CSR	MASSACHUSETTS INSTITUTE OF TECHNOLOGY CENTER FOR SPACE RESEARCH CAMBRIDGE, MASSACHUSETTS 02139

REVISION LOG	TITLE: Huffman Map	DOC. NO. 36-53244

Revision	Date (mm/dd/yy)	ECO No.	Page(s) Affected	Reason	Approval
A	05/29/96	36-652	ALL	Initial Code & Documation	[illegible] 6/4/96

4.0 Huffman Map (36-53244 A)

4.1 Purpose

The Huffman Map is used to provide access to the Huffman Tables.

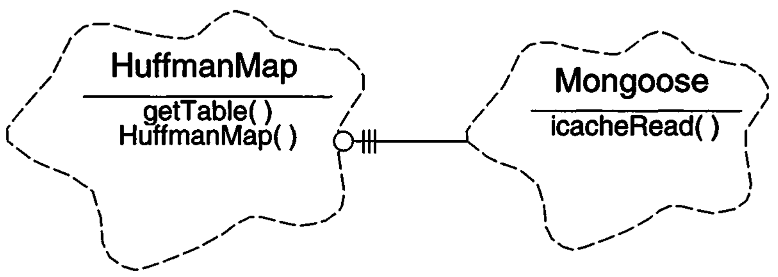
4.2 Uses

Use 1:: The Huffman Map provides a specific HuffmanTable address in I-cache.

4.3 Organization

Figure 1 illustrates the relationship between the classes used by the Huffman Map.

FIGURE 1. Huffman Map Class Relationships



The HuffmanMap is a member of the ***Science*** class category.

HuffmanMap - this function is responsible for the delivery of the I-Cache address of a requested Huffman Table.

Mongoose - This class provides a means of extracting data from I-Cache. It is a subclass of ***Executive***.

4.4 Huffman Map

While on orbit the characteristics of each CCD is expected to vary slowly through time. The optimum size of truncated tables may vary. There may be a desire for unique tables for sets of CCDs or for individual CCDs. Modification of the tables is expected to be infrequent. The use of the table reference scheme, described below, is intended to maximize flexibility when the tables are revised.

The Huffman tables are stored in I-Cache. When the `HuffmanMap` constructor instance is created, it is passed the I-Cache location of the *`huffTableList`*, this structure contains the array *offset[32]* which **HuffmanMap**::`getTable` accesses. The *huffTableList* structure precedes the set of Huffman tables and contains an array of slots that specify the offset from the end of the list structure to the header of each Huffman table. Inactive slots contain an impossible offset value. The Huffