefined) then

          -- it's a user-defined type, get the number of bytes
          udt_Info            :=            gdbudt.gdbuti.Element(Position            =>
attribute_Record.User_Data_Type);

          if Udt_Info.Primitive_Type =  gdbbas.Of_String  then
              Attribute_Size :=
                gdbbas.Attribute_Offset_T ( Udt_Info.String_Length )
                * gdbbas.Attribute_Offset_T ( Attribute_Record.Array_Size);
              --***********-- Get the number of bytes of the type.
              --***********-- Used to compute attribute_pad.  Can take out if
              --***********-- the pad is not wanted.
              --    att_type_size_in_bytes := Gdbbas.Long_Positive_T(
              --                                    Udt_Info.String_Length);
              att_type_size_in_bytes := 1;

          elsif
            Udt_Info.Primitive_Type = gdbbas.Of_Enumeration  then
              Attribute_Size :=
                gdbbas.Attribute_Offset_T(
                                    Attribute_Record.Array_Size );
              --***********-- Get the number of bytes of the type.
              --***********-- Used to compute attribute_pad.  Can take out if
              --***********-- the pad is not wanted.
              att_type_size_in_bytes := 1;

          else
              -- There are currently only two user-defined types.
              -- If got to here, there's a bug.
              --> Come back and make this an informative exception.
              raise Impossible_Condition;
          end if;  -- user-defined
        else
          case Attribute_Record.Primitive_Type is
              when  gdbbas.Of_Byte
                | gdbbas.Of_Short_Integer
                | gdbbas.Of_Integer
                | gdbbas.Of_Natural
                | gdbbas.Of_Long_Integer
                | gdbbas.Of_Long_Natural
                | gdbbas.Of_Float
                | gdbbas.Of_Long_Float
                | gdbbas.Of_Boolean =>
                  Attribute_Size
                    := Argument_Offset_Increments (
                                                Attribute_Record.Primitive_Type )
                    * gdbbas.Attribute_Offset_T(
                                              Attribute_Record.Array_Size );
                  --***********-- Get the number of bytes of the type.
                  --***********-- Used to compute attribute_pad.  Can take out if
                  --***********-- the pad is not wanted.
```

```
                att_type_size_in_bytes := Argument_Offset_Increments(

Attribute_Record.Primitive_Type);

            when gdbbas.Of_Signed_Bit | Gdbbas.Of_Unsigned_Bit =>

                if (gdbbas.Attribute_Offset_T(Attribute_Record.Array_Size)
                    mod 8) = 0 then
                    Attribute_Size := gdbbas.Attribute_Offset_T(
                                                          Attribute_Record.Array_Size   )
/ 8;
                else
                    Attribute_Size := gdbbas.Attribute_Offset_T(
                                                          Attribute_Record.Array_Size   )
/ 8 + 1;
                end if;

                --***********-- Get the number of bytes of the type.
                --***********-- Used to compute attribute_pad.  Can take out if
                --***********-- the pad is not wanted.
                -- NOTE:    Since bit offsets can be computed in instance def
                --          there is the chance that byte size for bit types will
                --          be low by 1.  This should not be a problem since
                --          other info is returned like type,arrayness,& offset.
                --          From these, a correct size can be determined.
                att_type_size_in_bytes := Argument_Offset_Increments(

Attribute_Record.Primitive_Type);

            when gdbbas.Of_String
              |  gdbbas.Of_Enumeration
              |  gdbbas.Of_Data_Object
              |  gdbbas.Undefined  =>
                raise Impossible_Condition;
        end case;
    end if;

    -- Set the attribute's size and offset, and update the
    -- class offset field.
    Attribute_Record.Size := Attribute_Size;

    if Byte_Offset_Valid then
        Attribute_Record.Offset := Byte_Offset;
    else
        --***********-- here is where a pad is added to the offset to allow
        --***********-- for types to be on correct byte boundaries.  This
        --***********-- can be taken out if the padding is not needed.  Just
        --***********-- get rid of att_type_size_in_bytes and the following
        --***********-- if-then statement.
        if (Class_Record.Last_Offset rem att_type_size_in_bytes) /= 0 then
            Attribute_Record.Offset :=
              ((Class_Record.Last_Offset/att_type_size_in_bytes)
               + 1) * att_type_size_in_bytes;
        else
            Attribute_Record.Offset := Class_Record.Last_Offset;
        end if;
    end if;
    if (Attribute_Record.Offset + Attribute_Size) > Class_Record.Last_Offset then
        Class_Record.Last_Offset := Attribute_Record.Offset + Attribute_Size ;
    end if;

exception
    when Constraint_Error =>
        raise impossible_condition;
        --          gdberr.Report_Error (gdberr.ERROR_COMP_ATTR_OFFSET);
end Compute_Offset;

procedure Compute_Offset (
      Command_Record     : in out  Gdbcom.Command_Record_T;
      Parameter_Record   : in out  Gdbprm.Parameter_Record_T ) is
```

```
      Parameter_Size        : gdbbas.Parameter_Offset_T;
      Udt_Info              : gdbudt.User_Data_Type_Record_T;
   begin

      if Parameter_Record.Primitive_Type =
           gdbbas.Primitives'( gdbbas.Undefined ) then
         -- it's a user-defined type, get the number of bytes
          udt_Info := gdbudt.gdbuti.Element(Position => Parameter_Record.User_Data_Type
);

         if Udt_Info.Primitive_Type =
            gdbbas.Primitives'( gdbbas.Of_String )  then

            Parameter_Size :=
               gdbbas.Parameter_Offset_T (
                  Udt_Info.String_Length )
               * gdbbas.Parameter_Offset_T (
                  Parameter_Record.Array_Size );

         elsif
            Udt_Info.Primitive_Type =
            gdbbas.Primitives'( gdbbas.Of_Enumeration )  then
               Parameter_Size :=
                 gdbbas.Parameter_Offset_T(
                    Parameter_Record.Array_Size );
          else
               -- There are currently only two user-defined types.
               -- If got to here, there's a bug.
               --> Come back and make this an informative exception.
            raise Impossible_Condition;
         end if;
      else
        case Parameter_Record.Primitive_Type is
          when  gdbbas.Primitives'( gdbbas.Of_Byte )
              | gdbbas.Primitives'( gdbbas.Of_Short_Integer )
              | gdbbas.Primitives'( gdbbas.Of_Integer )
              | gdbbas.Primitives'( gdbbas.Of_Natural )
              | gdbbas.Primitives'( gdbbas.Of_Long_Integer )
              | gdbbas.Primitives'( gdbbas.Of_Long_Natural )
              | gdbbas.Primitives'( gdbbas.Of_Float )
              | gdbbas.Primitives'( gdbbas.Of_Long_Float )
              | gdbbas.Primitives'( gdbbas.Of_Boolean ) =>
            Parameter_Size
             := Argument_Offset_Increments (
                   Parameter_Record.Primitive_Type )
                * gdbbas.Parameter_Offset_T(
                   Parameter_Record.Array_Size );
          when     gdbbas.Primitives'(     gdbbas.Of_Signed_Bit     )     |     gdbbas.Primitives'(
gdbbas.Of_Unsigned_Bit ) =>
            if (gdbbas.Parameter_Offset_T(Parameter_Record.Array_Size)
                                          mod 8) = 0 then
              Parameter_Size := gdbbas.Parameter_Offset_T(
                   Parameter_Record.Array_Size ) / 8;
            else
              Parameter_Size := gdbbas.Parameter_Offset_T(
                   Parameter_Record.Array_Size ) / 8 + 1;
            end if;
          when gdbbas.Primitives'( gdbbas.Of_Data_Object ) =>
              Parameter_Size := 0;
          when gdbbas.Primitives'( gdbbas.Of_String )
               | gdbbas.Primitives'( gdbbas.Of_Enumeration )
               | gdbbas.Primitives'( gdbbas.Undefined ) =>
             raise Impossible_Condition;
          end case;
      end if;

      --

         Parameter_Record.Size := Parameter_Size;
         Parameter_Record.Offset := Command_Record.Last_Offset;
         Command_Record.Last_Offset :=
```

```
            Command_Record.Last_Offset + Parameter_Size ;
      exception
        when Constraint_Error =>
raise impossible_condition;
--            gdberr.Report_Error(gdberr.ERROR_CALC_PARAM_SIZE);
   end Compute_Offset;



------------------------------------------------------------------------------
--                              Visible Subprograms
------------------------------------------------------------------------------
   procedure Add_Class (
         Name            :   in      gdbbas.Name_T;
          Class_Link       :   out     gdbcli.Map ) is
      Local_Name         : gdbbas.Name_T := Name;
      Class_Record       :   Gdbcla.Class_Record_T;
       Exists_In_Map      :   Boolean;

   begin
      To_Upper ( Local_Name );
      Class_Record.Name := Local_Name;
      Class_Record.User_Data_Types := gdbudt.gdbuti.Empty_Map;
       Class_Record.Attributes := gdbatt.gdbati.Empty_Map;
      Class_Record.Last_Offset := 0;

        -- Add the node to the class table.
       Exists_In_Map := gdbcli.Contains(Gdbses.Global_Name_Tables.Class, Local_Name);
Class_Link := Gdbses.Global_Name_Tables.Class;
      if Exists_In_Map then
         gdberr.Report_Error(gdberr.DUP_CLASS_NAME, name);
      else
         gdbcli.Insert(Container => Gdbses.Global_Name_Tables.Class,
                       Key       => Local_Name,
                       New_Item  => Class_Record);
      end if;

   end Add_Class;

   procedure Add_Protocol ( Name : in gdbbas.Name_T;
                            Class_Link : in gdbcli.Map) is
      Class_Info     : gdbcla.Class_Record_T :=
        gdbcli.Element(Container => Class_Link,
                       Key       => Name);
      table : gdbcli.Map := Class_Link;
   begin
      -- Set the protocol in the class info
      Class_Info.Protocol := Name;
      Case_Conversion.to_upper(Class_Info.Protocol);

      -- update the class_Info in the class table

      gdbcli.Insert(Container => table,
                    Key       => Class_Info.Name,
                    New_Item  => Class_Info);
   end Add_Protocol;

   procedure Add_String_Type (
         Name            :   in      gdbbas.Name_T;
         Length          :   in      gdbbas.User_String_Range_T;
          Class_Link       :   in      gdbcli.map ) is

      Udt_Record      :   gdbudt.User_Data_Type_Record_T(
                             gdbbas.Of_String );
      Exists_Already :   Boolean;
      Local_Name      : gdbbas.Name_T := name;
       Class_Info      : gdbcla.Class_Record_T :=
        gdbcli.Element(Container => Class_Link,
```

```
                        Key        => name);
    Udt_Node_Ptr    : gdbuti.map;
begin

      -- The default is string; set the string length
   To_Upper( Local_Name );
   Udt_Record.Name := Local_Name;
   Udt_Record.String_Length := Length;


   -- Add the node to the user data type table in the class table
    Exists_Already := gdbuti.Contains(Container => Class_Info.User_Data_Types,
                                      Key       => Local_Name);

   if Exists_Already then
      gdberr.Report_Error (gdberr.DUP_CLASS_TYPE_NAME, Name);
   else
     -- update the class_Info in the class table
      gdbuti.Insert(Container => Class_Info.User_Data_Types,
                    Key       => Local_Name,
                    New_Item  => Udt_Record);
   end if;

end Add_String_Type;


procedure Add_String_Type (
      Name            :   in      gdbbas.Name_T;
      Length          :   in      gdbbas.User_String_Range_T;
       Command_Link     :   in      Gdbcmi.map ) is

   Udt_Record     :   gdbudt.User_Data_Type_Record_T(
                        gdbbas.Of_String );
   Exists_Already :   Boolean;
   Local_Name     :   gdbbas.Name_T := Name;
   Command_Info    :   Gdbcom.Command_Record_T :=
      Gdbcmi.Element(Container => Command_Link,
                     Key       => Name);
    Udt_Node_Ptr    :   gdbuti.Map;
begin

      -- The default is string; set the string length
   To_Upper( Local_Name );
   Udt_Record.Name := Local_Name;
   Udt_Record.String_Length := Length;


      -- Add the node to the user data type table in the command table
    Exists_Already := gdbuti.Contains(Container => Command_Info.User_Data_Types,
                                      Key       => Local_Name);

   if Exists_Already then
      gdberr.Report_Error (gdberr.DUP_CMD_TYPE_NAME, Name);
   else
     -- update the command info in the command table
      gdbuti.Insert(Container => Command_Info.User_Data_Types,
                    Key       => Local_Name,
                    New_Item  => Udt_Record);
   end if;

end Add_String_Type;


procedure Add_Enumeration_Type (
      Name            :   in      gdbbas.Name_T;
      Class_Link       :   in      gdbcli.map;
       Type_Link        :   out     gdbuti.map ) is
```

```
   Udt_Record : gdbudt.User_Data_Type_Record_T(
                    gdbbas.Of_Enumeration );
   Class_Info : gdbcla.Class_Record_T :=
      gdbcli.Element(Container => Class_Link,
                     Key       => Name);
   Exists     : Boolean;
   Local_Name : gdbbas.Name_T := Name;

begin
   To_Upper ( Local_Name);

   Udt_Record.Name := Local_Name;
    Udt_Record.Enumeration_Pairs := gdbudt.gdbeni.Empty_Map;
   Udt_Record.Last_Value := -1;

      -- Add the node to the user data type table.

    Exists := gdbudt.gdbuti.Contains(Container => Class_Info.User_Data_Types,
                                     Key       => Local_Name);
    Type_Link := Class_Info.User_Data_Types;
   if Exists then
       gdberr.Report_Error (
         Package_Name       =>
              "Gdbbpt:Add_Enumeration_Type",
         Error_Description  =>
              "Duplicate type name on class: " & Name );
   else
     gdbudt.gdbuti.Insert(Container => Class_Info.User_Data_Types,
                          Key       => Local_Name,
                          New_Item  => Udt_Record);
   end if;

end Add_Enumeration_Type;


procedure Add_Enumeration_Type (
      Name           :   in      gdbbas.Name_T;
       Command_Link    :   in      Gdbcmi.map;
       Type_Link       :   out     gdbuti.map ) is

   Udt_Record   :  gdbudt.User_Data_Type_Record_T(
                      gdbbas.Of_Enumeration );
   Command_Info :  Gdbcom.Command_Record_T :=
      Gdbcmi.Element(Container => Command_Link,
                     Key       => Name);
   Exists       :  Boolean;
   Local_Name   :  gdbbas.Name_T := Name;

begin
   To_Upper ( Local_Name );

   Udt_Record.Name := Local_Name;
    Udt_Record.Enumeration_Pairs := gdbudt.gdbeni.Empty_Map;
   Udt_Record.Last_Value := -1;

      -- Add the node to the user data type table.
    Type_Link := Command_Info.User_Data_Types;
    Exists := gdbudt.gdbuti.Contains(Container => Command_Info.User_Data_Types,
                                     Key       => Local_Name);

   if Exists then
      gdberr.Report_Error (
         Package_Name       =>
              "Gdbbpt:Add_Enumeration_Type",
         Error_Description  =>
              "Duplicate type name on command: " & Name );
   else
           gdbudt.gdbuti.Insert(Container => Command_Info.User_Data_Types,
```

```
                          Key        => Local_Name,
                          New_Item   => Udt_Record);
   end if;

end Add_Enumeration_Type;


procedure Add_Enumeration_Pair (
      Enum_Identifier   :   in      gdbbas.Name_T;
      Internal_Code     :   in      gdbbas.User_Internal_Code_T;
      Type_Link         :   in      gdbuti.map ) is

   Enum_Record         :   gdbudt.Enumeration_Pair_Record_T;
   Enum_Table_Link     :   gdbudt.Gdbeni.map;
   Udt_Info            :   gdbudt.User_Data_Type_Record_T(
                             gdbbas.of_Enumeration );
   Exists              :   Boolean;
   Local_Name          :   gdbbas.Name_T := Enum_Identifier;

begin
   To_Upper ( Local_Name );

   Udt_Info := gdbuti.Element(Container => Type_Link,
                              Key       => Local_Name);

 -- should get a constraint error here if type_Link is
 -- not pointing to enumeration UDT


     begin
       Udt_Info.Last_Value := Udt_Info.Last_Value + 1;
     exception
       when Constraint_Error =>
         gdberr.Report_Error(
           Package_Name     =>   "Gdbbpt:Add_Enumeration_Pair",
           Error_Description =>  "Enumeration value out of range " &
                                    Enum_Identifier);
         return;
     end;
     enum_Record      := ( Name => Local_Name,
                       Internal_Code => Internal_Code,
                       Value => Udt_Info.Last_Value);

      -- Finally, add the node to the enumeration-pair table.

    Exists := gdbudt.gdbeni.Contains(Container => Udt_Info.Enumeration_Pairs,
                          Key       => Local_Name);

   if  Exists then
      gdberr.Report_Error (
         Package_Name    =>   "Gdbbpt:Add_Enumeration_Pair",
         Error_Description  =>
           "Duplicate enumeration value identifier  " &
               Enum_Identifier);
   else
      gdbudt.gdbeni.Insert(Container => Udt_Info.Enumeration_Pairs,
                           Key       => Local_Name,
                           New_Item  => Enum_Record);
   end if;
 exception
   when Constraint_Error =>
     gdberr.Report_Error (
       Package_Name =>
               "Gdbbpt: Add Enumeration Pair",
       Error_Description =>
               "UDT corrupt, unable to retrieve enumeration info");
end Add_Enumeration_Pair;


procedure Lookup_String_Type(Type_Name  : in  gdbbas.Name_T;
```

```
                              Str_Length : out gdbbas.User_String_Range_T;
                              Valid_Type : out Boolean) is
    Trailing_Spaces : String(1..71) := (others=>' ');
begin
    Str_Length := gdbbas.User_String_Range_T'first;
    Valid_Type := false;
    if (Type_Name(1..6) = "STRING") and
       (Type_Name(7) in '1'..'9') and
       (Type_Name(10..80) = Trailing_Spaces) then

        if ((Type_Name(9) = ' ') and
            ((Type_Name(8) = ' ') or (Type_Name(8) in '0'..'9'))) or
           ((Type_Name(9) in '0'..'9') and (Type_Name(8) in '0'..'9'))then
            declare
            begin
                Str_Length := gdbbas.User_String_Range_T'Value(Type_Name(7..9));
                Valid_Type := true;
            exception
                when others =>
                    null;
            end;
        end if;
    end if;
end Lookup_String_Type;

procedure Add_Attribute (
      Name               :   in      gdbbas.Name_T;
      AttType            :   in      gdbbas.Name_T;
      Array_Size         :   in      gdbbas.User_Array_Range_T;
      Write_Flag         :   in      gdbbas.Write_Flag_T;
      Machine_Type       :   in      gdbbas.Machine_Type_T;
      Class_Link         :   in      gdbcli.map;
      Byte_Offset_Valid  :   in      Boolean;
      Byte_Offset        :   in      Gdbbas.Attribute_Offset_t;
      Protocol_Info      :   in      Gdbbas.Protocol_Info_t) is

   Attribute_Record  :   Gdbatt.Attribute_Record_T;
   Attribute_Table   :   Gdbatt.Gdbati.map;
   Primitive_Type    :   gdbbas.Primitives := gdbbas.Undefined;
   Class_Info        : Gdbcla.Class_Record_T :=
      gdbcli.Element(Container => class_Link,
                     Key       => Name);
   Local_Name        : gdbbas.Name_T := Name;
   Local_Type_Name   : gdbbas.Name_T := AttType;
   Exists            : Boolean;
   String_Length     : gdbbas.User_String_Range_T := 1;

   --- Helper function to determine if data type
   --- Note this function has side effects on local variables external in Add_Attribute
   function  Validate_Data_Type(Data_Type_Name  : gdbbas.Name_T) return boolean is
       Udt_Info          : gdbudt.User_Data_Type_Record_T;
       Udt_Table           : gdbudt.gdbuti.cursor;
       Exists            : Boolean;
       Valid_Type        : Boolean;
       Resolved_Type_Name : gdbbas.Name_T;
   begin

       --- Look type up in primitives table
       Primitive_Type := Lookup_Primitive_Type ( Data_Type_Name );
       if Primitive_Type = gdbbas.Undefined  then

          -- It may be a user-defined type, look it up in the types table
         Udt_Info.Name := Data_Type_Name;
         Exists := gdbudt.gdbuti.Contains(Container => Class_Info.User_Data_Types,
                                          Key       => Data_Type_Name);

          Udt_Info := gdbudt.gdbuti.Element(Container => Class_Info.User_Data_Types,
                                            Key       => Data_Type_Name);
```

```
            --- If the type does not exist as a user type
            if not Exists then

                --- Check to see if this is a built-in string type
                Lookup_String_Type(Data_Type_name, String_Length, Valid_Type);

                --- If this is a valid string type, add to user defined types
                if Valid_Type then
                    Add_String_Type(Data_Type_Name, String_Length, Class_link);
                  Class_Info := gdbcli.Element(Container => class_Link,
                                               Key       => Data_Type_Name);
                Udt_Info.Name := Data_Type_Name;
                  Udt_Info         :=         gdbudt.gdbuti.Element(Container         =>
class_Info.User_Data_Types,
                                                 Key       => Data_Type_Name);
                else
                    --- Look up in data type definitions
                    Gdb_Definitions.Lookup_Data_Type(Data_Type_Name,         Resolved_Type_Name,
Valid_Type);
                    if Valid_Type then
                        return Validate_Data_Type(Resolved_Type_Name);
                    end if;

                end if;
            end if;

            --- If this resolves to a string type
            if Exists then
                String_Length := Udt_Info.String_Length;
                Attribute_Record.User_Data_Type := Udt_Table;
                -- note: keep Attribute primitive type undefined for user
                -- defined type get attribute info from Udt_Table
                return true;
            else
                return false;
            end if;

        elsif Primitive_Type = gdbbas.Of_Data_Object then
                return false;

        else
          -- If not user-defined, just record the primitive type.
          -- String and Enumeration are also entered here, since
          -- there was already an error message in
          -- Match_Primitive_Type.

            Attribute_Record.Primitive_Type := Primitive_Type;
            return true;

        end if ;

    end Validate_Data_Type;

  begin

    -- This procedure fills in information specific to the attribute
    -- in a record, and gets it linked in to the attribute table for
    -- that class.  It also updates the class table so that the next
    -- offset will start at the next byte after this attribute.

    To_Upper( Local_Name );
    To_Upper( Local_Type_Name );

    Attribute_Record.Name := Local_Name;

       -- Fill in the information specific to the attribute

       -- Process the data type, return if invalid
    if not Validate_Data_Type(Local_Type_Name) then
        gdberr.Report_Error(gdberr.BAD_ATTR_DATA_TYPE, AttType);
```

```
         return;
      end if;

         -- Process the array size.
         -- Lower Array size range has already been checked on the
         -- syntax side.
      if Primitive_Type = GDBBAS.Of_Boolean then
          if Array_Size > Max_Supported_Boolean_Array_Size  then
              gdberr.Report_Error                                    (gdberr.BAD_ATTR_ARRAY_SIZE,
gdbbas.User_Array_Range_T'image(array_size));
          end if;
      elsif (Primitive_Type = GDBBAS.Of_String or
             Primitive_Type = GDBBAS.Undefined) then
          if  (integer(Array_Size) * integer(String_Length)) > Max_Supported_Character_Array_Size
then
              gdberr.Report_Error                                    (gdberr.BAD_ATTR_ARRAY_SIZE,
gdbbas.User_Array_Range_T'image(array_size));
          end if;
      else -- Numeric Data Type
          if Array_Size > Max_Supported_Numeric_Array_Size then
              gdberr.Report_Error                                    (gdberr.BAD_ATTR_ARRAY_SIZE,
gdbbas.User_Array_Range_T'image(array_size));
          end if;
      end if;

      Attribute_Record.Array_Size := Array_Size;

         -- Process the Write Flag.  Range checked on syntax side.
      Attribute_Record.Write_Flag := Write_Flag;

         -- Process the Machine Type.  Range checked on syntax side.
      Attribute_Record.Machine_Type := Machine_Type;

         -- Fill in the offset & do bookkeeping in the class node.

      Compute_Offset ( Class_Record       =>  Class_Info,
                       Attribute_Record   =>  Attribute_Record,
                       Byte_Offset_Valid  =>  Byte_Offset_Valid,
                       Byte_Offset        =>  Byte_Offset);

         -- Compare end of attribute storage with maximum object size

      if Attribute_Record.Offset + Attribute_Record.Size >
         GDB_INITIAL_VALUES.Max_Object_Size then
          gdberr.Report_Error                                        (gdberr.BAD_ATTR_BYTE_OFFSET,
Gdbbas.Attribute_Offset_T'image(Attribute_Record.Offset));
      end if;

         -- Set the protocol info
      Attribute_Record.Protocol_Info := Protocol_Info;

         -- Finally, add the node to the attribute table.
       Exists := gdbatt.Gdbati.Contains(Container => Class_Info.Attributes,
                                        Key       => Attribute_Record.Name);

      if Exists then
         gdberr.Report_Error (gdberr.DUP_ATTR_NAME, Name);
      else
              gdbatt.Gdbati.Insert(Container => Class_Info.Attributes,
                                   Key       => Attribute_Record.Name,
                                   New_Item  => Attribute_Record);
      end if;

   end Add_Attribute;



   procedure Add_Instance (
         Instance_Name       :   in      gdbbas.Name_T;
         Alternate_Name      :   in      gdbbas.Name_T;
```

```
      Class_Name          :   in      gdbbas.Name_T;
      Location_Id         :   in      gdbbas.Instance_Id_T;
      Bit_Offset          :   in      gdbbas.Bit_Offset_T;
      Instance_Ctr        :   in      gdbbas.Long_Natural_T;
      Protocol_Info       :   in      Gdbbas.Protocol_Info_T;
      Offset              :   out     gdbbas.Attribute_Offset_T ) is

   Class_Info         : Gdbcla.Class_Record_T;
   Class_Link           : gdbcla.gdbcli.cursor;
   Att_Info           : Gdbatt.Attribute_Record_T;
   null_att_name      : gdbbas.Name_T := (others => ' ');
   Instance_Record    : Gdbins.Instance_Record_T;
   Instance_Table       : Gdbins.Gdbini.map;
   Exists             : Boolean;
   Is_Unique          : boolean;
   Local_Name         : gdbbas.Name_T := Instance_Name;
   Local_Alt_Name     : gdbbas.Name_T := Alternate_Name;
   Local_Id           : gdbbas.Instance_Id_T := Location_Id;
    cursor : gdbins.Gdbini.Cursor := gdbins.Gdbini.No_Element;

begin
   To_Upper ( Local_Name );
   To_Upper ( Local_Alt_Name );
   Instance_Record.Name     := Local_Name;
   Instance_Record.Alt_Name := Local_Alt_Name;

      -- Process the instance id
      -- It must be non-null.
   if  Local_Id < 1 then
      if Local_Id /= -1 then
         gdberr.Report_Error(gdberr.BAD_INST_ID, gdbbas.Instance_Id_T'image(local_id));
         -- Reset the id, so later processing won't blow up
      end if;
      Local_Id := 1;
      Is_Unique := true; -- force Class_Name processing
   else
      Gdbivi.Check_Id (
         Id         =>   Local_Id,
         Unique     =>   Is_Unique );
      if not is_unique then
         gdberr.Report_Error(gdberr.DUP_INST_ID, gdbbas.Instance_Id_T'image(local_id));
       end if;
   end if;

   Instance_Record.Location_Id := Local_Id;
   Instance_Record.Protocol_Info := Protocol_Info;

      -- Process the class name
      -- It must match one previously processed in this build.
   Local_Name := Class_Name;
   To_Upper ( Local_Name);
   Class_Info.Name := Local_Name;
   Exists:= gdbcla.gdbcli.contains(Container => Gdbses.Global_Name_Tables.Class,
                                   Key       => Local_Name);

   if Exists then
      Instance_Record.Of_Class := Class_Link;
      Class_Info           :=          gdbcla.gdbcli.element(Container          =>
Gdbses.Global_Name_Tables.Class,
                                   Key       => Local_Name);
      -- return the next offset for adjusting the location_id in case it is
      -- a '+'.
      Offset := Class_Info.Last_Offset;
   else
       gdberr.Report_Error(gdberr.UNDEFINED_CLASS, Instance_Name);
   end if;

      -- Process the Bit Offset.  Prolly a good time to check and see if
      -- it is the right type of instance to have a nonzero bit offset.
   begin
```

```
        Att_info := gdbatt.gdbati.Element(Container => Class_Info.Attributes,
                                          Key       => Local_Name);
      if (Att_Info.Name = null_att_name) --this should only be single types
          AND (Att_Info.Primitive_Type = gdbbas.Of_Signed_Bit or
               Att_Info.Primitive_Type = gdbbas.Of_Unsigned_Bit) then
        Instance_Record.Bit_Offset := Bit_Offset;
      else --it is not a bit type so bit offset should be 0
        if Bit_Offset /= 0 then
          gdberr.Report_Warning (
            Package_Name      =>   "Gdbbpt:Add_Instance",
            Warning_Description => "Illegal Bit Offset",
            Warning_Elaboration =>
                "Nonzero Bit Offsets only allowed for single types Bit." &
                "  Assigning bit offset 0: " &  Instance_Name);
        end if;
        --set the Bit Offset to 0 and continue parsing
        Instance_Record.Bit_Offset := 0;
      end if;
    -- this needed in case no attributes
    exception when others =>
      Instance_Record.Bit_Offset := 0;
    end;

       -- Process the Instance Ctr.  Checked on syntax side.
    Instance_Record.Instance_Ctr := gdbbas.long_natural_T(Local_Id);

       -- Process initial value, if provided
    GDB_INITIAL_VALUES.Add_Instance(Instance_Record, Att_Info);

       -- Finally, add the node to the instance table.

    gdbins.Gdbini.Insert(Container => Gdbses.Global_Name_Tables.Instance,
                         Key       => Instance_Record.Name,
                         New_Item  => Instance_Record,
                         Position  => cursor,
                         Inserted  => Exists);

    if Exists then
       gdberr.Report_Error(gdberr.DUP_INST_NAME, Instance_Name);
    end if;


  end Add_Instance;


  procedure Add_Command (
       Name               :   in      gdbbas.Name_T;
       Machine_Type       :   in      gdbbas.Machine_Type_T;
       Command_Shell      :   in      gdbbas.Command_Shell_T;
        Command_Link      :   out     Gdbcmi.map)
  is

     Command_Record   :   Gdbcom.Command_Record_T;
     Command_Table    :   Gdbcmi.map;
     Exists           :   Boolean;
     Uppercase_Name   :   gdbbas.Name_T := Name;
cursor : Gdbcmi.Cursor := Gdbcmi.No_Element;
  begin
       -- Commands combine aspects of classes and instances
       -- and the parameter situation is different.
       -- This procedure just processes name, machine type, and the command
       -- shell.  The rest is filled in when processing parameters.
     To_Upper ( Uppercase_Name );
     Command_Record.Name:= Uppercase_Name;
     Command_Record.User_Data_Types := gdbudt.gdbuti.Empty_Map;
      Command_Record.Parameters  :=  gdbprm.Gdbpri.Empty_Map;
     Command_Record.Last_Offset := 0;
```

```
         -- Process the machine type.

         -- Machine type already checked on syntax side.

      Command_Record.Machine_Type := Machine_Type;

         -- Process the command shell.

         -- Command Shell already checked on syntax side.

      Command_Record.Command_Shell := Command_Shell;

         -- Finally, add the node to the command table.

      Gdbcmi.Insert(Gdbses.Global_Name_Tables.Command,              Command_Record.Name,
Command_Record, cursor, Exists);
Command_Link := Gdbses.Global_Name_Tables.Command;
      if Exists then
         gdberr.Report_Error (gdberr.DUP_CMD_NAME, Name);
      end if;
   end Add_Command;


   procedure Add_Parameter (
         Name               :   in      gdbbas.Name_T;
         PrmType            :   in      gdbbas.Name_T;
         Array_Size         :   in      gdbbas.User_Array_Range_T;
         Byte_Offset        :   in      gdbbas.Byte_Offset_T;
         Bit_Offset         :   in      gdbbas.Bit_Offset_T;
          Command_Link        :   in      Gdbcmi.map;
         Next_Offset        :   out     Primitive_Data_Types.Integer_T) is
      use Primitive_Data_Types;

      Parameter_Record :   Gdbprm.Parameter_Record_T ;
      Parameter_Table    :   Gdbprm.Gdbpri.map;
      Primitive_Type   :   gdbbas.Primitives
                              := gdbbas.Primitives'(gdbbas.Undefined );
      Exists           :   Boolean;
      Command_Info       :   Gdbcom.Command_Record_T :=
         gdbcmi.Element(Container => command_Link,
                        Key       => Name);
      Local_Name       :   gdbbas.Name_T := Name;
      Local_Type_Name  :   gdbbas.Name_T := PrmType;
      String_Length      : gdbbas.User_String_Range_T := 1;
       cursor : gdbprm.gdbpri.Cursor;

      --- Helper function to determine if data type is valid
      --- Note this function has side effects on local variables external in Add_Parameter
      function  Validate_Data_Type(Data_Type_Name  : gdbbas.Name_T) return boolean is
          Udt_Info           : gdbudt.User_Data_Type_Record_T;
          Udt_Table             : gdbudt.gdbuti.cursor;
          Exists             : Boolean;
          Valid_Type         : Boolean;
          Resolved_Type_Name : gdbbas.Name_T;
      begin

          --- Look type up in primitives table
          Primitive_Type := Lookup_Primitive_Type ( Data_Type_Name );
          if Primitive_Type = gdbbas.Undefined  then

             -- It may be a user-defined type, look it up in the types table
            Udt_Info.Name := Data_Type_Name;
            exists := gdbudt.gdbuti.contains(Container => Command_Info.User_Data_Types,
                                             Key       => Data_Type_Name);
              Udt_Info           :=           gdbudt.gdbuti.element(Container           =>
Command_Info.User_Data_Types,
                                             Key       => Data_Type_Name);

             --- If the type does not exist as a user type
```

```
            if not Exists then

                --- Check to see if this is a built-in string type
                Lookup_String_Type(Data_Type_name, String_Length, Valid_Type);

                --- If this is a valid string type, add to user defined types
                if Valid_Type then
                    Add_String_Type(Data_Type_Name, String_Length, Command_link);
                Command_Info := gdbcmi.Element(Container => Command_Link,
                                               Key       => Data_Type_Name);
                Udt_Info.Name := Data_Type_Name;
                Udt_Info          :=          gdbudt.gdbuti.element(Container          =>
command_Info.User_Data_Types,
                                                                   Key       => Data_Type_Name);
                else
                    --- Look up in data type definitions
                    Gdb_Definitions.Lookup_Data_Type(Data_Type_Name,        Resolved_Type_Name,
Valid_Type);
                    if Valid_Type then
                        return Validate_Data_Type(Resolved_Type_Name);
                    end if;

                end if;

            end if;

            --- If this resolves to a string type
            if Exists then
                String_Length := Udt_Info.String_Length;
                Parameter_Record.User_Data_Type := Udt_Table;
                -- note: keep parameter primitive type undefined for user
                --       defined type to get parameter info from Udt_Table
                return true;
            else
                gdberr.Report_Error(gdberr.BAD_PARAM_DATA_TYPE, PrmType);
                return false;
            end if;

        else
          -- If not user-defined, just record the primitive type.
          -- String and Enumeration are also entered here, since
          -- there was already an error message in
          -- Match_Primitive_Type.

            Parameter_Record.Primitive_Type := Primitive_Type;
            return true;

        end if ;

    end Validate_Data_Type;

  begin

    -- This procedure fills in information specific to the parameter
    -- in a record, and gets it linked in to the parameter table for
    -- that command.  It also updates the command table so that the next
    -- offset will start at the next byte after this parameter.

    To_Upper( Local_Name );
    To_Upper( Local_Type_Name );
    Next_Offset := 0;

    Parameter_Record.Name := Local_Name;

       -- Fill in the information specific to the attribute

       -- Process the type
       -- Process the data type, return if invalid
    if not Validate_Data_Type(Local_Type_Name) then
        return;
```

```
      end if;

         -- Process the array size.

         -- Array size lower bound range has already been checked on the
         -- syntax side.
      if Primitive_Type = GDBBAS.Of_Boolean then
          if Array_Size > Max_Supported_Boolean_Array_Size  then
              gdberr.Report_Error                                    (gdberr.BAD_PARAM_ARRAY_SIZE,
gdbbas.User_Array_Range_T'image(array_size));
          end if;
      elsif (Primitive_Type = GDBBAS.Of_String or
             Primitive_Type = GDBBAS.Undefined) then
          if  (integer(Array_Size) * integer(String_Length)) > Max_Supported_Character_Array_Size
then
              gdberr.Report_Error                                    (gdberr.BAD_PARAM_ARRAY_SIZE,
gdbbas.User_Array_Range_T'image(array_size));
          end if;
      elsif (Primitive_Type = GDBBAS.Of_Data_Object) then
          if Array_Size /= 1 then
              gdberr.Report_Error                                    (gdberr.BAD_PARAM_ARRAY_SIZE,
gdbbas.User_Array_Range_T'image(array_size));
          end if;
      else -- Numeric Data Type
          if Array_Size > Max_Supported_Numeric_Array_Size then
              gdberr.Report_Error                                    (gdberr.BAD_PARAM_ARRAY_SIZE,
gdbbas.User_Array_Range_T'image(array_size));
          end if;
      end if;

      Parameter_Record.Array_Size := Array_Size;

         -- Process the byte offset.

         -- Byte offset range has already been checked on the
         -- syntax side.

      Parameter_Record.Byte_Offset := Byte_Offset;

         -- Process the Bit Offset.  Check and see if
         -- it is the right type to have a nonzero bit offset.

      if Primitive_Type = gdbbas.Primitives'( gdbbas.Of_Signed_Bit ) or
         Primitive_Type = gdbbas.Primitives'( gdbbas.Of_Unsigned_Bit )then

        -- Fill in the bit offset.
        Parameter_Record.Bit_Offset := Bit_Offset;

      else --it is not a bit type so bit offset should be 0
        if Bit_Offset /= 0 then
          gdberr.Report_Warning (
                Package_Name      =>   "Gdbbpt:Add_Parameter",
                Warning_Description => "Illegal Bit Offset",
                Warning_Elaboration =>
                    "Nonzero Bit Offsets only allowed for type Bit." &
                    "  Assigning bit offset 0: " &  Local_Name);
          --set the Bit Offset to 0 and continue parsing
          Parameter_Record.Bit_Offset := 0;
        end if;
      end if;

         --Compute Offset for table storage.  This offset and size are not
         --needed for output.
       Compute_Offset ( Command_Record =>   Command_Info,
                        Parameter_Record   =>   Parameter_Record );

         -- Compare end of parameter storage with maximum byte shell size

      if gdbbas.Parameter_Offset_T(Parameter_Record.Byte_Offset) + Parameter_Record.Size >
         (gdbbas.Parameter_Offset_T(Param_Value_Location_Type'last) + 1) then
          gdberr.Report_Error (gdberr.BAD_PARAM_OFFSET_SIZE, name);
```

```
      end if;

         -- Check for too many data object command parameters
      if Primitive_Type = gdbbas.Primitives'( gdbbas.Of_Data_Object ) then
          if   Command_Info.Num_Data_Object_Params   >=   gdbbas.Max_Data_Object_Params_Per_Command
then
              gdberr.Report_Error (gdberr.MAX_DATA_OBJECT_PARAMS, Command_Info.name);
          else
              Command_Info.Num_Data_Object_Params := Command_Info.Num_Data_Object_Params + 1;
          end if;
      end if;

         -- Finally, add the node to the parameter table.

      gdbprm.Gdbpri.Insert(Command_Info.Parameters,                Parameter_Record.Name,
Parameter_Record, cursor, Exists);

      if Exists then
           gdberr.Report_Error(gdberr.DUP_PARAM_NAME, Name);
      end if;

      Next_Offset := Primitive_Data_Types.Integer_T(Parameter_Record.Byte_Offset) +
                     Primitive_Data_Types.Integer_T(Parameter_Record.Size);

  end Add_Parameter;

  procedure Initialize is

          class_name      : gdbbas.Name_T;
           the_class_link  : gdbcli.map;
          attribute_type  : gdbbas.Name_T;
          til_name        : gdbbas.Name_T;
  begin
        --- Create class definition
        class_name(1..18) := "$BUFFER_ID_GENERAL";
        class_name(19..80):= (others => ' ');
        Add_Class( Name       => class_name,
                   Class_Link => the_class_link);
        til_name(1..7) := "GENERAL";
        til_name(8..80) := (others => ' ');
        Add_Protocol( Name       => til_name,
                  Class_Link => the_class_link);
        attribute_type := (others => ' ');
        attribute_type(1..12) := "LONG_NATURAL";
        Add_Attribute( Name              => (others => ' '),
                       AttType           => attribute_type,
                       Array_Size        => 1,
                       Write_Flag        => READ_ONLY,
                       Machine_Type      => SUN,
                       Class_Link        => the_class_link,
                       Byte_Offset_Valid => true,
                       Byte_Offset       => 0,
                       Protocol_Info     => (others => ' '));
  end Initialize;

end Gdbbpt_Basic_Parser_To_Transform_Interface;
```

Gdbalt_parser.y

```
------------------------------------------------------------------------------
--
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--
------------------------------------------------------------------------------
--
```

```
--    File: gdb_Parser.Y
--
--    Purpose: Ayacc description for the gdb input processor
--             grammar definition.  Includes the Ada code to
--             convert the input to proper internal representations
--             and calls to the proper "add" functions.
--
--    Author: rdb2207 (Robert Beal) (modified significantly by
--                                   Jim Napier (JMN2489))
--
--    Creation Date: 2/24/94
--
--    Change History:
--
--         2/28/94 - rdb2207 - Added Producer and Tier1 Rules
--         6/8/95  - JMN2489 - Got rid of Producer and Tier1 rules and
--                             default values.  Added some extra fields.
--         3/05/97 - fak6330 - Add PUI name to access interface; PCR 469
--         5/05/97 - fak6330 - Remove extra comma when no PUI in instance; PCR 469
--         7/08/97 - dmd2273 - Expanded "location" in "an_instance" rule
--                             to eliminate shift/reduce conflict; PCR 469
--         7/28/97 - DJP2489 - Added shell params section for commands.
--         2/11/98 - JSM1269 - added changes needed for TL for Solaris use.
--        11/19/98 - DMD2273 - PCR 064: Make bit location in command parameters optional
--                           - Make shell bytes optional, allow replacement by ZEROS
--                           - Change name of shell_params to shell_overlays
--                           - Move shell_overlay section to after byte shell
--                           - Change format of specification of shell_overlays to:
--                             Name, type, byteoffset, value;
--                           - Eliminate bit offset for instances
--        11/24/98 - DMD2273 - PCR 057: Initialize command shell bytes to zero at start
--                             of processing command
--        12/21/98 - DMD2273 - PCR 070: Eliminate calls to generate instance summary file
--        05/30/99 - D.DiBiaso - PCR X071 : Extend range of long_natural type shell overlays
--                                          to 0..2^32-1
--        06/01/99 - D.DiBiaso - PCR X017 : Allow "BIG_ENDIAN" and "LITTLE_ENDIAN" as
--                                          possible machine types, in place of "SUN" and
--                                          "INTEL", respectively
--        06/01/99 - D.DiBiaso - PCR X008 : Add parsing and processing to support string
--                                          type values for command shell overlays
--        06/03/99 - D.DiBiaso - PCR X019 : Add optional byte offset field to attribute
--                                          definitions
--        06/09/99 - D.DiBiaso - PCR X067 : Clean up error reporting messages
--        06/14/99 - D.DiBiaso - PCR X025 : Change instance counter from short to long natural
--        06/26/99 - D.DiBiaso - PCR X003 : Modify parsing to handle initial values for
--                                          starred object instances
--        07/05/99 - D.DiBiaso - PCR X098 : Add optional machine type to shell overlay definition
--        01/17/00 - D.DiBiaso - PCR X203 : Add TIL Name to class definitions and
--                                          TIL info to attribute and instance definitions
--        02/04/00 - D.DiBiaso - PCR X202 : Extend command parameters to byte offset limit 320
--        03/15/00 - D.DiBiaso - PCR X205 : Increase command shell size from 320 to 1024
--             04/24/00 - D.DiBiaso - PCR X308 : Pass string value into set_float_value,
set_int_value,
--                                          and set_boolean_value procedures
--         05/02/00 - D.DiBiaso - PCR X059 : Add shell overlay support for Boolean, Float, and
Long
--                                          Float data types
--        05/02/00 - D.DiBiaso - PCR X229 : Add support for single quote delimitted strings; Add
--                                          support for STRING# data types
--        05/03/00 - D.DiBiaso - PCR X250 : Fix grammar so that classes must have atleast one
--                                          attribute defined; Cleanup grammar BNF
--           05/04/00 - D.DiBiaso - PCR X200 : Add "DEFINITIONS" section with subsection for
defining
--                                          name associations for TIL names, system command Ids,
--                                          data types, and shell locations.
--        05/05/00 - D.DiBiaso - PCR X235 : Added "CONSTANT_VALUES" definition subsection for
--                                          defining nam associations for shell overlay, instance
--                                            initial and til info values. Added routing protocol
tag
--                                          for command definitions.
--             05/17/00 - D.DiBiaso - PCR X314 : Use Max Object Size from package spec
```

```
GDB_INITIAL_VALUES
--        10/02/00 - D.DiBiaso - PCR X362 : Added REMOTE_TIS tag parsing
--        01/06/03 - D.DiBiaso - PCR X516 : Added support for buffer IDs
--         01/15/03 - D.DiBiaso - PCR X517 : Added support for Data_Object data type for command
parameters
--        03/05/03 - D.DiBiaso - PCR X531 : Increased max TIL Info value size from 256 to 512
--        11/16/04 - D.DiBiaso - PCR X627 : Added special identifier for class names
--                                          Start number buffer instance IDs at 2_010_000_000
--              05/06/09 - I.Charny  - TBD        :  Changed  Build  Function  Interface
Variables from binary trees to hash tables
-------------------------------------------------------------------------



-------------------------------------------------------------------------
-- Declarations Section -- This section becomes the gdb_parser_tokens.ada
--                         file after running this file through AYacc.
-------------------------------------------------------------------------

%token   IDENTIFIER_T,   INTEGER_T,   SIGNED_INTEGER_T,   HEX_INTEGER_T,   BIN_INTEGER_T,   FLOAT_T,
STRING_VALUE_T
%token ZEROS_T, BASED_ON_T, AFTER_T SPECIAL_IDENTIFIER_T

%token CLASS_TAG, COMMAND_TAG, TYPES_TAG, ATTRIBUTES_TAG, TIL_TAG
%token PARAMETERS_TAG, INSTANCES_TAG, MACHINE_TAG, SHELL_TAG, SHELL_OVERLAYS_TAG
%token DEFINITIONS_TAG, TIL_NAMES_TAG, COMMAND_IDS_TAG, SHELL_LOCATIONS_TAG, DATA_TYPES_TAG
%token CONSTANT_VALUES_TAG, UDP_PROTOCOL_TAG, REMOTE_TIS_TAG, BUFFER_IDS_TAG

%token PLUS_T, ASTERISK_T, ARROW_T, RIGHT_SQUARE_BRACKET_T, TIL_INFO_T, LEFT_BRACE_T
%token COMMA_T LEFT_PARENTHESIS_T, RIGHT_PARENTHESIS_T, SEMICOLON_T

%token BT1_8_T, BT9_16_T, BT17_24_T, BT25_32_T, BT33_40_T, BT41_48_T
%token BT49_56_T, BT57_64_T, BT65_72_T, BT73_80_T, BT81_88_T
%token BT89_96_T, BT97_104_T, BT105_112_T, BT113_120_T, BT121_128_T
%token BT129_136_T, BT137_144_T, BT145_152_T, BT153_160_T, BT161_168_T
%token BT169_176_T, BT177_184_T, BT185_192_T, BT193_200_T, BT201_208_T
%token BT209_216_T, BT217_224_T, BT225_232_T, BT233_240_T, BT241_248_T
%token BT249_256_T, BT257_264_T, BT265_272_T, BT273_280_T, BT281_288_T
%token BT289_296_T, BT297_304_T, BT305_312_T, BT313_320_T, BT321_328_T
%token BT329_336_T, BT337_344_T, BT345_352_T, BT353_360_T, BT361_368_T
%token BT369_376_T, BT377_384_T, BT385_392_T, BT393_400_T, BT401_408_T
%token BT409_416_T, BT417_424_T, BT425_432_T, BT433_440_T, BT441_448_T
%token BT449_456_T, BT457_464_T, BT465_472_T, BT473_480_T, BT481_488_T
%token BT489_496_T, BT497_504_T, BT505_512_T, BT513_520_T, BT521_528_T
%token BT529_536_T, BT537_544_T, BT545_552_T, BT553_560_T, BT561_568_T
%token BT569_576_T, BT577_584_T, BT585_592_T, BT593_600_T, BT601_608_T
%token BT609_616_T, BT617_624_T, BT625_632_T, BT633_640_T, BT641_648_T
%token BT649_656_T, BT657_664_T, BT665_672_T, BT673_680_T, BT681_688_T
%token BT689_696_T, BT697_704_T, BT705_712_T, BT713_720_T, BT721_728_T
%token BT729_736_T, BT737_744_T, BT745_752_T, BT753_760_T, BT761_768_T
%token BT769_776_T, BT777_784_T, BT785_792_T, BT793_800_T, BT801_808_T
%token BT809_816_T, BT817_824_T, BT825_832_T, BT833_840_T, BT841_848_T
%token BT849_856_T, BT857_864_T, BT865_872_T, BT873_880_T, BT881_888_T
%token BT889_896_T, BT897_904_T, BT905_912_T, BT913_920_T, BT921_928_T
%token BT929_936_T, BT937_944_T, BT945_952_T, BT953_960_T, BT961_968_T
%token BT969_976_T, BT977_984_T, BT985_992_T, BT993_1000_T, BT1001_1008_T
%token BT1009_1016_T, BT1017_1024_T

{

   ----------------------------------------------------------------------
   --
   -- File: gdb_parser_tokens.ada
   --
   -- Author: rdb2207 (with help from AYacc)
   --
   --
   -- Notes:
   --
   -- Buffer_Size defines the size of the buffer used to pass
   -- information from the Lexer to the Parser (Class Names,
   -- Command Names, Instance Names, etc.)
```

```
    --
    -- Identifier_size is used to mantain the maximum allowed
    -- size for an identifier.
    --
    -- String_Buffer_Size is used to maintain the maximum allowed
    -- size for a quoted string literal.
    --
    ------------------------------------------------------------------------
    buffer_size        : CONSTANT Integer := 514;
    string_buffer_size : CONSTANT Integer := 258;
    identifier_size    : CONSTANT Integer := 80;

    subtype YYSType is String(1..buffer_size);
}

%%

-------------------------------------
-- Rules Section
-------------------------------------


-------------------------------------------------
-- The Root Rule, the one that starts the parsing
-------------------------------------------------
start :
section_definition |
start section_definition;

section_definition :
defines_definition |
class_definition   |
command_definition |
instance_definition;

--------------------------------
--- Defines Definition Rules ---
--------------------------------
defines_definition :
DEFINITIONS_TAG defines;

defines :
definition |
defines definition;

definition :
define_til_names |
define_command_ids |
define_shell_locations |
define_data_types |
define_constant_values |
define_buffer_ids;

---------------------------------
--- TIL Name  Definition Rules ---
---------------------------------

define_til_names :
TIL_NAMES_TAG til_names;

til_names :
til_name_definition |
til_names til_name_definition;

til_name_definition :
IDENTIFIER_T SEMICOLON_T {
        GDB_PROTOCOL.Add_Protocol($1(1..identifier_size));

        --- Create class definition
        class_name(1..11) := "$BUFFER_ID_";
        class_name(12..80):= $1(1..69);
```

```
        Add_Class( Name       => class_name,
                   Class_Link => the_class_link);
        Add_Protocol( Name       => $1(1..identifier_size),
                  Class_Link => the_class_link);
        attribute_type := (others => ' ');
        attribute_type(1..12) := "LONG_NATURAL";
        Add_Attribute( Name             => (others => ' '),
                       AttType          => attribute_type,
                       Array_Size       => 1,
                       Write_Flag       => READ_ONLY,
                       Machine_Type     => SUN,
                       Class_Link       => the_class_link,
                       Byte_Offset_Valid => true,
                       Byte_Offset      => 0,
                       Protocol_Info    => (others => ' '));
};

-----------------------------------
--- Command ID Definition Rules ---
-----------------------------------

define_command_ids :
COMMAND_IDS_TAG command_ids;

command_ids :
command_id_definition |
command_ids command_id_definition;

command_id_definition :
IDENTIFIER_T COMMA_T INTEGER_T SEMICOLON_T {
    declare
        Exists : Boolean;
        Id     : TYPES.Long_Integer_T;
    begin
        Id := Types.long_integer_t'Value($3);
        if Id > 0 then
            GDB_DEFINITIONS.Define_Command_Id($1(1..Identifier_Size), Id, Exists);
            if Exists then
                Report_Error(DUP_COMMAND_ID, $1(1..identifier_size));
            end if;
        end if;
    exception
        when others =>
            Report_Error(BAD_COMMAND_ID_DEF, $3(1..identifier_size));
    end;

};

---------------------------------------
--- Shell Location Definition Rules ---
---------------------------------------

define_shell_locations :
SHELL_LOCATIONS_TAG shell_locations;

shell_locations :
shell_location_definition |
shell_locations shell_location_definition;

shell_location_definition :
IDENTIFIER_T COMMA_T INTEGER_T SEMICOLON_T {
     declare
        Exists : Boolean;
        Offset : Byte_Offset_T;
    begin
        Offset := Byte_Offset_t'Value($3);
        GDB_DEFINITIONS.Define_Shell_Offset($1(1..Identifier_Size),      TYPES.Integer_T(Offset),
Exists);
        if Exists then
            Report_Error(DUP_SHELL_LOCATION, $1(1..identifier_size));
        end if;
```

```
    exception
        when others =>
            Report_Error(BAD_SHELL_LOCATION_DEF, $3(1..identifier_size));
    end;
};

---------------------------------
--- Data Type Definition Rules ---
---------------------------------
define_data_types :
DATA_TYPES_TAG data_types;

data_types :
data_type_definition |
data_types data_type_definition;

data_type_definition :
IDENTIFIER_T COMMA_T IDENTIFIER_T SEMICOLON_T {
    declare
        Type_Value  : Name_T := $3(1..Identifier_Size);
        Type_Name   : Name_T := $1(1..Identifier_Size);
        Data_Type   : Primitives;
        Str_Length : User_String_Range_T;
        Valid_Type : Boolean;
        Valid_Name : Boolean;
        Exists      : Boolean;
    begin
        --- Check to see if data type value is valid, check base primitive types,
        --- then extended string data types
        CASE_CONVERSION.To_Upper(Type_Value);
        Data_Type :=  Lookup_Primitive_Type (Type_Value);
        if (Data_Type /= Of_Byte) and
           (Data_Type /= Of_Short_Integer) and
           (Data_Type /= Of_Natural) and
           (Data_Type /= Of_Integer) and
           (Data_Type /= Of_Long_Natural) and
           (Data_Type /= Of_Long_Integer) and
           (Data_Type /= Of_Float) and
           (Data_Type /= Of_Long_Float) and
           (Data_Type /= Of_Boolean) and
           (Data_Type /= Of_Data_Object) then
            Lookup_String_Type(Type_Value, Str_Length, Valid_Type);
        else
            Valid_Type := true;
        end if;

        --- Check to see if data type name is valid, check base primitive types,
        --- then extended string data types
        CASE_CONVERSION.To_Upper(Type_Name);
        Data_Type :=  Lookup_Primitive_Type (Type_Name, true);
        if (Data_Type = Of_Byte) or
           (Data_Type = Of_Short_Integer) or
           (Data_Type = Of_Natural) or
           (Data_Type = Of_Integer) or
           (Data_Type = Of_Long_Natural) or
           (Data_Type = Of_Long_Integer) or
           (Data_Type = Of_Float) or
           (Data_Type = Of_Long_Float) or
           (Data_Type = Of_Boolean) or
           (Data_Type = Of_String) or
           (Data_Type = Of_Data_Object) then
            Valid_Name := false;
        else
            Lookup_String_Type(Type_Name, Str_Length, Exists);
            Valid_Name := not Exists;
        end if;

        --- Error if invalid name or value
        if not Valid_Type then
            Report_Error(BAD_DATA_TYPE_DEF, $3(1..Identifier_Size));
        end if;
```

```
        if not Valid_Name then
            Report_Error(BAD_DATA_TYPE_NAME, $1(1..Identifier_Size));
        end if;

        --- If data type name is valid, store in table, check for duplicates
        if Valid_Type and Valid_Name then
            GDB_DEFINITIONS.Define_Data_Type($1(1..Identifier_Size), Type_Value, Exists);
            if Exists then
                Report_Error(DUP_DATA_TYPE, $1(1..Identifier_Size));
            end if;
        end if;
    end;
};


-----------------------------------
--- Buffer IDs Definition Rules ---
-----------------------------------

define_buffer_ids :
BUFFER_IDS_TAG buffer_ids;

buffer_ids :
buffer_id_definition |
buffer_ids buffer_id_definition;

buffer_id_definition :
IDENTIFIER_T COMMA_T IDENTIFIER_T opt_bufid_info {
    declare
        Exists : Boolean := true;
        Val : String(1..6) := $1(1..6);
        Buffer_Id : Long_Integer_T;
        Found : Boolean := False;
        len : integer;
    begin
        --- Create constant value from next buffer id
        GDB_INITIAL_VALUES.Clear_Last_Value;
        Buffer_ID := Next_Buffer_ID;
        if Next_Buffer_ID < 999_999_999 then
            Next_Buffer_ID := Next_Buffer_ID + 1;
        end if;
        GDB_INITIAL_VALUES.Set_Int_Value(Primitive_Data_Types.Long_Integer_t(Buffer_ID),
                                         Trim(Long_Integer_t'image(Buffer_ID)));

        --- Check for bad name
        case_conversion.to_upper(Val);
        if Val = "TRUE  " or Val = "FALSE " then
            Report_Error(BAD_CONSTANT_VALUE_NAME, $1(1..identifier_size));
        else
            GDB_DEFINITIONS.Define_Constant_Value($1(1..Identifier_Size),
                                                  GDB_INITIAL_VALUES.Current, Exists);
            if Exists then
                Report_Error(DUP_CONSTANT_VALUE, $1(1..identifier_size));
            end if;
        end if;

        if not Exists then

            --- Check protocol name
            GDB_PROTOCOL.Lookup_Protocol($3(1..identifier_size), Found);

            if Found then

                --- Create class name
                class_name(1..11) := "$BUFFER_ID_";
                class_name(12..80) := $3(1..69);

                --- Create instance
                if buffer_location_ctr < long_integer_t'last then
                    buffer_location_ctr := buffer_location_ctr + 1;
                else
```

```
                    report_error(BAD_INST_ID, "+");
                    buffer_location_ctr := -1;
                end if;
                instance_id := Instance_Id_T(buffer_location_ctr);

                instance_name := $1(1..identifier_size);
                if instance_name(1) in '0'..'9' then
                    report_error(PARSING_ERROR, instance_name);
                end if;
                pui_name      := $1(1..identifier_size);
                bit_offset    := 0;

                --increment the instance counter to keep track of order.
                instance_ctr := instance_ctr + 1;

                --- Add $BUFFER_ID TIL Info Tag
                len := Long_Integer_t'image(Buffer_ID)'length;
                if inst_til_last <= inst_til_info'length - (len+12) then
                    inst_til_info(inst_til_last..(inst_til_last+12+len))                    :=
"{$BUFFER_ID:"&Long_Integer_t'image(Buffer_ID)&"}";
                    inst_til_last := inst_til_last + 13 + len;
                else
                    report_error(TIL_INFO_TOO_LONG);
                end if;

                Add_Instance ( Instance_Name  => instance_name,
                               Alternate_Name => pui_name,
                               Class_Name     => class_name,
                               Location_Id    => instance_id,
                               Bit_Offset     => bit_offset,
                               Instance_Ctr   => instance_ctr,
                               Protocol_Info  => inst_til_info,
                               Offset         => next_instance_loc_offset);
            end if;
        end if;

        inst_til_last := 1;
        inst_til_info := (others => ' ');
    end;
};

opt_bufid_info :
SEMICOLON_T {
    inst_til_info := (others => ' ');
    inst_til_last := 1;
} |
COMMA_T inst_key_value_pairs SEMICOLON_T ;

----------------------------------------
--- Constant Values Definition Rules ---
----------------------------------------

define_constant_values :
CONSTANT_VALUES_TAG {
    value_definition_type := CONSTANT_VALUE_TYPE;
}
constant_values;

constant_values :
constant_value_definition |
constant_values constant_value_definition;

constant_value_definition :
IDENTIFIER_T {
    GDB_INITIAL_VALUES.Clear_Last_Value;
}
COMMA_T value_definition SEMICOLON_T {
    declare
        Exists : Boolean;
        Val : String(1..6) := $1(1..6);
    begin
```

```
        --- Check for bad name
        case_conversion.to_upper(Val);
        if Val = "TRUE  " or Val = "FALSE " then
            Report_Error(BAD_CONSTANT_VALUE_NAME, $1(1..identifier_size));
        else
            GDB_DEFINITIONS.Define_Constant_Value($1(1..Identifier_Size),
                                                  GDB_INITIAL_VALUES.Current, Exists);
            if Exists then
                Report_Error(DUP_CONSTANT_VALUE, $1(1..identifier_size));
            end if;
        end if;
    end;
};


-------------------------------------
-- Class Definition Rules
-------------------------------------
class_definition :
class_header class_info;

class_identifier :
IDENTIFIER_T {
    $$:=$1;
} |
SPECIAL_IDENTIFIER_T {
    $$:=$1;
};

class_header :
CLASS_TAG class_identifier RIGHT_SQUARE_BRACKET_T {
    class_name := $2(1..identifier_size);
    current_state := CLASS;
    Add_Class( Name       => class_name,
               Class_Link => the_class_link);
};

class_info :
til_name class_objects |
class_objects {
    declare
        Found : boolean;
    begin
        Add_Protocol( Name       => default_til,
                      Class_Link => the_class_link);
        GDB_PROTOCOL.Lookup_Protocol( Name => default_til, Found => Found);
    end;
};

til_name :
TIL_TAG IDENTIFIER_T RIGHT_SQUARE_BRACKET_T {
    declare
        Found : boolean;
    begin
        Add_Protocol( Name       => $2(1..identifier_size),
                      Class_Link => the_class_link);
        GDB_PROTOCOL.Lookup_Protocol( Name => $2(1..identifier_size), Found => Found);
    end;
};


class_objects :
types_definition attributes_definition |
attributes_definition;


-------------------------------------
-- Attribute Definition Rules
-------------------------------------
attributes_definition :
ATTRIBUTES_TAG attribute_list;
```

```
attribute_list :
an_asterisk_declaration |
attributes;

an_asterisk_declaration :
asterisk_name an_attribute_definition;

asterisk_name  :
ASTERISK_T {
    BEGIN
        -- probably will need to check on gdb access to
        -- make sure they don't access it with a dot in
        -- the middle of the PUI. Just store spaces
        -- as the attribute name.
        attribute_name := (others => ' ');
    END;
};

attributes :
normal_name an_attribute_definition |
attributes normal_name an_attribute_definition;

normal_name :
IDENTIFIER_T {
    BEGIN
        attribute_name := $1(1..identifier_size);
    END;
};

an_attribute_definition :
COMMA_T IDENTIFIER_T COMMA_T INTEGER_T COMMA_T IDENTIFIER_T COMMA_T IDENTIFIER_T opt_attr_info {
    BEGIN
        attribute_type := $2(1..identifier_size);
        begin
            array_size     := User_Array_Range_T'Value($4);
            if array_size < 1 then
                Report_Error(BAD_ATTR_ARRAY_SIZE, $4);
                array_size := 1;
            end if;
        exception
            when others =>
                Report_Error(BAD_ATTR_ARRAY_SIZE, $4);
                array_size := 1;
        end;
        -- now convert the write_flag and machine_type to their
        -- respective types from strings
        begin
            write_flag     := Write_Flag_T'Value($6);
        exception
            when others =>
                Report_Error(BAD_ATTR_WRITE_FLAG, $6);
                write_flag := READ_ONLY;
        end;
        declare
            Alt_machine_Found : boolean := false;
        begin
            declare
                type Alt_Machine_T is (BIG_ENDIAN, LITTLE_ENDIAN);
                Alt_Machine : Alt_Machine_T;
            begin
                Alt_Machine := Alt_Machine_T'Value($8);
                if Alt_Machine = BIG_ENDIAN then
                    Alt_Machine_Found := true;
                    Machine := Sun;
                elsif Alt_Machine = LITTLE_ENDIAN then
                    Alt_Machine_Found := true;
                    Machine := Intel;
                end if;
            exception
                when others => null;
```

```
            end;
            if not Alt_Machine_Found then
                machine       := Machine_Type_T'Value($8);
            end if;
        exception
            when others =>
                Report_Error(BAD_ATTR_MACHINE_TYPE, $8);
                machine := Sun;
        end;

        -- Now add the attribute info to the database.
        Add_Attribute( Name              => attribute_name,
                       AttType           => attribute_type,
                       Array_Size        => array_size,
                       Write_Flag        => write_flag,
                       Machine_Type      => machine,
                       Class_Link        => the_class_link,
                       Byte_Offset_Valid => attr_byte_offset_valid,
                       Byte_Offset       => attr_byte_offset,
                       Protocol_Info     => attr_til_info);
        attr_til_last := 1;
        attr_til_info := (others => ' ');

    END;
};

opt_attr_info :
SEMICOLON_T {
    attr_byte_offset_valid := false;
    attr_byte_offset := 0;
    attr_til_info := (others => ' ');
    attr_til_last := 1;
} |
COMMA_T opt_attr_info1;

opt_attr_info1 :
byte_offset opt_attr_info2 |
attr_key_value_pairs SEMICOLON_T {
    attr_byte_offset_valid := false;
    attr_byte_offset := 0;
};

opt_attr_info2 :
SEMICOLON_T {
    attr_til_info := (others => ' ');
    attr_til_last := 1;
} |
COMMA_T attr_key_value_pairs SEMICOLON_T;


attr_key_value_pairs :
attr_til_info |
attr_key_value_pairs attr_til_info;

attr_til_info :
LEFT_BRACE_T IDENTIFIER_T TIL_INFO_T {
    declare
        i1, i2 : integer := $3'last;
        name   : name_T := (others=> ' ');
        Value_Rec : GDB_INITIAL_VALUES.Value_Record;
        Exists : boolean;
        Value : YYSType;
    begin
        -- Determine length of TIL info value
        i1 := 1;
        for i in 2..$3'last loop
           if i1 = 1 and then
             (($3(i) /= ' ') and ($3(i) /= ASCII.HT)) then
               i1 := i;
           end if;
           if $3(i) = RBRACE then
```

```
                i2 := i-1;
                exit;
            end if;
        end loop;

        -- Attempt to resolve constant value name
        if (i2+1-i1) <= name'last then
            name(1..(i2+1-i1)) := $3(i1..i2);
            GDB_DEFINITIONS.Lookup_Constant_Value(Name, Value_Rec, Exists);
        else
            Exists := false;
        end if;

        -- If constant value was resolved, then set attribute value to it,
        -- otherwise use token
        if Exists then
            Value(1..integer(Value_Rec.String_Length)+2)            :=            ":"            &
Value_Rec.String_Value(1..integer(Value_Rec.String_Length)) & "}";
        else
            Value := $3;
        end if;

        -- Copy Values to attribute til info string
        attr_til_info(attr_til_last) := '{';
        attr_til_last := attr_til_last + 1;
        for i in $2'range loop
            if $2(i) = ' '  then
                exit;
            end if;
            attr_til_info(attr_til_last) := $2(i);
            attr_til_last := attr_til_last + 1;
        end loop;
        for i in value'range loop
            attr_til_info(attr_til_last) := value(i);
            attr_til_last := attr_til_last + 1;
            if value(i) = RBRACE then
               exit;
            end if;
        end loop;
    exception
        when others =>
            report_error(TIL_INFO_TOO_LONG);
    end;
};

byte_offset :
INTEGER_T {
    BEGIN
        attr_byte_offset_valid := false;
        attr_byte_offset := attribute_offset_t'value($1);
        if  (attr_byte_offset  <  0)  or  (attr_byte_offset  >=  GDB_INITIAL_VALUES.MAX_OBJECT_SIZE)
then
            Report_Error(BAD_ATTR_BYTE_OFFSET, $1);
            attr_byte_offset := 0;
        else
            attr_byte_offset_valid := true;
        end if;
    exception
        when others =>
            Report_Error(BAD_ATTR_BYTE_OFFSET, $1);
            attr_byte_offset := 0;
    END;
};


------------------------------
-- Type Definition Rules
------------------------------
types_definition :
TYPES_TAG |
TYPES_TAG types;
```

```
types :
a_type_declaration |
types a_type_declaration;

a_type_declaration :
string_type;

string_type :
IDENTIFIER_T COMMA_T IDENTIFIER_T COMMA_T INTEGER_T SEMICOLON_T {
    declare
        type USER_TYPE_T is (STRING);
    BEGIN
        if USER_TYPE_T'VALUE($3) = STRING then
            string_name   := $1(1..identifier_size);
            begin
                string_length := User_String_Range_T'Value($5);
            EXCEPTION
                WHEN others =>
                    IF current_state = CLASS THEN
                        Report_Error(BAD_ATTR_STRING_LENGTH, $5);
                    ELSIF current_state = COMMAND THEN
                        Report_Error(BAD_PARAM_STRING_LENGTH, $5);
                    END IF;
                    String_Length := 1;
            end;
            IF current_state = CLASS THEN

                Add_String_Type( Name       => string_name,
                                 Length     => string_length,
                                 Class_Link => the_class_link);

            ELSIF current_state = COMMAND THEN

                Add_String_Type( Name         => string_name,
                                 Length       => string_length,
                                 Command_Link => the_command_link);
            END IF;
        ELSE
            Report_Error(BAD_USER_DATA_TYPE, $3);
        END IF;
    EXCEPTION
        WHEN others =>
            Report_Error(BAD_USER_DATA_TYPE, $3);

    END;

};

--enumeration_type : IDENTIFIER_T COMMA_T ENUM_T COMMA_T
--                  {  enum_name := $1(1..identifier_size);
--
--                     IF current_state = CLASS THEN
--
--                        Add_Enumeration_Type( Name       => enum_name,
--                                              Class_Link => the_class_link,
--                                              Type_Link  => the_type_link);
--
--                     ELSIF current_state = COMMAND THEN
--
--                        Add_Enumeration_Type (Name         => enum_name,
--                                              Command_link => the_command_link,
--                                              Type_Link    => the_type_link);
--
--                     END IF;
--
--                  }
--                  enum_description SEMICOLON_T;
--
--
--enum_description : LEFT_PARENTHESIS_T enum_values RIGHT_PARENTHESIS_T;
```

```
--
--enum_values : an_enum_value | an_enum_value COMMA_T enum_values;
--
--an_enum_value : IDENTIFIER_T ARROW_T INTEGER_T
--                { BEGIN
--                     enum_id := $1(1..identifier_size);
--                     internal_code := User_Internal_Code_T'Value($3);
--
--                     Add_Enumeration_Pair( Enum_Identifier => enum_id,
--                                           Internal_Code   => internal_code,
--                                           Type_Link       => the_type_link);
--
--                  EXCEPTION
--                     WHEN CONSTRAINT_ERROR =>
--                        Report_Error(BAD_ENUMERATION_VALUE, $3);
--                  END;
--
--                };



---------------------------------
-- Command Defintion Rules
---------------------------------
command_definition :
command_header  protocol_definition  remote_tis_definition  machine_definition  shell_definition
shell_overlays_definition {
    declare
        Found : Boolean;
        Id    : TYPES.Long_Integer_T;
        Overlay_Name : Name_T := (others => ' ');
        Overlay_Type : Name_T := (others => ' ');
    begin

        --- If command Id is defined ...
        if $1(1) /= ' ' then

            --- Look up command id definition
            GDB_DEFINITIONS.Lookup_Command_Id($1(1..Identifier_Size), Id, Found);

            --- If command id was found, set shell overlay value
            if Found then
                GDB_INITIAL_VALUES.Set_Int_Value(Id, $1(1..identifier_size));
                Overlay_Name(1..20) := "$COMMAND_ID         ";
                Overlay_Type(1..12) := "LONG_NATURAL";
                Overlay_Machine_Type := sun;
                Add_Shell_Overlay(Overlay_Name, Overlay_Type, COMMAND_ID_BYTE_OFFSET);
            else
                Report_Error(BAD_COMMAND_ID_REF,     $1(1..Identifier_Size),     true,     true,
trim(Command_Line));
            end if;
        end if;

        --- If routing protocol is defined as UDP
        if Routing_Protocol = UDP then

            --- Set routing protocol shell overlay
                GDB_INITIAL_VALUES.Set_Int_Value(2, "2");
                Overlay_Name(1..20) := "$ROUTING_PROTOCOL   ";
                Overlay_Type(1..12) := "NATURAL     ";
                Overlay_Machine_Type := sun;
                Add_Shell_Overlay(Overlay_Name, Overlay_Type, ROUTING_PROTOCOL_BYTE_OFFSET);

            --- Set routing port shell overlay
                GDB_INITIAL_VALUES.Set_Int_Value(Routing_Port,
Trim(Types.Long_Integer_T'Image(Routing_Port)));
                Overlay_Name(1..20) := "$ROUTING_PORT       ";
                Overlay_Type(1..12) := "LONG_NATURAL";
                Overlay_Machine_Type := sun;
                Add_Shell_Overlay(Overlay_Name, Overlay_Type, ROUTING_PORT_BYTE_OFFSET);
```

```
            --- Set routing host shell overlay
                GDB_INITIAL_VALUES.Set_String_Value(Routing_Host, Routing_Host'Length);
                Overlay_Name(1..20) := "$ROUTING_HOST       ";
                Overlay_Type(1..12) := "STRING      ";
                Overlay_Machine_Type := sun;
                Add_Shell_Overlay(Overlay_Name, Overlay_Type, ROUTING_HOST_BYTE_OFFSET);

            --- Set routing packet size shell overlay
                GDB_INITIAL_VALUES.Set_Int_Value(Routing_Size,
Trim(Types.Long_Integer_T'Image(Routing_Size)));
                Overlay_Name(1..20) := "$ROUTING_SIZE       ";
                Overlay_Type(1..12) := "LONG_NATURAL";
                Overlay_Machine_Type := sun;
                Add_Shell_Overlay(Overlay_Name, Overlay_Type, ROUTING_SIZE_BYTE_OFFSET);

        end if;

        --- If routing to a remote TIS is specified
        if Remote_Tis_Flag then

            --- Set routing path shell overlay
                GDB_INITIAL_VALUES.Set_Int_Value(1, "1");
                Overlay_Name(1..20) := "$ROUTING_PATH       ";
                Overlay_Type(1..12) := "NATURAL     ";
                Overlay_Machine_Type := sun;
                Add_Shell_Overlay(Overlay_Name, Overlay_Type, ROUTING_PATH_BYTE_OFFSET);

            --- Set session name shell overlay
                GDB_INITIAL_VALUES.Set_String_Value(Session_Name, Session_Name'Length);
                Overlay_Name(1..20) := "$SESSION_NAME       ";
                Overlay_Type(1..12) := "STRING      ";
                Overlay_Machine_Type := sun;
                Add_Shell_Overlay(Overlay_Name, Overlay_Type, SESSION_NAME_BYTE_OFFSET);

        end if;

        --- Add command to tables
        Add_Command( Name         => command_name,
                     Machine_Type => machine_type,
                     Command_Shell=> command_shell,
                     Command_Link => the_command_link);
    end;
}
command_types_definition parameters_definition;

command_types_definition :
types_definition |
;


command_header :
COMMAND_TAG IDENTIFIER_T RIGHT_SQUARE_BRACKET_T {
    declare
        Line_Number : constant string := Get_Current_Input_Line;
    begin
        Command_Name  := $2(1..Identifier_Size);
        Current_State := COMMAND;
        Command_Shell := (others => 0);
        Command_Line  := (others => ' ');
        Command_Line(1..line_number'length) := Line_Number;
        $$ := (others => ' ');
        GDB_DEFINITIONS.Delete_Shell_Overlays;
    end;
} |
COMMAND_TAG IDENTIFIER_T BASED_ON_T IDENTIFIER_T RIGHT_SQUARE_BRACKET_T {
    declare
        Line_Number : constant string := Get_Current_Input_Line;
    begin
        command_name := $2(1..identifier_size);
        current_state := COMMAND;
        command_shell := (others => 0);
```

```
        Command_Line  := (others => ' ');
        Command_Line(1..line_number'length) := Line_Number;
        $$ := $4;
        GDB_DEFINITIONS.Delete_Shell_Overlays;
    end;

};


protocol_definition :
UDP_PROTOCOL_TAG routing_host COMMA_T routing_port routing_size {
    Routing_Protocol := UDP;
} |
{
    Routing_Protocol := NONE;
}
;

remote_tis_definition :
REMOTE_TIS_TAG tis_session_name SEMICOLON_T {
  Remote_Tis_Flag := true;
} |
{
  Remote_Tis_Flag := false;
}
;

routing_host :
STRING_VALUE_T {
    declare
        str_len   : integer := 0;
    begin
        Routing_Host := (others => ' ');
        for i in 2..$1'length loop
            if $1(i) = $1(1) then
                str_len := i - 2;
                exit;
            elsif i-1 <= Routing_Host'length then
                Routing_Host(i-1) := $1(i);
            end if;
        end loop;
        if (str_len > Routing_Host'length) or
           (str_len = 0 ) then
            Report_Error(BAD_ROUTING_HOST, $1);
            Routing_Host := (others => ' ');
        end if;
    end;
} |
IDENTIFIER_T {
    declare
        use GDB_INITIAL_VALUES;
        Value : GDB_INITIAL_VALUES.Value_Record;
        Exists : Boolean;
    begin
        Routing_Host := (others => ' ');
        GDB_DEFINITIONS.Lookup_Constant_Value($1(1..Identifier_Size), Value, Exists);
        if exists then
            if Value.Value_Type = GDB_INITIAL_VALUES.STR and then
              Value.String_Length <= Routing_Host'length then
                Routing_Host(1..integer(Value.String_Length))                                    :=
Value.String_Value(1..integer(Value.String_Length));
            else
                Report_Error(BAD_ROUTING_HOST,
Value.String_Value(1..integer(Value.String_Length)));
            end if;
        else
            Report_Error(BAD_ROUTING_HOST, $1(1..Identifier_Size));
        end if;
    end;
};
```

```
tis_session_name :
STRING_VALUE_T {
    declare
        str_len   : integer := 0;
    begin
        Session_Name := (others => ' ');
        for i in 2..$1'length loop
            if $1(i) = $1(1) then
                str_len := i - 2;
                exit;
            elsif i-1 <= Session_Name'length then
                Session_Name(i-1) := $1(i);
            end if;
        end loop;
        if (str_len > Session_Name'length) or
           (str_len = 0 ) then
            Report_Error(BAD_SESSION_NAME, $1);
            Session_Name := (others => ' ');
        end if;
    end;
} |
IDENTIFIER_T {
    declare
        use GDB_INITIAL_VALUES;
        Value : GDB_INITIAL_VALUES.Value_Record;
        Exists : Boolean;
    begin
        Session_Name := (others => ' ');
        GDB_DEFINITIONS.Lookup_Constant_Value($1(1..Identifier_Size), Value, Exists);
        if exists then
            if Value.Value_Type = GDB_INITIAL_VALUES.STR and then
              Value.String_Length <= Session_Name'length then
                Session_Name(1..integer(Value.String_Length))                    :=
Value.String_Value(1..integer(Value.String_Length));
            else
                Report_Error(BAD_SESSION_NAME,
Value.String_Value(1..integer(Value.String_Length)));
            end if;
        else
            Report_Error(BAD_SESSION_NAME, $1(1..Identifier_Size));
        end if;
    end;
};

routing_port :
INTEGER_T {
    begin
        --- Determine and range check routing protocol size
        Routing_Port := Types.Long_Integer_T'Value($1);
        if Routing_Port < 1 or Routing_Port > 65535 then
            Report_Error(BAD_ROUTING_PORT, $1);
            Routing_Port := 0;
        end if;
    exception
        when others =>
            Report_Error(BAD_ROUTING_PORT, $1);
            Routing_Port := 0;
    end;
} |
IDENTIFIER_T {
    declare
        use GDB_INITIAL_VALUES;
        Value : GDB_INITIAL_VALUES.Value_Record;
        Exists : Boolean;
    begin
        GDB_DEFINITIONS.Lookup_Constant_Value($1(1..Identifier_Size), Value, Exists);
        if exists then
            if Value.Value_Type = GDB_INITIAL_VALUES.INT then
                Routing_Port := Types.Long_Integer_T(Value.Int_Value);
            elsif Value.Value_Type = GDB_INITIAL_VALUES.FLOAT then
                Routing_Port := Types.Long_Integer_T(Value.Float_Value);
```

```
            else
                Routing_Port := 0;
            end if;
            if Routing_Port < 1 or Routing_Port > 65535 then
                Report_Error(BAD_ROUTING_PORT,
Value.String_Value(1..integer(Value.String_Length)));
                Routing_Port := 0;
            end if;
        else
            Report_Error(BAD_ROUTING_PORT, $1(1..Identifier_Size));
            Routing_Port := 0;
        end if;

    exception
        when others =>
            Report_Error(BAD_ROUTING_PORT, Value.String_Value(1..integer(Value.String_Length)));
            Routing_Port := 0;
    end;
};

routing_size :
COMMA_T INTEGER_T SEMICOLON_T {
    begin
        --- Determine and range check routing protocol size
        Routing_Size := Types.Long_Integer_T'Value($2);
        if Routing_Size < 1 or Routing_Size > 32768 then
            Report_Error(BAD_ROUTING_SIZE, $2);
            Routing_Size := 0;
        end if;
    exception
        when others =>
            Report_Error(BAD_ROUTING_SIZE, $2);
            Routing_Size := 0;
    end;
} |
COMMA_T IDENTIFIER_T SEMICOLON_T {
    declare
        use GDB_INITIAL_VALUES;
        Value : GDB_INITIAL_VALUES.Value_Record;
        Exists : Boolean;
    begin
        GDB_DEFINITIONS.Lookup_Constant_Value($2(1..Identifier_Size), Value, Exists);
        if exists then
            if Value.Value_Type = GDB_INITIAL_VALUES.INT then
                Routing_Size := Types.Long_Integer_T(Value.Int_Value);
            elsif Value.Value_Type = GDB_INITIAL_VALUES.FLOAT then
                Routing_Size := Types.Long_Integer_T(Value.Float_Value);
            else
                Routing_Size := 0;
            end if;
            if Routing_Size < 1 or Routing_Size > 65535 then
                Report_Error(BAD_ROUTING_SIZE,
Value.String_Value(1..integer(Value.String_Length)));
                Routing_Size := 0;
            end if;
        else
            Report_Error(BAD_ROUTING_SIZE, $2(1..Identifier_Size));
            Routing_Size := 0;
        end if;

    exception
        when others =>
            Report_Error(BAD_ROUTING_SIZE, Value.String_Value(1..integer(Value.String_Length)));
            Routing_Size := 0;
    end;
} |
SEMICOLON_T {
    routing_size := 0;
};
```

```
--------------------------------
-- Machine Definition Rules
--------------------------------

machine_definition :
MACHINE_TAG IDENTIFIER_T RIGHT_SQUARE_BRACKET_T {
    declare
        Alt_machine_Found : boolean := false;
    begin
        declare
            type Alt_Machine_T is (BIG_ENDIAN, LITTLE_ENDIAN);
            Alt_Machine : Alt_Machine_T;
        begin
            Alt_Machine := Alt_Machine_T'Value($2);
            if Alt_Machine = BIG_ENDIAN then
                Alt_Machine_Found := true;
                Machine_Type := Sun;
            elsif Alt_Machine = LITTLE_ENDIAN then
                Alt_Machine_Found := true;
                Machine_Type := Intel;
            end if;
        exception
            when constraint_error => null;
        end;
        if not Alt_Machine_Found then
            machine_type       := Machine_Type_T'Value($2);
        end if;
    exception
        when Constraint_Error =>
            Report_Error(BAD_CMD_MACHINE_TYPE,$2);
            Machine_Type := Sun;
    END;
};


--*****************************************************************************
--*** Grammar for handling command shell overlay definitions                ***
--*****************************************************************************

shell_overlays_definition :
SHELL_OVERLAYS_TAG {
    offset_definition_type := OVERLAY_OFFSET_TYPE;
    next_cmd_offset := 0;
}
shell_overlays_defs
|
;

shell_overlays_defs :
shell_overlay_def |
shell_overlays_defs shell_overlay_def
;

shell_overlay_def :
IDENTIFIER_T COMMA_T IDENTIFIER_T COMMA_T command_byte_offset COMMA_T
      shell_overlay_value overlay_machine SEMICOLON_T {
    begin
        parameter_name := $1(1..identifier_size);
        parameter_type := $3(1..identifier_size);
        Add_Shell_Overlay(parameter_name, parameter_type, byte_offset);
    end;
};

command_byte_offset :
INTEGER_T {
    begin
        byte_offset       := byte_offset_T'Value($1);
    exception
        when others =>
            if offset_definition_type = OVERLAY_OFFSET_TYPE then
```

```
                Report_Error(BAD_OVERLAY_BYTE_OFFSET, $1);
            elsif offset_definition_type = PARAMETER_OFFSET_TYPE then
                Report_Error(BAD_PARAM_BYTE_OFFSET, $1);
            end if;
            byte_offset := 0;
    end;
} |
IDENTIFIER_T {
    declare
        Exists : Boolean;
        Value  : TYPES.Integer_T;
    begin
        GDB_Definitions.Lookup_Shell_Offset($1(1..Identifier_Size), Value, Exists);
        if Exists then
            byte_offset :=  byte_offset_T(value);
        else
            Report_Error(BAD_SHELL_LOCATION_REF, $1(1..Identifier_Size));
            byte_offset := 0;
        end if;
    exception
        when others =>
            if offset_definition_type = OVERLAY_OFFSET_TYPE then
                Report_Error(BAD_OVERLAY_BYTE_OFFSET, $1(1..Identifier_Size));
            elsif offset_definition_type = PARAMETER_OFFSET_TYPE then
                Report_Error(BAD_PARAM_BYTE_OFFSET, $1(1..Identifier_Size));
            end if;
            byte_offset := 0;
    end;
}|
PLUS_T {
    begin
        byte_offset := byte_offset_t(next_cmd_offset);
    exception
        when others =>
            if offset_definition_type = OVERLAY_OFFSET_TYPE then
                Report_Error(BAD_OVERLAY_BYTE_OFFSET, Types.Integer_T'image(Next_Cmd_Offset));
            elsif offset_definition_type = PARAMETER_OFFSET_TYPE then
                Report_Error(BAD_PARAM_BYTE_OFFSET, Types.Integer_T'image(Next_Cmd_Offset));
            end if;
            byte_offset := 0;
    end;

}|
IDENTIFIER_T PLUS_T INTEGER_T {
    declare
        Exists : Boolean;
        Value  : TYPES.Integer_T;
    begin
        GDB_Definitions.Lookup_Shell_Offset($1(1..Identifier_Size), Value, Exists);
        if Exists then
            byte_offset := byte_offset_T(value);
            byte_offset := byte_offset + byte_offset_t'value($3);
        else
            byte_offset := 0;
            Report_Error(BAD_SHELL_LOCATION_REF, $1);
        end if;
    exception
        when others =>
            byte_offset := 0;
            if offset_definition_type = OVERLAY_OFFSET_TYPE then
                Report_Error(BAD_OVERLAY_BYTE_OFFSET, Types.Integer_T'Image(Value)&"+"&$3);
            elsif offset_definition_type = PARAMETER_OFFSET_TYPE then
                Report_Error(BAD_PARAM_BYTE_OFFSET, Types.Integer_T'Image(Value)&"+"&$3);
            end if;
    end;
} |
INTEGER_T PLUS_T INTEGER_T {
    begin
        byte_offset := byte_offset_t'value($1);
        byte_offset := byte_offset + byte_offset_t'value($3);
    exception
```

```
        when others =>
            byte_offset := 0;
            if offset_definition_type = OVERLAY_OFFSET_TYPE then
                Report_Error(BAD_OVERLAY_BYTE_OFFSET, trim($1)&"+"&$3);
            elsif offset_definition_type = PARAMETER_OFFSET_TYPE then
                Report_Error(BAD_PARAM_BYTE_OFFSET, trim($1)&"+"&$3);
            end if;
    end;
} |
AFTER_T IDENTIFIER_T {
    declare
        Found : Boolean;
        Value : Types.Integer_T;
    begin
        GDB_Definitions.Lookup_Shell_Overlay($2(1..Identifier_Size), Command_Name, Value, Found);
        if found then
            byte_offset := byte_offset_t(value);
        else
            byte_offset := 0;
            Report_Error(BAD_SHELL_OVERLAY_REF, $2(1..Identifier_Size));
        end if;
    exception
        when others =>
            byte_offset := 0;
            if offset_definition_type = OVERLAY_OFFSET_TYPE then
                Report_Error(BAD_OVERLAY_BYTE_OFFSET, Types.Integer_T'Image(Value));
            elsif offset_definition_type = PARAMETER_OFFSET_TYPE then
                Report_Error(BAD_PARAM_BYTE_OFFSET, Types.Integer_T'Image(Value));
            end if;
    end;
};

overlay_machine :
COMMA_T IDENTIFIER_T {
    declare
        Alt_machine_Found : boolean := false;
    begin
        declare
            type Alt_Machine_T is (BIG_ENDIAN, LITTLE_ENDIAN);
            Alt_Machine : Alt_Machine_T;
        begin
            Alt_Machine := Alt_Machine_T'Value($2);
            if Alt_Machine = BIG_ENDIAN then
                Alt_Machine_Found := true;
                Overlay_Machine_Type := Sun;
            elsif Alt_Machine = LITTLE_ENDIAN then
                Alt_Machine_Found := true;
                Overlay_Machine_Type := Intel;
            end if;
        exception
            when constraint_error => null;
        end;
        if not Alt_Machine_Found then
            overlay_machine_type       := Machine_Type_T'Value($2);
        end if;
    exception
        when Constraint_Error =>
            Report_Error(BAD_OVERLAY_MACHINE_TYPE,$2);
            Overlay_Machine_Type := Machine_Type;
    end;
} |
{
    Overlay_Machine_Type := Machine_Type;
};


shell_overlay_value :
{
    GDB_INITIAL_VALUES.Clear_Last_Value;
    value_definition_type := OVERLAY_VALUE_TYPE;
}
```

```
value_definition;


--*****************************************************************************
--*** Grammar for handling binary shell definition                          ***
--*****************************************************************************

shell_definition :
SHELL_TAG byte_definitions |
SHELL_TAG ZEROS_T SEMICOLON_T;

byte_definitions :
BT1_8_T eight_byte_value {
    command_shell(1..8) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT9_16_T eight_byte_value {
    command_shell(9..16) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT17_24_T eight_byte_value {
    command_shell(17..24) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT25_32_T eight_byte_value {
    command_shell(25..32) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT33_40_T eight_byte_value {
    command_shell(33..40) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT41_48_T eight_byte_value {
    command_shell(41..48) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT49_56_T eight_byte_value {
    command_shell(49..56) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT57_64_T eight_byte_value {
    command_shell(57..64) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT65_72_T eight_byte_value {
    command_shell(65..72) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT73_80_T eight_byte_value {
    command_shell(73..80) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT81_88_T eight_byte_value {
    command_shell(81..88) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT89_96_T eight_byte_value {
    command_shell(89..96) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT97_104_T eight_byte_value {
    command_shell(97..104) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT105_112_T eight_byte_value {
    command_shell(105..112) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT113_120_T eight_byte_value {
    command_shell(113..120) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT121_128_T eight_byte_value {
    command_shell(121..128) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT129_136_T eight_byte_value {
    command_shell(129..136) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT137_144_T eight_byte_value {
    command_shell(137..144) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT145_152_T eight_byte_value {
    command_shell(145..152) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT153_160_T eight_byte_value {
    command_shell(153..160) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
```

```
}
BT161_168_T eight_byte_value {
    command_shell(161..168) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT169_176_T eight_byte_value {
    command_shell(169..176) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT177_184_T eight_byte_value {
    command_shell(177..184) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT185_192_T eight_byte_value {
    command_shell(185..192) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT193_200_T eight_byte_value {
    command_shell(193..200) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT201_208_T eight_byte_value {
    command_shell(201..208) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT209_216_T eight_byte_value {
    command_shell(209..216) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT217_224_T eight_byte_value {
    command_shell(217..224) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT225_232_T eight_byte_value {
    command_shell(225..232) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT233_240_T eight_byte_value {
    command_shell(233..240) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT241_248_T eight_byte_value {
    command_shell(241..248) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT249_256_T eight_byte_value {
    command_shell(249..256) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT257_264_T eight_byte_value {
    command_shell(257..264) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT265_272_T eight_byte_value {
    command_shell(265..272) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT273_280_T eight_byte_value {
    command_shell(273..280) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT281_288_T eight_byte_value {
    command_shell(281..288) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT289_296_T eight_byte_value {
    command_shell(289..296) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT297_304_T eight_byte_value {
    command_shell(297..304) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT305_312_T eight_byte_value {
    command_shell(305..312) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT313_320_T eight_byte_value {
    command_shell(313..320) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT321_328_T eight_byte_value {
    command_shell(321..328) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT329_336_T eight_byte_value {
    command_shell(329..336) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT337_344_T eight_byte_value {
    command_shell(337..344) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT345_352_T eight_byte_value {
```

```
    command_shell(345..352) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT353_360_T eight_byte_value {
    command_shell(353..360) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT361_368_T eight_byte_value {
    command_shell(361..368) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT369_376_T eight_byte_value {
    command_shell(369..376) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT377_384_T eight_byte_value {
    command_shell(377..384) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT385_392_T eight_byte_value {
    command_shell(385..392) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT393_400_T eight_byte_value {
    command_shell(393..400) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT401_408_T eight_byte_value {
    command_shell(401..408) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT409_416_T eight_byte_value {
    command_shell(409..416) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT417_424_T eight_byte_value {
    command_shell(417..424) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT425_432_T eight_byte_value {
    command_shell(425..432) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT433_440_T eight_byte_value {
    command_shell(433..440) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT441_448_T eight_byte_value {
    command_shell(441..448) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT449_456_T eight_byte_value {
    command_shell(449..456) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT457_464_T eight_byte_value {
    command_shell(457..464) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT465_472_T eight_byte_value {
    command_shell(465..472) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT473_480_T eight_byte_value {
    command_shell(473..480) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT481_488_T eight_byte_value {
    command_shell(481..488) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT489_496_T eight_byte_value {
    command_shell(489..496) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT497_504_T eight_byte_value {
    command_shell(497..504) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT505_512_T eight_byte_value {
    command_shell(505..512) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT513_520_T eight_byte_value {
    command_shell(513..520) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT521_528_T eight_byte_value {
    command_shell(521..528) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT529_536_T eight_byte_value {
    command_shell(529..536) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
```

```
BT537_544_T eight_byte_value {
    command_shell(537..544) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT545_552_T eight_byte_value {
    command_shell(545..552) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT553_560_T eight_byte_value {
    command_shell(553..560) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT561_568_T eight_byte_value {
    command_shell(561..568) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT569_576_T eight_byte_value {
    command_shell(569..576) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT577_584_T eight_byte_value {
    command_shell(577..584) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT585_592_T eight_byte_value {
    command_shell(585..592) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT593_600_T eight_byte_value {
    command_shell(593..600) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT601_608_T eight_byte_value {
    command_shell(601..608) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT609_616_T eight_byte_value {
    command_shell(609..616) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT617_624_T eight_byte_value {
    command_shell(617..624) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT625_632_T eight_byte_value {
    command_shell(625..632) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT633_640_T eight_byte_value {
    command_shell(633..640) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT641_648_T eight_byte_value {
    command_shell(641..648) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT649_656_T eight_byte_value {
    command_shell(649..656) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT657_664_T eight_byte_value {
    command_shell(657..664) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT665_672_T eight_byte_value {
    command_shell(665..672) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT673_680_T eight_byte_value {
    command_shell(673..680) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT681_688_T eight_byte_value {
    command_shell(681..688) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT689_696_T eight_byte_value {
    command_shell(689..696) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT697_704_T eight_byte_value {
    command_shell(697..704) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT705_712_T eight_byte_value {
    command_shell(705..712) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT713_720_T eight_byte_value {
    command_shell(713..720) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT721_728_T eight_byte_value {
    command_shell(721..728) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
```

```
}
BT729_736_T eight_byte_value {
    command_shell(729..736) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT737_744_T eight_byte_value {
    command_shell(737..744) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT745_752_T eight_byte_value {
    command_shell(745..752) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT753_760_T eight_byte_value {
    command_shell(753..760) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT761_768_T eight_byte_value {
    command_shell(761..768) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT769_776_T eight_byte_value {
    command_shell(769..776) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT777_784_T eight_byte_value {
    command_shell(777..784) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT785_792_T eight_byte_value {
    command_shell(785..792) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT793_800_T eight_byte_value {
    command_shell(793..800) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT801_808_T eight_byte_value {
    command_shell(801..808) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT809_816_T eight_byte_value {
    command_shell(809..816) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT817_824_T eight_byte_value {
    command_shell(817..824) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT825_832_T eight_byte_value {
    command_shell(825..832) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT833_840_T eight_byte_value {
    command_shell(833..840) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT841_848_T eight_byte_value {
    command_shell(841..848) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT849_856_T eight_byte_value {
    command_shell(849..856) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT857_864_T eight_byte_value {
    command_shell(857..864) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT865_872_T eight_byte_value {
    command_shell(865..872) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT873_880_T eight_byte_value {
    command_shell(873..880) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT881_888_T eight_byte_value {
    command_shell(881..888) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT889_896_T eight_byte_value {
    command_shell(889..896) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT897_904_T eight_byte_value {
    command_shell(897..904) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT905_912_T eight_byte_value {
    command_shell(905..912) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT913_920_T eight_byte_value {
```

```
    command_shell(913..920) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT921_928_T eight_byte_value {
    command_shell(921..928) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT929_936_T eight_byte_value {
    command_shell(929..936) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT937_944_T eight_byte_value {
    command_shell(937..944) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT945_952_T eight_byte_value {
    command_shell(945..952) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT953_960_T eight_byte_value {
    command_shell(953..960) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT961_968_T eight_byte_value {
    command_shell(961..968) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT969_976_T eight_byte_value {
    command_shell(969..976) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT977_984_T eight_byte_value {
    command_shell(977..984) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT985_992_T eight_byte_value {
    command_shell(985..992) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT993_1000_T eight_byte_value {
    command_shell(993..1000) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT1001_1008_T eight_byte_value {
    command_shell(1001..1008) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT1009_1016_T eight_byte_value {
    command_shell(1009..1016) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
}
BT1017_1024_T eight_byte_value {
    command_shell(1017..1024) :=(byte_1,byte_2,byte_3,byte_4,byte_5,byte_6,byte_7,byte_8);
};

eight_byte_value :
ARROW_T byte_value_def {
    byte_1 := byte_value;
}
COMMA_T byte_value_def {
    byte_2 := byte_value;
}
COMMA_T byte_value_def {
    byte_3 := byte_value;
}
COMMA_T byte_value_def {
    byte_4 := byte_value;
}
COMMA_T byte_value_def {
    byte_5 := byte_value;
}
COMMA_T byte_value_def {
    byte_6 := byte_value;
}
COMMA_T byte_value_def {
    byte_7 := byte_value;
}
COMMA_T byte_value_def {
    byte_8 := byte_value;
}
SEMICOLON_T;

byte_value_def :
HEX_INTEGER_T {
```

```
    begin
        --it is supposed to be in HEX.
        --location rule is similar but decided to duplicate
        --code since each are tied up with different ideas
        hex_num_str := $1(1..identifier_size);
        --Got to get rid of the `H' in the string and
        --then add a `16#' to the front and a `#' to
        --the back so it can be converted to an integer
        --using 'Value.
        for i in hex_num_str'range loop
            if (hex_num_str(i) = 'H') or
              (hex_num_str(i) = 'h') then -- get rid of the 'H'
                hex_num_str(i) := '#';
            end if;
        end loop;
        hex_num_str := "16#" &
          hex_num_str(1..identifier_size-3);

        --now try to convert to an integer
        begin
            byte_value := byte_Value_T'Value(hex_num_str);
        exception
            when others => --there was a problem in the conversion
                --so set the value to 0 so GDB won't
                --blow up.
                Report_Error(BAD_SHELL_BYTE_VALUE, $1);
                byte_value := 0;
        end;
    end;
} |
IDENTIFIER_T {
    begin
        --it is supposed to be in HEX.
        --location rule is similar but decided to duplicate
        --code since each are tied up with different ideas
        hex_num_str := $1(1..identifier_size);
        --Got to get rid of the `H' in the string and
        --then add a `16#' to the front and a `#' to
        --the back so it can be converted to an integer
        --using 'Value.
        for i in hex_num_str'range loop
            if (hex_num_str(i) = 'H') or
               (hex_num_str(i) = 'h') then -- get rid of the 'H'
                  hex_num_str(i) := '#';
            end if;
        end loop;
        hex_num_str := "16#" & hex_num_str(1..identifier_size-3);

        --now try to convert to an integer
        begin
            byte_value := byte_Value_T'Value(hex_num_str);
        exception
            when others =>
                --there was a problem in the conversion
                --so set the value to 0 so GDB won't blow up.
                Report_Error(BAD_SHELL_BYTE_VALUE, $1);
                byte_value := 0;
        end;
    end;
} |
BIN_INTEGER_T {
    begin
        --it is supposed to be in BIN.
        --location rule is similar but decided to duplicate
        --code since each are tied up with different ideas
        hex_num_str := $1(1..identifier_size);
        --Got to get rid of the `H' in the string and
        --then add a `16#' to the front and a `#' to
        --the back so it can be converted to an integer
        --using 'Value.
        large_num_flag := false;
```

```
        for i in hex_num_str'range loop
            if (hex_num_str(i) = 'B') or
              (hex_num_str(i) = 'b') then -- get rid of the 'B'
                hex_num_str(i) := '#';
                if i > 32 then
                    if hex_num_str(i-32) = '1' then
                        hex_num_str(i-32) := '0';
                        large_num_flag := true;
                    end if;
                end if;
            end if;
        end loop;
        hex_num_str := "2#" &
          hex_num_str(1..identifier_size-2);

        --now try to convert to an integer
        begin
            byte_value := byte_Value_T'Value(hex_num_str);
        exception
            when others => --there was a problem in the conversion
                --so set the value to 0 so GDB won't
                --blow up.
                Report_Error(BAD_SHELL_BYTE_VALUE, $1);
                byte_value := 0;
        end;
    end;
};


--*****************************************************************************
--*** Grammar for handling command parameter definition                     ***
--*****************************************************************************

parameters_definition :
PARAMETERS_TAG {
    next_cmd_offset := 0;
    offset_definition_type := PARAMETER_OFFSET_TYPE;
}|
PARAMETERS_TAG {
    next_cmd_offset := 0;
    offset_definition_type := PARAMETER_OFFSET_TYPE;
}
parameters;

parameters :
a_parameter |
parameters a_parameter;

a_parameter :
IDENTIFIER_T COMMA_T IDENTIFIER_T COMMA_T INTEGER_T COMMA_T command_byte_offset SEMICOLON_T {
    BEGIN
        parameter_name := $1(1..identifier_size);
        parameter_type := $3(1..identifier_size);
        --get the array size
        begin
            array_size        := User_Array_Range_T'Value($5);
            if array_size < 1 then
                Report_Error(BAD_PARAM_ARRAY_SIZE, $5);
                Array_size := 0;
            end if;
        EXCEPTION
            WHEN OTHERS =>
                Report_Error(BAD_PARAM_ARRAY_SIZE, $5);
                Array_size := 0;
        end;

        Add_Parameter( Name         => parameter_name,
                       PrmType      => parameter_type,
                       Array_Size   => array_size,
                       Byte_Offset  => byte_offset,
                       Bit_Offset   => 0,
```

```
                        Command_Link => the_command_link,
                        Next_Offset  => Next_Cmd_Offset);
    END;
};


--*****************************************************************************
--*** Grammar for handling object instance definitions                      ***
--*****************************************************************************

instance_definition :
instance_header |
instance_header instances;

instance_header :
INSTANCES_TAG;

instances :
an_instance |
instances an_instance;

an_instance :
LEFT_PARENTHESIS_T IDENTIFIER_T COMMA_T class_identifier COMMA_T INTEGER_T opt_inst_info {
    BEGIN                                              -- without PUI
        --now try to convert to an integer
        begin
            instance_id := Instance_Id_T'Value($6(1..identifier_size));
        exception
            when others => --there was a problem in the conversion
                --so set the location to 0 so GDB won't
                --blow up.
                Report_Error(BAD_INST_ID, $6);
                instance_id := -1;
        end;
        -- need to reset the instance_location_ctr to the
        -- new location
        GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr:= instance_id;

        instance_name := $2(1..identifier_size);
        pui_name      := $2(1..identifier_size);
        class_name    := $4(1..identifier_size);
        bit_offset    := 0;

        --increment the instance counter to keep track of order.
        instance_ctr := instance_ctr + 1;

        Add_Instance ( Instance_Name  => instance_name,
                       Alternate_Name => pui_name,
                       Class_Name     => class_name,
                       Location_Id    => instance_id,
                       Bit_Offset     => bit_offset,
                       Instance_Ctr   => instance_ctr,
                       Protocol_Info  => inst_til_info,
                       Offset         => next_instance_loc_offset);
        inst_til_last := 1;
        inst_til_info := (others => ' ');

        -- The size of each attribute is computed in gdbbpt.
        -- This gives us offsets for address locations.
        -- Changed add_instance so that it returns the offset
        -- in next_instance_loc_offset.

    END;
} |
LEFT_PARENTHESIS_T HEX_INTEGER_T COMMA_T class_identifier COMMA_T INTEGER_T opt_inst_info {
    BEGIN                                              -- without PUI
        --now try to convert to an integer
        begin
            instance_id := Instance_Id_T'Value($6(1..identifier_size));
        exception
            when others => --there was a problem in the conversion
```

```
                --so set the location to 0 so GDB won't
                --blow up.
                Report_Error(BAD_INST_ID, $6);
                instance_id := -1;
        end;
        -- need to reset the instance_location_ctr to the
        -- new location
        GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr:= instance_id;

        instance_name := $2(1..identifier_size);
        if instance_name(1) in '0'..'9' then
            report_error(PARSING_ERROR, instance_name);
        end if;
        pui_name      := $2(1..identifier_size);
        class_name    := $4(1..identifier_size);
        bit_offset    := 0;

        --increment the instance counter to keep track of order.
        instance_ctr := instance_ctr + 1;

        Add_Instance ( Instance_Name  => instance_name,
                       Alternate_Name => pui_name,
                       Class_Name     => class_name,
                       Location_Id    => instance_id,
                       Bit_Offset     => bit_offset,
                       Instance_Ctr   => instance_ctr,
                       Protocol_Info  => inst_til_info,
                       Offset         => next_instance_loc_offset);
        inst_til_last := 1;
        inst_til_info := (others => ' ');
        -- The size of each attribute is computed in gdbbpt.
        -- This gives us offsets for address locations.
        -- Changed add_instance so that it returns the offset
        -- in next_instance_loc_offset.
    END;
} |
LEFT_PARENTHESIS_T IDENTIFIER_T COMMA_T class_identifier COMMA_T PLUS_T opt_inst_info {
    BEGIN                                          -- without PUI
        -- Need to compute the address from previous
        -- address locations.  If no address previously
        -- set, then gets initialized to default value of 0.
        if GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr < long_integer_t'last then
            GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr                                    :=
GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr + 1;
        else
            report_error(BAD_INST_ID, "+");
            GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr := -1;
        end if;
        instance_id := Instance_Id_T(GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr);

        instance_name := $2(1..identifier_size);
        pui_name      := $2(1..identifier_size);
        class_name    := $4(1..identifier_size);
        bit_offset    := 0;

        --increment the instance counter to keep track of order.
        instance_ctr := instance_ctr + 1;

        Add_Instance ( Instance_Name  => instance_name,
                       Alternate_Name => pui_name,
                       Class_Name     => class_name,
                       Location_Id    => instance_id,
                       Bit_Offset     => bit_offset,
                       Instance_Ctr   => instance_ctr,
                       Protocol_Info  => inst_til_info,
                       Offset         => next_instance_loc_offset);
        inst_til_last := 1;
        inst_til_info := (others => ' ');
        -- The size of each attribute is computed in gdbbpt.
        -- This gives us offsets for address locations.
        -- Changed add_instance so that it returns the offset
```

```
        -- in next_instance_loc_offset.
    END;
} |
LEFT_PARENTHESIS_T HEX_INTEGER_T COMMA_T class_identifier COMMA_T PLUS_T opt_inst_info {
    BEGIN                                                -- without PUI
        -- Need to compute the address from previous
        -- address locations.  If no address previously
        -- set, then gets initialized to default value of 0.
        if GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr < long_integer_t'last then
            GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr                                    :=
GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr + 1;
        else
            report_error(BAD_INST_ID, "+");
            GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr := -1;
        end if;
        instance_id := Instance_Id_T(GDB_FILE_SYSTEM_INTERFACE.instance_location_ctr);

        instance_name := $2(1..identifier_size);
        if instance_name(1) in '0'..'9' then
            report_error(PARSING_ERROR, instance_name);
        end if;
        pui_name      := $2(1..identifier_size);
        class_name    := $4(1..identifier_size);
        bit_offset    := 0;

        --increment the instance counter to keep track of order.
        instance_ctr := instance_ctr + 1;

        Add_Instance ( Instance_Name  => instance_name,
                       Alternate_Name => pui_name,
                       Class_Name     => class_name,
                       Location_Id    => instance_id,
                       Bit_Offset     => bit_offset,
                       Instance_Ctr   => instance_ctr,
                       Protocol_Info  => inst_til_info,
                       Offset         => next_instance_loc_offset);
        inst_til_last := 1;
        inst_til_info := (others => ' ');
        -- The size of each attribute is computed in gdbbpt.
        -- This gives us offsets for address locations.
        -- Changed add_instance so that it returns the offset
        -- in next_instance_loc_offset.

    END;
} ;

opt_inst_info :
RIGHT_PARENTHESIS_T {
    GDB_INITIAL_VALUES.Clear_Last_Value;
    inst_til_info := (others => ' ');
    inst_til_last := 1;
} |
COMMA_T opt_inst_info1;

opt_inst_info1 :
instance_value opt_inst_info2 |
inst_key_value_pairs RIGHT_PARENTHESIS_T {
    GDB_INITIAL_VALUES.Clear_Last_Value;
};

opt_inst_info2 :
RIGHT_PARENTHESIS_T {
    inst_til_info := (others => ' ');
    inst_til_last := 1;
} |
COMMA_T inst_key_value_pairs RIGHT_PARENTHESIS_T ;

inst_key_value_pairs :
inst_til_info |
inst_key_value_pairs inst_til_info;
```

```
inst_til_info :
LEFT_BRACE_T IDENTIFIER_T TIL_INFO_T {
    declare
        i1, i2 : integer := $3'last;
        name  : name_T := (others=> ' ');
        Value_Rec : GDB_INITIAL_VALUES.Value_Record;
        Exists : boolean;
        Value : YYSType;
    begin
        -- Determine length of TIL info value
        i1 := 1;
        for i in 2..$3'last loop
           if i1 = 1 and then
             (($3(i) /= ' ') and ($3(i) /= ASCII.HT)) then
               i1 := i;
           end if;
           if $3(i) = RBRACE then
               i2 := i-1;
               exit;
           end if;
        end loop;

        -- Attempt to resolve constant value name
        if (i2+1-i1) <= name'last then
            name(1..(i2+1-i1)) := $3(i1..i2);
            GDB_DEFINITIONS.Lookup_Constant_Value(Name, Value_Rec, Exists);
        else
            Exists := false;
        end if;

        -- If constant value was resolved, then set instance value to it,
        -- otherwise use token
        if Exists then
            Value(1..integer(Value_Rec.String_Length)+2)            :=            ":"            &
Value_Rec.String_Value(1..integer(Value_Rec.String_Length)) & "}";
        else
            Value := $3;
        end if;

        -- Copy Values to instance til info string
        inst_til_info(inst_til_last) := '{';
        inst_til_last := inst_til_last + 1;
        for i in $2'range loop
            if $2(i) = ' '  then
                exit;
            end if;
            inst_til_info(inst_til_last) := $2(i);
            inst_til_last := inst_til_last + 1;
        end loop;
        for i in value'range loop
            inst_til_info(inst_til_last) := value(i);
            inst_til_last := inst_til_last + 1;
            if value(i) = RBRACE then
               exit;
            end if;
        end loop;
    exception
        when others =>
            report_error(TIL_INFO_TOO_LONG);
    end;
};

instance_value :
{
    value_definition_type := INSTANCE_VALUE_TYPE;
}
value_definition;


--*************************************************************************
--*** Grammar for handling instance initial values and shell overlay values  ***
```

```
--*****************************************************************************

value_definition :
FLOAT_T {
    declare
        last : positive;
        float_value : Types.long_float_t;
    begin
        lfio.get($1(1..identifier_size), float_value, last);
        GDB_INITIAL_VALUES.Set_Float_Value(float_value, $1(1..identifier_size));
    exception
        when others =>
            if value_definition_type = INSTANCE_VALUE_TYPE then
                Report_Error(BAD_INSTANCE_VALUE, $1(1..identifier_size));
            elsif value_definition_type = OVERLAY_VALUE_TYPE then
                Report_Error(BAD_OVERLAY_VALUE, $1(1..identifier_size));
            elsif value_definition_type = CONSTANT_VALUE_TYPE then
                Report_Error(BAD_CONSTANT_VALUE, $1(1..identifier_size));
            end if;
            GDB_INITIAL_VALUES.Clear_Last_Value;
    end;
} |
INTEGER_T {
    begin
        declare
            float_value : Types.long_float_t;
            Value : Types.Long_Integer_T;
            last : positive;
        begin
            hex_num_str := $1(1..identifier_size);
            for i in hex_num_str'range loop
                if hex_num_str(i) = ' ' then
                    hex_num_str(i..i+1) := ".0";
                    exit;
                end if;
            end loop;
            lfio.get(hex_num_str, float_value, last);
            GDB_INITIAL_VALUES.Set_Float_Value(float_value, $1(1..identifier_size));
        end;
    exception
        when others =>
            if value_definition_type = INSTANCE_VALUE_TYPE then
                Report_Error(BAD_INSTANCE_VALUE, $1(1..identifier_size));
            elsif value_definition_type = OVERLAY_VALUE_TYPE then
                Report_Error(BAD_OVERLAY_VALUE, $1(1..identifier_size));
            elsif value_definition_type = CONSTANT_VALUE_TYPE then
                Report_Error(BAD_CONSTANT_VALUE, $1(1..identifier_size));
            end if;
            GDB_INITIAL_VALUES.Clear_Last_Value;
    end;
} |
SIGNED_INTEGER_T {
    begin
        GDB_INITIAL_VALUES.Set_Int_Value(Types.long_integer_t'Value($1), $1(1..identifier_size));
    exception
        when others =>
            if value_definition_type = INSTANCE_VALUE_TYPE then
                Report_Error(BAD_INSTANCE_VALUE, $1(1..identifier_size));
            elsif value_definition_type = OVERLAY_VALUE_TYPE then
                Report_Error(BAD_OVERLAY_VALUE, $1(1..identifier_size));
            elsif value_definition_type = CONSTANT_VALUE_TYPE then
                Report_Error(BAD_CONSTANT_VALUE, $1(1..identifier_size));
            end if;
            GDB_INITIAL_VALUES.Clear_Last_Value;
    end;
} |
IDENTIFIER_T {
    declare
        Exists : Boolean;
        Val    : GDB_INITIAL_VALUES.Value_Record;
    begin
```

```
        --it is supposed to be in HEX.
        --location rule is similar but decided to duplicate
        --code since each are tied up with different ideas
        hex_num_str := $1(1..identifier_size);
        case_conversion.to_upper(hex_num_str);
        if hex_num_str(1..5) = "TRUE " then
            GDB_INITIAL_VALUES.Set_Boolean_Value(true, $1(1..identifier_size));
        elsif hex_num_str(1..6) = "FALSE " then
            GDB_INITIAL_VALUES.Set_Boolean_Value(false, $1(1..identifier_size));
        else

            -- Is this a defined constant value?
            if value_definition_type /=  CONSTANT_VALUE_TYPE then
                GDB_DEFINITIONS.Lookup_Constant_Value($1(1..identifier_size), Val, Exists);
            else
                Exists := false;
            end if;
            if Exists then
                GDB_INITIAL_VALUES.Current := Val;
            else

                --Got to get rid of the `H' in the string and
                --then add a `16#' to the front and a `#' to
                --the back so it can be converted to an integer
                --using 'Value.
                large_num_flag := false;
                for i in hex_num_str'range loop
                    if (hex_num_str(i) = 'H') or
                      (hex_num_str(i) = 'h') then -- get rid of the 'H'
                        hex_num_str(i) := '#';
                        if i > 8 then
                            if hex_num_str(i-8) = 'a' or hex_num_str(i-8) = 'A' then
                                hex_num_str(i-8) := '2';
                                large_num_flag := true;
                            elsif hex_num_str(i-8) = 'b' or hex_num_str(i-8) = 'B' then
                                hex_num_str(i-8) := '3';
                                large_num_flag := true;
                            elsif hex_num_str(i-8) = 'c' or hex_num_str(i-8) = 'C' then
                                hex_num_str(i-8) := '4';
                                large_num_flag := true;
                            elsif hex_num_str(i-8) = 'd' or hex_num_str(i-8) = 'D' then
                                hex_num_str(i-8) := '5';
                                large_num_flag := true;
                            elsif hex_num_str(i-8) = 'e' or hex_num_str(i-8) = 'E' then
                                hex_num_str(i-8) := '6';
                                large_num_flag := true;
                            elsif hex_num_str(i-8) = 'f' or hex_num_str(i-8) = 'F' then
                                hex_num_str(i-8) := '7';
                                large_num_flag := true;
                            end if;
                        end if;
                    end if;
                end loop;
                hex_num_str := "16#" & hex_num_str(1..identifier_size-3);

                --now try to convert to an integer
                declare
                    use types;
                    value : Types.Long_Integer_T;
                    float_value : Types.Long_Float_T;
                begin
                    value := Types.long_integer_T'Value(hex_num_str);
                    if large_num_flag then
                        float_value := Types.Long_Float_T(value);
                        float_value := float_value + 2147483648.0;
                        GDB_INITIAL_VALUES.Set_Float_Value(float_value, $1(1..identifier_size));
                    else
                        GDB_INITIAL_VALUES.Set_Int_Value(value, $1(1..identifier_size));
                    end if;
                exception
                    when others =>
```

```
                        --there was a problem in the conversion
                        --so set the value to 0 so GDB won't blow up.
                        if value_definition_type = INSTANCE_VALUE_TYPE then
                            Report_Error(BAD_INSTANCE_VALUE, $1(1..identifier_size));
                        elsif value_definition_type = OVERLAY_VALUE_TYPE then
                            Report_Error(BAD_OVERLAY_VALUE, $1(1..identifier_size));
                        elsif value_definition_type = CONSTANT_VALUE_TYPE then
                            Report_Error(BAD_CONSTANT_VALUE, $1(1..identifier_size));
                        end if;
                        GDB_INITIAL_VALUES.Clear_Last_Value;
                end;
            end if;
        end if;
    end;
} |
STRING_VALUE_T {
    declare
        str_value : String(1..256);
        str_len   : Long_Integer_T;
    begin
        for i in 2..$1'length loop
            if $1(i) = $1(1) then
                str_len := Long_Integer_T(i) - 2;
                exit;
            elsif i-1 <= str_value'length then
                str_value(i-1) := $1(i);
            end if;
        end loop;
        if (str_len > str_value'length) or
           (str_len = 0 and $1'length > 2) then
            str_len := str_value'length;
        end if;
        GDB_INITIAL_VALUES.Set_String_Value(Str_Value, Str_Len);
    exception
        when others =>
            if value_definition_type = INSTANCE_VALUE_TYPE then
                Report_Error(BAD_INSTANCE_VALUE, $1(1..identifier_size));
            elsif value_definition_type = OVERLAY_VALUE_TYPE then
                Report_Error(BAD_OVERLAY_VALUE, $1(1..identifier_size));
            elsif value_definition_type = CONSTANT_VALUE_TYPE then
                Report_Error(BAD_CONSTANT_VALUE, $1(1..identifier_size));
            end if;
            GDB_INITIAL_VALUES.Clear_Last_Value;
    end;
} |
HEX_INTEGER_T {
    declare
        large_num_flag : boolean;
    begin
        --it is supposed to be in HEX.
        --location rule is similar but decided to duplicate
        --code since each are tied up with different ideas
        hex_num_str := $1(1..identifier_size);
        --Got to get rid of the `H' in the string and
        --then add a `16#' to the front and a `#' to
        --the back so it can be converted to an integer
        --using 'Value.
        large_num_flag := false;
        for i in hex_num_str'range loop
            if (hex_num_str(i) = 'H') or
               (hex_num_str(i) = 'h') then -- get rid of the 'H'
                hex_num_str(i) := '#';
                if i > 8 then
                    if hex_num_str(i-8) = '8' then
                        hex_num_str(i-8) := '0';
                        large_num_flag := true;
                    elsif hex_num_str(i-8) = '9' then
                        hex_num_str(i-8) := '1';
                        large_num_flag := true;
                    end if;
                end if;
```

```
            end if;
        end loop;
        hex_num_str := "16#" & hex_num_str(1..identifier_size-3);

        --now try to convert to an integer
        declare
            use types;
            value : Types.Long_Integer_T;
            float_value : Types.Long_Float_T;
        begin
            value := Types.long_integer_T'Value(hex_num_str);
            if large_num_flag then
                float_value := Types.Long_Float_T(value);
                float_value := float_value + 2147483648.0;
                GDB_INITIAL_VALUES.Set_Float_Value(float_value, $1(1..identifier_size));
            else
                GDB_INITIAL_VALUES.Set_Int_Value(value, $1(1..identifier_size));
            end if;
        exception
            when others => --there was a problem in the conversion
                           --so set the value to 0 so GDB won't blow up.
                if value_definition_type = INSTANCE_VALUE_TYPE then
                    Report_Error(BAD_INSTANCE_VALUE, $1(1..identifier_size));
                elsif value_definition_type = OVERLAY_VALUE_TYPE then
                    Report_Error(BAD_OVERLAY_VALUE, $1(1..identifier_size));
                elsif value_definition_type = CONSTANT_VALUE_TYPE then
                    Report_Error(BAD_CONSTANT_VALUE, $1(1..identifier_size));
                end if;
                GDB_INITIAL_VALUES.Clear_Last_Value;
        end;
    end;
} |
BIN_INTEGER_T {
    declare
        large_num_flag : boolean;
    begin
        --it is supposed to be in BIN.
        --location rule is similar but decided to duplicate
        --code since each are tied up with different ideas
        hex_num_str := $1(1..identifier_size);
        --Got to get rid of the `H' in the string and
        --then add a `16#' to the front and a `#' to
        --the back so it can be converted to an integer
        --using 'Value.
        large_num_flag := false;
        for i in hex_num_str'range loop
            if (hex_num_str(i) = 'B') or
              (hex_num_str(i) = 'b') then -- get rid of the 'B'
                hex_num_str(i) := '#';
                if i > 32 then
                    if hex_num_str(i-32) = '1' then
                        hex_num_str(i-32) := '0';
                        large_num_flag := true;
                    end if;
                end if;
            end if;
        end loop;
        hex_num_str := "2#" &
          hex_num_str(1..identifier_size-2);

        --now try to convert to an integer
        declare
            use types;
            value : Types.long_integer_t;
            float_value : Types.Long_Float_T;
        begin
            value := Types.long_integer_T'Value(hex_num_str);
            if large_num_flag then
                float_value := Types.Long_Float_T(value);
                float_value := float_value + 2147483648.0;
                GDB_INITIAL_VALUES.Set_Float_Value(float_value, $1(1..identifier_size));
```

```
            else
                GDB_INITIAL_VALUES.Set_Int_Value(value, $1(1..identifier_size));
            end if;
        exception
            when others => --there was a problem in the conversion
                --so set the value to 0 so GDB won't
                --blow up.
                if value_definition_type = INSTANCE_VALUE_TYPE then
                    Report_Error(BAD_INSTANCE_VALUE, $1(1..identifier_size));
                elsif value_definition_type = OVERLAY_VALUE_TYPE then
                    Report_Error(BAD_OVERLAY_VALUE, $1(1..identifier_size));
                elsif value_definition_type = CONSTANT_VALUE_TYPE then
                    Report_Error(BAD_CONSTANT_VALUE, $1(1..identifier_size));
                end if;
                GDB_INITIAL_VALUES.Clear_Last_Value;
        end;
    end;
};

--****************************************************************************
--****************************************************************************

%%
-------------------------------------------------------------------------
-- File : gdb_parser.ada
--
-- Author : AYacc (with help from rdb)
--
-- This section defines the gdb_parser package that gets created when
-- running gdb_parser.y through AYacc.
--
-------------------------------------------------------------------------
WITH Text_IO;                   USE Text_IO;

WITH Gdb_Options;

WITH gdb_Parser_Tokens;         USE gdb_Parser_Tokens;
WITH gdb_Parser_Shift_Reduce;   USE gdb_Parser_Shift_Reduce;
WITH gdb_Parser_Goto;           USE gdb_Parser_Goto;
WITH gdb_Lexer;                 USE gdb_Lexer;

WITH gdbErr_Error_Reporting;    USE gdbErr_Error_Reporting;

WITH GDB_Timeliner_Interface_Types;
USE GDB_Timeliner_Interface_Types; --need operators

WITH GDB_FILE_SYSTEM_INTERFACE;

WITH gdbbas_Basic_Types;        USE gdbbas_Basic_Types;
WITH gdbbpt_Basic_Parser_To_Transform_Interface;
USE  gdbbpt_Basic_Parser_To_Transform_Interface;

WITH lists;

WITH unchecked_conversion;
With TL_DATA_CONVERSION;
with case_conversion;
with GDB_INITIAL_VALUES;
with GDB_PROTOCOL;
with primitive_data_types;
with GDB_DEFINITIONS;
with system;

PACKAGE gdb_Parser IS

    package types renames primitive_data_types;
    function ">"(Left, Right : Types.Long_Integer_T) return boolean renames Types.">";
    function "<"(Left, Right : Types.Long_Integer_T) return boolean renames Types."<";
    function ">"(Left, Right : Types.Long_Float_T) return boolean renames Types.">";
    function "-"(Left, Right : Types.Long_Float_T) return Types.Long_Float_T renames Types."-";
    function "-"(Left, Right : Types.Long_Integer_T) return Types.Long_Integer_T renames Types."-
```

```
";
    function "+"(Left, Right : Types.Integer_T) return Types.Integer_T renames Types."+";

   Syntax_Error : EXCEPTION;

   PROCEDURE YYParse;

   TYPE bit_t IS range 0..1;
   FOR bit_t'Size USE 1;

   TYPE word_t IS RANGE 0..65535;
   FOR word_t'Size USE 16;

   TYPE bit_size_t IS RANGE 1..16;

   TYPE bit_array IS ARRAY(bit_offset_t) OF bit_t;
   TYPE byte_array IS ARRAY(1..2) OF byte_value_t;

   function bits_to_bytes IS NEW unchecked_conversion(bit_array,byte_array);
   function bytes_to_bits IS NEW unchecked_conversion(byte_array,bit_array);
   function word_to_bits  IS NEW unchecked_conversion(word_t,bit_array);

   PACKAGE int_io IS NEW integer_io(integer);
   package lfio is new float_io(Types.long_float_t);

   ----------------------------------------
   -- TIS Specific Command Byte Offsets
   ----------------------------------------
   COMMAND_ID_BYTE_OFFSET : constant := 768;
   ROUTING_PATH_BYTE_OFFSET     : constant := 772;
   ROUTING_PROTOCOL_BYTE_OFFSET : constant := 774;
   ROUTING_PORT_BYTE_OFFSET     : constant := 776;
   ROUTING_HOST_BYTE_OFFSET     : constant := 780;
   ROUTING_SIZE_BYTE_OFFSET     : constant := 908;
   SESSION_NAME_BYTE_OFFSET     : constant := 912;

   ROUTING_HOST_LENGTH : constant := 128;
   SESSION_NAME_LENGTH  : constant := 64;

   ----------------------------------------
   -- Build Function Interface Variables
   ----------------------------------------
   RBRACE : character := '}';
   class_name      : Name_T;
   the_class_link  : gdbcli.map;
   large_num_flag : boolean;

   string_name     : Name_T;
   string_length   : User_String_Range_T;

   enum_name       : Name_T;
   the_type_link   : gdbuti.map;

   enum_id         : Name_T;
   internal_code   : User_Internal_Code_T;

   attribute_name  : Name_T;
   attribute_type  : Name_T;
   array_size      : User_Array_Range_T;
   write_flag      : Write_Flag_T;
   machine         : Machine_Type_T;

   instance_name   : Name_T;
   pui_name        : Name_T;
   last_num        : short_natural_T := 1;
   hex_num_str     : Name_T;
   instance_id     : Instance_Id_T;
   bit_offset      : Bit_Offset_T;
   attr_byte_offset     : attribute_offset_t;
   attr_byte_offset_valid : boolean;
   attr_til_info : gdbbas_Basic_Types.Protocol_Info_T := (others => ' ');
```

```
   inst_til_info : gdbbas_Basic_Types.Protocol_Info_T := (others => ' ');
   attr_til_last : integer := 1;
   inst_til_last : integer := 1;

   next_buffer_id : long_integer_t := 1;

   command_name    : Name_T;
   command_id      : Command_Id_T;
   the_command_link: gdbcmi.map;
   Command_Line    : String(1..10);

   machine_type    : Machine_Type_T;
   overlay_machine_type : Machine_Type_T;
   command_shell   : Command_Shell_T;

   word_value      : word_t;
   shell_bytes     : word_t;
   temp_word       : word_t;
   b1              : integer RANGE 1..2;
   b2              : integer RANGE 1..2;

   byte_value      : byte_Value_T;
   byte_1, byte_2  : byte_Value_T;
   byte_3, byte_4  : byte_Value_T;
   byte_5, byte_6  : byte_Value_T;
   byte_7, byte_8  : byte_Value_T;

   parameter_name  : Name_T;
   parameter_type  : Name_T;
   byte_offset     : Byte_Offset_T;
   parameter_value : Types.Long_Integer_T;
   str_param_value : String(1..256);
   str_param_len   : integer;
   default_til:                    Name_T                    :=                    "GENERAL
";

   next_cmd_offset : Types.Integer_T := 0;

   type      value_definition_type_t      is      (INSTANCE_VALUE_TYPE,      OVERLAY_VALUE_TYPE,
CONSTANT_VALUE_TYPE);
   value_definition_type : value_definition_type_t := INSTANCE_VALUE_TYPE;

   type offset_definition_type_t is (OVERLAY_OFFSET_TYPE, PARAMETER_OFFSET_TYPE);
   offset_definition_type : offset_definition_type_t := OVERLAY_OFFSET_TYPE;

   -- The following counters are used to compute addresses
   -- and offsets for instance/attribute pairs.
   instance_ctr                : long_natural_T := 0;
   --instance_location_ctr       : long_integer_T := 0;
   next_instance_loc_offset    : attribute_offset_T := 0;
   att_type_size               : short_natural_T;
   buffer_location_ctr         : long_integer_T := 2_010_000_000;

   -- The following is used to keep track of what the
   -- Parser is parsing (a command or a class) this
   -- information is needed for adding types to an object
   TYPE Parse_State is (CLASS, COMMAND);

   current_state : Parse_State;

   --- The following variables are related to command routing protocols
   type Routing_Protocol_T is (NONE, UDP);
   Routing_Protocol : Routing_Protocol_T := NONE;
   Routing_Host : String(1..ROUTING_HOST_LENGTH) := (others => ' ');
   Routing_Port : Types.Long_Integer_T := 0;
   Routing_Size : Types.Long_Integer_T := 0;
   Remote_Tis_Flag : Boolean := false;
   Session_Name : String(1..SESSION_NAME_LENGTH) := (others => ' ');

END gdb_Parser;
```

```
PACKAGE BODY gdb_Parser is

    PROCEDURE YYError(s : IN String) IS

    BEGIN

        Report_Error(PARSING_ERROR, s);
        Raise Syntax_Error;

    END YYError;

    function shift_left(word : in word_t; bits : in integer) return word_t is
    begin
        if (bits > 15) then
            return 0;
        end if;
        return word * word_t(2 ** bits);
    end shift_left;

    function shift_right(word : in word_t; bits : in integer) return word_t is
    begin
        if (bits > 15) then
            return 0;
        end if;
        return word / word_t(2 ** bits);
    end shift_right;

    --- TRIM STRING FUNCTION: TRIMS FRONT & BACK OF STRING
    function Trim(s : in string) return string is

        --- FIRST AND LAST NON-BLANK CHARACTERS
        c0 : integer := s'first;
        c1 : integer := s'last;

    begin

        --- IF STRING IS NULL OR ALL BLANKS...
        if s = (c0..c1 => ' ') then
            --- RETURN NULL STRING
            return "";

            --- OTHERWISE...
        else
            --- CHECK FORWARD FROM BEGINNING
            for i in c0..c1 loop
                if s(i) = ' ' or s(i) = ascii.ht or s(i) = ascii.cr then
                    c0 := c0 + 1;
                else
                    exit;
                end if;
            end loop;
            --- CHECK BACKWARD FROM END
            for i in reverse c0..c1 loop
                if s(i) = ' ' or s(i) = ascii.ht or s(i) = ascii.cr then
                    c1 := c1 - 1;
                else
                    exit;
                end if;
            end loop;
            --- RETURN TRIMMED STRING
            return s(c0..c1);
        end if;

    end Trim;

    --- Procedure to add a new shell overlay value
    procedure add_shell_overlay(overlay_name : in name_t;
                                overlay_type : in name_t;
                                byte_offset  : in byte_offset_t) is
        use GDB_INITIAL_VALUES;
```

```
        local_type : Primitives;
        local_offset : integer := integer(byte_offset)+1;
        overlay_size : integer;
        local_machine_type : TL_DATA_CONVERSION.machine_type_t;
        local_type_name : name_t := overlay_type;
        overlay_str_length : integer := integer(GDB_INITIAL_VALUES.Current.String_Length);
        value_ok : boolean;
        Value_Size : GDB_TYPES.Long_Integer_T;
        Quote : String(1..1) := (others => '"');
        Str_Type_Length : User_String_Range_T;
        Exists : boolean;

      --- Helper function to determine if data type is valid
      --- Note this function has side effects on local variables external in Add_Attribute
      function  Validate_Data_Type(Data_Type_Name  : Name_T) return boolean is
          Valid_Type         : Boolean;
          Resolved_Type_Name : Name_T;
      begin
            --- Look type up in primitives table
            local_type :=  Lookup_Primitive_Type (Data_Type_Name,true);

            --- Check the validity of the data type and set the size in bytes
            case local_type is
                when Of_Byte =>
                    overlay_size := 1;
                when Of_Short_Integer =>
                    overlay_size := 1;
                when Of_Integer =>
                    overlay_size := 2;
                when Of_Natural =>
                    overlay_size := 2;
                when Of_Long_Integer =>
                    overlay_size := 4;
                when Of_Long_Natural =>
                    overlay_size := 4;
                when Of_Float =>
                    overlay_size := 4;
                when Of_Long_Float =>
                    overlay_size := 8;
                when Of_Boolean =>
                    overlay_size := 1;
                when Of_String =>
                    overlay_size := overlay_str_length;
                when Of_Data_Object =>
                    return false;
                when others =>
                    Lookup_String_Type(Data_Type_Name, Str_Type_Length, Valid_Type);
                    if Valid_Type then
                        local_type := Of_String;
                        overlay_size := integer(str_type_length);
                        if (Current.Value_Type = GDB_INITIAL_VALUES.STR) and then
                           (overlay_size < overlay_str_length) then
                            Report_Error(OVERLAY_STRING_TOO_LONG,              QUOTE              &
Current.String_Value(1..overlay_str_length) & QUOTE);
                        end if;
                        GDB_INITIAL_VALUES.Current.String_Length                                  :=
Long_Integer_T(str_type_length);
                        return true;
                    else
                        --- Look up in data type definitions
                        Gdb_Definitions.Lookup_Data_Type(Data_Type_Name,       Resolved_Type_Name,
Valid_Type);
                        if Valid_Type then
                            return Validate_Data_Type(Resolved_Type_Name);
                        else
                            return false;
                        end if;
                    end if;
            end case;
            return true;
```

```
      end Validate_Data_Type;

    begin

        --- If value type is none, then an error occurred and was already reported,
        --- therefore skip this processing
        if Current.Value_Type /= none then

            --- Determine the destination machine type
            case overlay_machine_type is
                when sun =>
                    local_machine_type := tl_data_conversion.sun;
                when intel =>
                    local_machine_type := tl_data_conversion.intel;
                when vax =>
                    local_machine_type := tl_data_conversion.vax;
            end case;

            --- Determine the shell overlay data type
            case_conversion.to_upper(local_type_name);
            if not Validate_Data_Type(Local_Type_Name) then
                Report_Error(BAD_OVERLAY_DATA_TYPE, Overlay_Type);
                return;
            end if;

            --- Check for and report data type mismatch errors
            if (Local_Type = Of_String and Current.Value_Type /= GDB_INITIAL_VALUES.STR) then
                Report_Error(BAD_OVERLAY_VALUE, Current.String_Value(1..overlay_str_length));
                return;
            elsif (Local_Type = Of_Boolean and Current.Value_Type /= GDB_INITIAL_VALUES.BOOL)
then
                if Current.Value_Type = Str then
                    Report_Error(BAD_OVERLAY_VALUE,                    QUOTE                    &
Current.String_Value(1..overlay_str_length) & QUOTE);
                else
                    Report_Error(BAD_OVERLAY_VALUE, Current.String_Value(1..overlay_str_length));
                end if;
                return;
            elsif (Local_Type /= Of_String) and (Local_Type /= Of_Boolean) and
                  (Current.Value_Type /= GDB_INITIAL_VALUES.INT) and (Current.Value_Type /=
GDB_INITIAL_VALUES.FLOAT) then
                if Current.Value_Type = Str then
                    Report_Error(BAD_OVERLAY_VALUE,                    QUOTE                    &
Current.String_Value(1..overlay_str_length) & QUOTE);
                else
                    Report_Error(BAD_OVERLAY_VALUE, Current.String_Value(1..overlay_str_length));
                end if;
                return;
            end if;

            --- Check for shell overlay extending beyond the end of command shell
            Next_Cmd_Offset := Types.Integer_T(byte_offset) + Types.Integer_T(overlay_Size);
            if integer(next_cmd_offset) > command_shell'last then
                Report_Error(BAD_OVERLAY_BYTE_OFFSET, Integer'image(integer(byte_offset)));
                return;
            else
                --- Store association of shell name to next byte offset
                GDB_Definitions.Define_Shell_Overlay(Overlay_Name, Command_Name, Next_Cmd_Offset,
Exists);
                if Exists then
                    Report_Error(DUP_SHELL_OVERLAY_NAME, Overlay_Name);
                end if;

                --- Set the shell overlay value in the command shell
                begin
                    GDB_INITIAL_VALUES.Set_Initial_Value(local_type,          local_machine_type,
command_shell(local_offset)'address,
                                                         Value_Size, Value_OK);
                    if not Value_OK then
                        if Current.Value_Type = Str then
                            Report_Error(BAD_OVERLAY_VALUE,               QUOTE               &
```

```
Current.String_Value(1..overlay_str_length) & QUOTE);
                        else
                            Report_Error(BAD_OVERLAY_VALUE,
Current.String_Value(1..overlay_str_length));
                        end if;
                        return;
                    end if;
                exception

                    --- An error occured while setting the overlay value, report it
                    when others =>
                        if Current.Value_Type = Str then
                            Report_Error(BAD_OVERLAY_VALUE,                QUOTE                &
Current.String_Value(1..overlay_str_length) & QUOTE);
                        else
                            Report_Error(BAD_OVERLAY_VALUE,
Current.String_Value(1..overlay_str_length));
                        end if;
                        return;
                end;
            end if;
        end if;

    end add_shell_overlay;

##


END gdb_Parser;
```

Gdbses_input_processor_session_s

```
-- Change history:
-- 03/24/97 D.Bulpett - PCR 469 Sort by PUI name
-- 05/06/09 I.Charny  - TBD     Replaced binary trees with hash tables

with Gdbcla_Classes;
with Gdbins_Instances;
with Gdbcom_Commands;
with Gdbalt_Alternatives;

package Gdbses_Input_Processor_Session is

   package Gdbcla   renames Gdbcla_Classes;
   package Gdbins   renames Gdbins_Instances;
   package Gdbcom   renames Gdbcom_Commands;
   package Gdbalt   renames Gdbalt_Alternatives;

   -- This package contains the data types that "get to" all of the
   -- data descriptions collected by the Input Processor during a
   -- session.  It also has the procedure call to transform the
   -- in-memory data definitions to file formats and write them
   -- to disk.

   type Global_Name_Tables_T is

      record
          Class          :   Gdbcla.gdbcli.map ;
          Instance       :   Gdbins.Gdbini.map ;
          Command        :   Gdbcom.Gdbcmi.map ;
          Alternative    :   Gdbalt.Gdbaltt.map ;
      end record;

   Global_Name_Tables  :   Global_Name_Tables_T ;
      -- Anything in the Global_Name_Maps will be written to disk.

   procedure Initialize_session;

   procedure Write_Database_Files;
```

```
end Gdbses_Input_Processor_Session;
```

Gdbses_input_processor_session_b

```
-------------------------------------------------------------------------------
---
--- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--- COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
---
-------------------------------------------------------------------------------
---  Modification History:
---
---   Date      Name      PCR   Description
---   ----      ----      ---   -----------
---   06/09/99  D.DiBiaso X067  Clean up error reporting messages
---   01/22/03  D.DiBiaso X516  Added call to gdbbpt.initialize procedure
---
-------------------------------------------------------------------------------
with Gdbft_Ground_Data_File_Types;

with Gdb_File_System_Interface;

with GDBerr_Error_reporting;

with Gdbbpt_Basic_Parser_To_Transform_Interface;

With Gdb_File_System_Interface;

With GDBerr_Error_Reporting;

With Gdbbpt_Basic_Parser_To_Transform_Interface;

Package Body Gdbses_Input_Processor_Session Is



    Package Gdbft  Renames Gdbft_Ground_Data_File_Types;

    Package Gdbfsi Renames Gdb_File_System_Interface;

    Package Gdberr Renames GDBerr_Error_Reporting;

    Package Gdbbpt Renames Gdbbpt_Basic_Parser_To_Transform_Interface;

    Function Trim(s : In String) Return String;
    Function Trim(s : In String) Return String Is

    --- USE MASTER COMMON AREA
    --- FIRST AND LAST NON-BLANK CHARACTERS
        C0 : Integer := S'first;
        C1 : Integer := S'last;

    Begin

    --- DO NOTHING IF STRING IS NULL OR ALL BLANKS...
        If S = (c0..C1 => ' ') Then
            Return "";

    --- OTHERWISE...
        Else
        --- CHECK FORWARD FROM BEGINNING
            For I In C0..C1 Loop
                If S(i) = ' ' Or S(i) = Ascii.Ht Or S(i) = Ascii.Cr Then
                    C0 := C0 + 1;
                Else
                    Exit;
                End If;
            End Loop;
```

```
      --- CHECK BACKWARD FROM END
          For I In Reverse C0..C1 Loop
              If S(i) = ' ' Or S(i) = Ascii.Ht Or S(i) = Ascii.Cr Then
                  C1 := C1 - 1;
              Else
                  Exit;
              End If;
          End Loop;
      --- RETURN STRING
          Return S(c0..C1);
      End If;

  End Trim;
Procedure Initialize_Session Is

Begin

 Global_Name_Tables.Class := Gdbcla.Gdbcli.Empty_Map;

 Global_Name_Tables.Instance := Gdbins.Gdbini.Empty_Map;

 Global_Name_Tables.Command := Gdbcom.Gdbcmi.Empty_Map;

 Gdbbpt.Initialize;

End;


Procedure Write_Database_Files Is

Begin

       Gdbft.Dirini.Create (

                 File    =>    Gdbft.Instance_File,

                 Mode    =>    Gdbft.Dirini.Out_File,

                 Name    =>    Trim(Gdbfsi.Get_Instance_File_Name) );


       Gdbft.Diralt.Create (

                 File    =>    Gdbft.Alternative_File,

                 Mode    =>    Gdbft.Diralt.Out_File,

                 Name    =>    Trim(Gdbfsi.Get_Alternative_File_Name) );


     Gdbft.Dircmi.Create (

                 File    =>    Gdbft.Command_File,

                 Mode    =>    Gdbft.Dircmi.Out_File,

                 Name    =>    Trim(Gdbfsi.Get_Command_File_Name) );


     Gdbft.Dirusi.Create (

                 File    =>    Gdbft.User_Strings_File,

                 Mode    =>    Gdbft.Dirusi.Out_File,

                 Name    =>    Trim(Gdbfsi.Get_User_String_Type_File_Name) );
```

```
Gdbft.Diruei.Create (
            File    =>    Gdbft.User_Enumerations_File,
            Mode    =>    Gdbft.Diruei.Out_File,
            Name    =>    Trim(Gdbfsi.Get_User_Enum_Type_File_Name) );
Gdbft.Direni.Create(
            File    =>    Gdbft.Enumeration_Pair_File,
            Mode    =>    Gdbft.Direni.Out_File,
            Name    =>    Trim(Gdbfsi.Get_Enum_Pair_File_Name) );

Gdbft.Dircli.Create (
            File    =>    Gdbft.Class_File,
            Mode    =>    Gdbft.Dircli.Out_File,
            Name    =>    Trim(Gdbfsi.Get_Class_File_Name) );

Gdbft.Dirpri.Create (
            File    =>    Gdbft.Parameter_File,
            Mode    =>    Gdbft.Dirpri.Out_File,
            Name    =>    Trim(Gdbfsi.Get_Parameter_File_Name) );

Gdbft.Dirati.Create (
            File    =>    Gdbft.Attribute_File,
            Mode    =>    Gdbft.Dirati.Out_File,
            Name    =>    Trim(Gdbfsi.Get_Attribute_File_Name) );

 Gdbcla.Write_Table_To_File
        ( The_Table   =>    Global_Name_Tables.Class );

 Gdbins.Write_Table_To_File
        ( The_Table    =>    Global_Name_Tables.Instance );

 Gdbalt.Write_Table_To_File
        ( The_Table    =>    Global_Name_Tables.Alternative );
```

```
        Gdbcom.Write_Table_To_File

                ( The_Table     =>    Global_Name_Tables.Command );


       Gdbft.Dirini.Close ( File     =>    Gdbft.Instance_File );

       Gdbft.Diralt.Close ( File     =>    Gdbft.Alternative_File );

       Gdbft.Dircmi.Close ( File     =>    Gdbft.Command_File );

       Gdbft.Dirusi.Close ( File     =>    Gdbft.User_Strings_File );

       Gdbft.Diruei.Close ( File     =>    Gdbft.User_Enumerations_File );

       Gdbft.Direni.Close ( File     =>    Gdbft.Enumeration_Pair_File );

       Gdbft.Dircli.Close ( File     =>    Gdbft.Class_File );

       Gdbft.Dirpri.Close ( File     =>    Gdbft.Parameter_File );

       Gdbft.Dirati.Close ( File     =>    Gdbft.Attribute_File );


     Exception

       When Others =>

         Gdberr.Report_Error(gdberr.ERROR_GENERATING_DB, "", False, False);

   End Write_Database_Files;


End Gdbses_Input_Processor_Session;
```

Gdbsum_summary_files_b

```
-- ------------------------------------------------------------------------
--
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) The Charles Stark Draper Laboratory, Inc. 2009.  All Rights Reserved.
--  COPYRIGHT (C) Charles Stark Draper Laboratory 1997.  All Rights Reserved.
--  COPYRIGHT (C) Charles Stark Draper Laboratory 1996.  All Rights Reserved.
--
-- ------------------------------------------------------------------------
--
--  Package Name : GDBSUM_SUMMARY_FILES ( body )
--
--  Description  : Implements a capability for the ground database parser
--                     which outputs ASCII summary files for the compiled database
--
--  Modification History :
--
--      Date        Name       Authority    Description
--      ----        ----       ---------    -----------
--      03/13/96    JEC3337                 Created
--      05/20/96    JEC3337                 Integrated Tester_Interface code
--      05/13/97    KSL2441    TL_SOL       Use command line interface
--      03/16/98    JSM1269    TL_SOL       Modified Add_Instance_Summary_File_Record
--                                          to address new component in Instance_Record_T
--      11/23/98    DMD2273    TL_SOL       Write out object names as 80 characters instead of 32
--      12/16/98    DMD2273    TL_SOL       Cleanup formatting of command output files
--       12/21/98     DMD2273     TL_SOL        Generate instance summary file while generating
object
--                                          summary file, sorted by object, attribute name
--       05/17/99     D.DiBiaso   PCR X021      Change names of summary output files from GDB* to
TIDB*
```

```
--      06/01/99    D.DiBiaso  PCR X017     Output "BIG_ENDIAN" and "LITTLE_ENDIAN" for machine
--                                           types to command and instance summary files in stead
--                                           of "SUN" and "INTEL", respectively
--      06/09/99    D.DiBiaso  PCR X067     Clean up error reporting messages
--      06/15/99    D.DiBiaso  PCR X025     Change instance counter from short to long natural
--      06/28/99    D.DiBiaso  PCR X003     Add support for generation of instance values file
--       01/17/00     D.DiBiaso   PCR X203      Add call to generate TIL protocol object mapping
files
--      03/15/00    D.DiBiaso  PCR X205     Increase command shell size from 320 to 1024
--      04/27/00    D.DiBiaso  PCR X232     Add optional generation of TIDB command shell summary
--                                           file; Delete shell from TIDB command summary file
--            05/06/09      I.Charny   TBD                    Changed binary trees to
hash tables in
--
       Add_Instance_Summary_File_Record, Write_Command_Summary_File,
--                                                                              and
Write_Object_Summary_File
--
-- ------------------------------------------------------------------------

with Case_Conversion;
with Gdbudt_User_Data_Types;
with Gdbcom_Commands;
with Gdbprm_Parameters;
with GDB_FILE_SYSTEM_INTERFACE;
with GDB_Timeliner_Interface_Types;
with Gdbatt_Attributes;
with Gdbcla_Classes;
with Gdberr_Error_Reporting;
with Gdbins_Instances;
with Gdbses_Input_Processor_Session;
with gdb_initial_values;
with Text_Io;
with unchecked_conversion;
with GDB_PROTOCOL;
with TL_OPTIONS;

package body Gdbsum_Summary_Files is

   --- Package Renames
   package Gdbudt renames Gdbudt_User_Data_Types;
   package Gdbcom renames Gdbcom_Commands;
   package Gdbprm renames Gdbprm_Parameters;
   package Gdbatt renames Gdbatt_Attributes;
   package Gdbcla renames Gdbcla_Classes;
   package Gdberr renames Gdberr_Error_Reporting;
   package Gdbins renames Gdbins_Instances;
   package Gdbses renames Gdbses_Input_Processor_Session;
   package GDBFSI renames GDB_FILE_SYSTEM_INTERFACE;
   package TL     renames GDB_Timeliner_Interface_Types;

   --- Package use clauses
   --- Either do this or rename a bunch of operators
   use Gdbbas;

   --- Package Instances
   --- Package Variables

   type Alt_Machine_Type_T is (BIG_ENDIAN, LITTLE_ENDIAN, VAX);
   function Mach_to_alt is new unchecked_conversion(Gdbbas.Machine_Type_T, Alt_Machine_Type_T);

   --- the instance summary file name is kept here
   Instance_Summary_Filename : constant String := "TIDB_INSTANCE_SUMMARY.TXT";
   Instance_Summary_File     : Text_Io.File_Type;

   --- Internal subprograms
   function No_Spaces( Str : String) return String is
      --this function returns the first sequence of non-space characters
      --for instance no_spaces("  hello world  ") would return "hello"
      First : Natural := Str'First;
```

```
      Last  : Natural := Str'Last;

   begin
      for I in Str'range loop
         if Str(I) = ' ' then
            First := I + 1;
         else
            for J in I..Str'Last loop
               if Str(J) = ' ' then
                  Last := J-1;
                  exit;
               end if;
            end loop;
            exit;
         end if;
      end loop;
      if First > Last then --just return null string
         return "";
      else
         return Str(First..Last);
      end if;
   end No_Spaces;

       --- TRIM STRING FUNCTION: TRIMS FRONT & BACK OF STRING
       function Trim(s : in string) return string;
       function Trim(s : in string) return string is

           --- FIRST AND LAST NON-BLANK CHARACTERS
           c0 : integer := s'first;
           c1 : integer := s'last;

       begin

       --- IF STRING IS NULL OR ALL BLANKS...
           if s = (c0..c1 => ' ') then
           --- RETURN NULL STRING
               return "";

       --- OTHERWISE...
           else
           --- CHECK FORWARD FROM BEGINNING
               for i in c0..c1 loop
                   if s(i) = ' ' or s(i) = ascii.ht or s(i) = ascii.cr then
                       c0 := c0 + 1;
                   else
                       exit;
                   end if;
               end loop;
           --- CHECK BACKWARD FROM END
               for i in reverse c0..c1 loop
                   if s(i) = ' ' or s(i) = ascii.ht or s(i) = ascii.cr then
                       c1 := c1 - 1;
                   else
                       exit;
                   end if;
               end loop;
           --- RETURN TRIMMED STRING
               return s(c0..c1);
           end if;

       end Trim;

   --- External Subprograms
   --- open instance summary file
   procedure Open_Instance_Summary_File is
      --- Local Variables
      Instance_File_Header : constant String :=
        "-- TIDB_INSTANCE_SUMMARY.TXT : Summary of Instances Defined in Target System Database" &
Ascii.Lf &
        "-- Entry Legend:" & Ascii.Lf &
        "--   Object Id" & Ascii.Lf &
```

```
        "--   Attribute Byte Offset" & Ascii.Lf &
        "--   Object Name.Attribute Name" & Ascii.Lf &
        "--   Attribute Data Type" & Ascii.Lf &
        "--   Attribute Array Size" & Ascii.Lf &
        "--   Attribute Size (Bytes)" & Ascii.Lf &
        "--   Write Flag" & Ascii.Lf &
        "--   Machine Type" & Ascii.Lf &
        "--   Object Address (Real-Time Only)" & Ascii.Lf &
        "--   Bit Offset";

   begin

       --- OPEN THE SUMMARY FILE TO BE WRITTEN
           Text_IO.Create (Instance_Summary_File,
                           Text_Io.Out_File,
                           GDBFSI.Config_Path(    1    ..    GDBFSI.Config_Path_Length    )    &
Instance_Summary_Filename );

       --- WRITE HEADER TO FILE
           Text_Io.Put_Line(Instance_Summary_File,Instance_File_Header);

       --- OPEN INSTANCE VALUES FILE
           GDB_INITIAL_VALUES.Open_Instance_Values_File(GDBFSI.Config_Path(         1         ..
GDBFSI.Config_Path_Length ));

   exception
      when others => Gdberr.Report_Error(gdberr.INST_SUMMARY_ERROR,
                                         GDBFSI.Config_Path( 1 .. GDBFSI.Config_Path_Length ) &
Instance_Summary_Filename, false, false);

   end Open_Instance_Summary_File;

   --- Add instance summary file record
   procedure Add_Instance_Summary_File_Record
     (Instance_Name : in Gdbbas.Name_T;
      Class_Name    : in Gdbbas.Name_T;
      Location_ID   : in Gdbbas.Instance_ID_T;
      Bit_Offset    : in Gdbbas.Bit_Offset_T;
      Instance_Ctr  : in Gdbbas.Long_Natural_T)
    is
      --use gdbcla.gdbcli.Cursor;
      Local_Name    :  Gdbbas.Name_T := Instance_Name;
      Ins_Subtable   :  Gdbins.Gdbini.map;
      Att_Subtable : Gdbatt.gdbati.Map;
      classes_cursor : gdbcla.gdbcli.Cursor;
       Instance_Rec  :   Gdbins.Instance_Record_T :=
         (Name => Local_Name,
          Alt_Name => Local_Name,
          Location_Id => Gdbbas.Null_Instance_Id,
          Bit_Offset => Gdbbas.Default_Bit_Offset,
          Instance_Ctr => Gdbbas.Long_Natural_T'First,
           Of_Class => classes_cursor,
          Protocol_Info => (others => ' '));
       Exists   :   Boolean;
       Class_Rec        :   Gdbcla.Class_Record_T;
       Att_Rec  :   Gdbatt.Attribute_Record_T;
       -- this is used for outputting primitive type if it is a user-defined type
       Udt_Data :   Gdbudt.User_Data_Type_Record_T;
      cursor : Gdbatt.Gdbati.Cursor :=  Gdbatt.Gdbati.First(Container => Att_Subtable);
       use Gdbatt.gdbati;
    begin
       --capitalize the instance_name
       Case_Conversion.TO_UPPER(Instance_Rec.Name);
       --need to first grab the instance record from the instance tree
        Exists       :=       Gdbins.Gdbini.Contains(Gdbses.Global_Name_Tables.Instance,
Instance_Rec.Name);

      if Exists then
          Instance_Rec    :=    Gdbins.Gdbini.Element(Gdbses.Global_Name_Tables.Instance,
Instance_Rec.Name);
```

```
         --now need to grab the class record from the class tree
          Class_rec := Gdbcla.Gdbcli.Element(Instance_Rec.Of_Class);

         -- Start creating object instance values
         GDB_INITIAL_VALUES.Start_Object_Values(Class_Rec.Last_Offset);

         --then need to grab each attribute record from the attribute tree
         --in class_rec
         --while cursor /= Gdbatt.Gdbati
        while cursor /= Gdbatt.Gdbati.No_Element loop
              if Text_IO.IS_OPEN(Instance_Summary_File) then
               --need to write out correct data in correct format
               Text_IO.Put(Instance_Summary_File,
                           No_Spaces(Gdbbas.Long_Natural_T'Image(Instance_Rec.Instance_Ctr)) &
                           "," &
                           No_Spaces(Gdbbas.Attribute_Offset_T'Image(Att_Rec.Offset)) & ",");
               --we need to see if the PUI is of dot notation or just one word
               --it is assumed that if the first character of the attribute name is
               --a space then it is not dot notation
               if Att_Rec.Name(1) = ' ' then --PUI is just one word
                  Text_IO.Put(Instance_Summary_File, No_Spaces(Instance_Rec.Name));
               else --PUI should be in dot notation
                  Text_IO.Put(Instance_Summary_File, No_Spaces(Instance_Rec.Name) & "." &
                              No_Spaces(Att_Rec.Name));
               end if;
               --put the type info into it
               if (Att_Rec.Primitive_Type = Gdbbas.Undefined) then
                  -- it is a user-defined type (currently only a string or enumeration
                    Udt_Data        :=        Gdbudt.Gdbuti.Element(Position        =>
Att_Rec.User_Data_Type);
                  Text_IO.Put(Instance_Summary_File, "," & Gdbbas.Primitives'Image(

Udt_Data.Primitive_Type));
                  GDB_INITIAL_VALUES.Process_Attribute_Value(Instance_Rec.Location_Id,
                                                             Att_Rec.Name,
Udt_Data.Primitive_Type,
                                                             Att_Rec.Array_Size, Att_Rec.Offset,
                                                             Udt_Data.String_Length);
               else    -- it is a primitive type
                  Text_IO.Put(Instance_Summary_File, "," & Gdbbas.Primitives'Image(

Att_Rec.Primitive_Type));

                  -- Generate instance attribute initial value for non-string attributes
                  GDB_INITIAL_VALUES.Process_Attribute_Value(Instance_Rec.Location_Id,
                                                             Att_Rec.Name,
Att_Rec.Primitive_Type,
                                                             Att_Rec.Array_Size,
Att_Rec.Offset);

               end if;
               --now we can put the rest of the info
               Text_IO.Put_Line(Instance_Summary_File, "," &
                                No_Spaces(Gdbbas.User_Array_Range_T'Image(Att_Rec.Array_Size)) &
                                "," & No_Spaces(
                                              Gdbbas.Attribute_Offset_T'Image(Att_Rec.Size)) &
                                "," & Gdbbas.Write_Flag_T'Image(Att_Rec.Write_Flag) &
                                ","                                                          &
Alt_Machine_Type_T'Image(Mach_to_alt(Att_Rec.Machine_Type))
                                & "," & No_Spaces(

Gdbbas.Instance_ID_T'Image(Instance_Rec.Location_ID))
                                & "," &
                                No_Spaces(
                                          Gdbbas.Bit_Offset_T'Image(Instance_Rec.Bit_Offset)));
            else
               --there is some mistake because the calling proc should open and close
               --the instance_summary_file
               null;
         end if;
```

```
gdbatt.gdbati.next(cursor);
         end loop;

          -- Finish creating object instance values
          GDB_INITIAL_VALUES.Finish_Object_Values(Class_Rec.Last_Offset);


       else -- the instance does not exit
         null;
--          Gdberr.Report_Error("gdb", "Could not write instance to summary file." &
--                              "  Summary file will be incorrect.");
       end if;
    exception
       when others =>
         null;
--                  Gdberr.Report_Error("gdb", " Unknown problem in writing" &
--                                      " summary file:" & Instance_Summary_Filename);
    end Add_Instance_Summary_File_Record;


     --- Close instance summary file
    procedure Close_Instance_Summary_File is
    begin
        --- CLOSE INSTANCE VALUES FILE
        GDB_INITIAL_VALUES.Close_Instance_Values_File;

        --- CLOSE_INSTANCE SUMMARY FILE
        Text_IO.Close(Instance_Summary_File);
   exception
       when others => null;
    end Close_Instance_Summary_File;


    procedure Write_Command_Summary_File is
      use gdbcom.gdbcmi;
      use Gdbprm.gdbpri;
      --the command file name
      Summary_Com_Filename : constant String  := "TIDB_COMMAND_SUMMARY.TXT";
      Summary_Com_File     : Text_IO.FILE_TYPE;
      Shell_Com_Filename   : constant String  := "TIDB_COMMAND_SHELLS.TXT";
      Shell_Com_File       : Text_IO.FILE_TYPE;
       Command_Tree : Gdbcom.Gdbcmi.map := Gdbses.Global_Name_Tables.Command;
     Com_Rec : Gdbcom.Command_Record_T;
      Com_Map : Gdbcom.Gdbcmi.Map;
      Com_Iter: Gdbcom.Gdbcmi.Cursor := Gdbcom.Gdbcmi.first(Container => Com_Map);
      Prm_Map : Gdbprm.Gdbpri.Map;
       Prm_Rec : Gdbprm.Parameter_Record_T;
       Prm_Iter: Gdbprm.Gdbpri.Cursor := Gdbprm.Gdbpri.first(Container => Prm_Map);
      -- this is used for outputting primitive type if it is a user-defined type
      Udt_Data: Gdbudt.User_Data_Type_Record_T;
      --- File Headers
      Command_File_Header : constant String :=
        "-- TIDB_COMMAND_SUMMARY.TXT : Summary of Commands Defined in Target System Database" &
        Ascii.Lf &
        "-- Entry Legend:" & Ascii.Lf &
        "--   Command Name : <Command_name>" & Ascii.Lf &
        "--   Target Machine Type : <Machine_Type>" & Ascii.Lf &
        "--   Parameters :" & Ascii.Lf &
        "--       Parameter Name : <Parameter_Name>" & Ascii.Lf &
        "--           Parameter Type : <Parameter_Type>" & Ascii.Lf &
        "--           Parameter Array Size : <Num_Array_Elements>" & Ascii.Lf &
        "--           Parameter Total Size (Bytes) : <Byte_Size>" & Ascii.Lf &
        "--           Parameter Offset (Bytes) : <Byte_Offset>" & Ascii.Lf &
        "--           Parameter Offset (Bits) : <Bits>" & Ascii.Lf &
        "--       End Parameter" & Ascii.Lf &
        "--   End Command";
      Command_Shell_File_Header : constant String :=
        "--  TIDB_COMMAND_SHELLS.TXT  :  Summary  of  Command  Shells  Defined  in  Target  System
Database" &
        Ascii.Lf &
```

```
         "-- Entry Legend:" & Ascii.Lf &
         "--   Command Name : <Command_name>" & Ascii.Lf &
         "--   Command Shell :" & Ascii.lf &
         "--       Shell(0..1023) : <Shell_Values>" & Ascii.Lf &
         "--   End Command";

      -- Parameters from TL_Options
       generate_command_shells : boolean := false;
       count : integer;
   begin

      begin

           Text_IO.Create( Summary_Com_File,
                           TEXT_IO.Out_File,
                           GDBFSI.Config_Path(    1    ..    GDBFSI.Config_Path_Length    )    &
Summary_Com_Filename );

      exception
         when others => Gdberr.Report_Error(gdberr.CMD_SUMMARY_ERROR,
                                            GDBFSI.Config_Path( 1 .. GDBFSI.Config_Path_Length )
& Summary_Com_Filename, false, false);
         return;
      end;

      --- Create command shells file, if requested
      TL_Options.Determine_Flag_Option("generate_command_shells",       generate_command_shells,
count);
      if generate_command_shells then
          begin

           Text_IO.Create( Shell_Com_File,
                           TEXT_IO.Out_File,
                           GDBFSI.Config_Path(    1    ..    GDBFSI.Config_Path_Length    )    &
Shell_Com_Filename );

          exception
             when others => Gdberr.Report_Error(gdberr.CMD_SUMMARY_ERROR,
                                            GDBFSI.Config_Path( 1 .. GDBFSI.Config_Path_Length )
& Shell_Com_Filename, false, false);
             generate_command_shells := false;
          end;

          --- Output File Header
          Text_Io.Put_Line(Shell_Com_File, Command_Shell_File_Header);

      else
          --- Delete the command shells file if it exists
          begin
              Text_IO.Open(Shell_Com_File,
                           TEXT_IO.Out_File,
                           GDBFSI.Config_Path(    1    ..    GDBFSI.Config_Path_Length    )    &
Shell_Com_Filename );
              Text_IO.Delete(Shell_Com_File);
          exception
              when others => null;
          end;
      end if;

      --- Output File Header
      Text_Io.Put_Line(Summary_Com_File, Command_File_Header);

      --run through all of the commands one at a time
      --for each command, write the parameters associated with that command
      --and put a delimiter to denote a new command
      --put command name, id, machine type followed by each parameter name,
      --basic type, arrayness, size, offset, and bit offset.

      --need to grab each command record from the command tree
      while Com_Iter /= gdbcom.gdbcmi.No_Element loop --get all commands one at a time
          com_rec := gdbcom.gdbcmi.Element(Position => com_iter);
```

```
        --the file should be opened by calling procedure.
          --but need to make sure file is open
          if Text_IO.IS_OPEN(Summary_Com_File) then
             --write the commands & parameters
             Text_Io.Put(Summary_Com_File, "Command Name : ");
             Text_IO.Put_Line(Summary_Com_File,Trim(Com_Rec.Name));
             --- Now write the shell contents
             if generate_command_shells then
                 Text_Io.Put(Shell_Com_File, "Command Name : ");
                 Text_IO.Put_Line(Shell_Com_File,Trim(Com_Rec.Name));
                 declare
                    package Byte_Io is new Text_Io.Integer_Io(Tl.Byte_T);
                    package Int_Io is new Text_Io.Integer_Io(Integer);
                    Bytes_Per_Line : constant := 8;
                    Current_Byte   : Integer;
                    Max_Lines : Integer := TL.CCSDS_Command_Shell_Type'Length/
                    Bytes_Per_Line;
                    Temp_Str : String(1..7);
                 begin
                    Text_Io.Put_Line(Shell_Com_File, "Command Shell :");
                    for Lines in 1..Max_Lines loop
                       Text_Io.Put(Shell_Com_File,"    Shell(");
                       Int_Io.Put(Shell_Com_File,
                                  ((Lines-1) * Bytes_Per_Line), Width =>4);
                       Text_Io.Put(Shell_Com_File, "..");
                       Int_Io.Put(Shell_Com_File,
                                  (Lines * Bytes_Per_Line) - 1, Width=>4);
                       Text_Io.Put(Shell_Com_File, "):");
                       for Byte in 1..Bytes_Per_Line loop
                          Current_Byte := ((Lines-1) * Bytes_Per_Line) + Byte;
                          Byte_Io.Put(Temp_Str,
                                      Com_Rec.Command_Shell(Current_Byte),
                                      Base=>16);
                          if Temp_Str(5) = '#' then
                              Temp_Str(2..5) := "16#0";
                          end if;
                          Text_Io.Put(Shell_Com_File, Temp_Str);
                       end loop;
                       Text_Io.New_Line(Shell_Com_File);
                    end loop;
                 end;
                 Text_IO.Put_Line(Shell_Com_File, "End Command");
             end if;

             --put machine type
             Text_Io.Put(Summary_Com_File, "Target Machine Type : ");
             Text_IO.Put_Line(Summary_Com_File,
                              Alt_Machine_Type_T'Image(Mach_to_Alt(Com_Rec.Machine_Type)));
             --need to grab each parameter record from the parameter tree
             --in com_rec
             Text_Io.Put_Line(Summary_Com_File, "Parameters :");
             while Prm_Iter /= gdbprm.gdbpri.No_Element loop --get parameters one at a
time
               prm_rec := gdbprm.gdbpri.Element(Position => prm_iter);
                --write the name for the parameter
                Text_Io.Put(Summary_Com_File, "    Parameter Name : ");
                Text_IO.Put_Line(Summary_Com_File, Trim(Prm_Rec.Name));
                --write the type
                Text_Io.Put(Summary_Com_File, "        Parameter Type : ");
                if (Prm_Rec.Primitive_Type = Gdbbas.Undefined) then
                   -- it is a user-defined type(currently only a string or enumeration)
                  Udt_Data := Gdbudt.Gdbuti.Element(Position => prm_rec.User_Data_Type);
                   Text_IO.Put_Line(Summary_Com_File, Gdbbas.Primitives'Image(

Udt_Data.Primitive_Type));
                else    -- it is a primitive type
                   Text_IO.Put_Line(Summary_Com_File, Gdbbas.Primitives'Image(
                     Prm_Rec.Primitive_Type));
                end if;
                --write the arrayness of the parameter
```

```
                Text_Io.Put(Summary_Com_File, "        Parameter Array Size : ");
                Text_IO.Put_Line(Summary_Com_File,
                                 No_Spaces(Gdbbas.User_Array_Range_T'Image(Prm_Rec.Array_Size)));
                --write the size of the parameter
                Text_Io.Put(Summary_Com_File, "        Parameter Total Size (Bytes) : ");
                Text_IO.Put_Line(Summary_Com_File,
                                 No_Spaces(Gdbbas.Parameter_Offset_T'Image(Prm_Rec.Size)));
                --write the offset of the parameter
                Text_Io.Put(Summary_Com_File, "        Parameter Offset (Bytes) : ");
                Text_IO.Put_Line(Summary_Com_File, No_Spaces(

Gdbbas.Byte_Offset_T'Image(Prm_Rec.Byte_Offset)));
                --write the bit offset of the parameter
                Text_Io.Put(Summary_Com_File, "        Parameter Offset (Bits) : ");
                Text_IO.Put_Line(Summary_Com_File, No_Spaces(
                  Gdbbas.Bit_Offset_T'Image(Prm_Rec.Bit_Offset)));

               Text_Io.Put_Line(Summary_Com_File, "    End Parameter");
                 gdbprm.gdbpri.next(Position => prm_iter);
             end loop;

             --now must delimit file so reader knows finished with current command
             Text_IO.Put_Line(Summary_Com_File, "End Command");

             --error check
          else
               null;
--             Gdberr.Report_Error("gdb", Summary_Com_Filename & " is not open." &
--                                 "  Therefore, nothing will get written to it.");
         end if;
                   Gdbcom.Gdbcmi.Next(Position => Com_Iter);

       end loop;

       begin
          Text_IO.CLOSE(Summary_Com_File);
          if generate_command_shells then
              Text_IO.CLOSE(Shell_Com_File);
          end if;
       exception
          when others => null;
       end;

    end Write_Command_Summary_File;

    --- Procedure to write instance summary
    procedure Write_Object_Summary_File is
       --- Object Summary File Variables
       Object_Filename   : constant String := "TIDB_OBJECT_SUMMARY.TXT";
       Object_File       : Text_IO.FILE_TYPE;
       Object_File_Header : constant String :=
         "-- TIDB_OBJECT_SUMMARY.TXT : Summary of Objects Defined in Target System Database" &
Ascii.Lf &
         "-- Object Name                                                                    ID
Size(Bytes)" & Ascii.Lf ;
       --- Top of Instance Tree
        Object_Map        : Gdbins.Gdbini.Map := Gdbses.Global_Name_Tables.Instance;
        --- Object Record
       Object_Record     : Gdbins.Instance_Record_T;
       --- Object Iterator
       Object_Iterator    : Gdbins.Gdbini.cursor := Gdbins.Gdbini.First(Object_Map);
       --- Class Record
      Class_Record        : Gdbcla.Class_Record_T;
      use Gdbins.Gdbini;
    begin

       --- CREATE THE OBJECT SUMMARY FILE - EXCEPTIONS PROPAGATED TO CALLER
       Text_IO.Create ( Object_File,
                        TEXT_IO.Out_File,
                        GDBFSI.Config_Path( 1 .. GDBFSI.Config_Path_Length ) & Object_Filename );
```

```
      --- CREATE THE INSTANCE SUMMARY FILE
      Open_Instance_Summary_File;

      --- Output Object File Header
      Text_Io.Put(Object_File, Object_File_Header);

      --- Walk the Object Tree to get all object records.  For each record,
      --- write summary information to the object summary file.
      while Object_Iterator /= Gdbins.Gdbini.No_Element loop
         --- Write out the name to file
         Text_IO.Put(Object_File, Object_Record.Name);
         --- Write PUI ID to file
         Text_Io.Put(Object_File,
                     Gdbbas.Long_Natural_T'Image(Object_Record.Instance_Ctr));
         --- Now, grab the class record of which this PUI is an object.
         Class_Record := Gdbcla.Gdbcli.Element(Object_Record.Of_Class);
         --- Write size of object from class record (in bytes) to file
         Text_Io.Put(Object_File,
                     Gdbbas.Attribute_Offset_T'Image(Class_Record.Last_Offset));
         --- End line of text
         Text_Io.New_Line(Object_File);

         --- Write the object to the instance summary file
         Add_Instance_Summary_File_Record(Object_Record.Name, Class_Record.Name,
                                          Object_Record.Location_Id, Object_Record.Bit_Offset,
                                          Object_Record.Instance_Ctr);

        Gdbins.Gdbini.Next(Position => Object_Iterator);
      end loop;

      --- CLOSE THE INSTANCE SUMMARY FILE
      Close_Instance_Summary_File;

      --- Close Object File
      Text_Io.Close(Object_File);

      --- Write protocol mapping summary files
      GDB_PROTOCOL.Write_Protocol_Mapping_Files;

      exception
         when others=>
                    Gdberr.Report_Error(gdberr.OBJ_SUMMARY_ERROR,
                                        GDBFSI.Config_Path(  1  ..  GDBFSI.Config_Path_Length  )  &
Object_Filename, false, false);
    end Write_Object_Summary_File;

    end Gdbsum_Summary_Files;
```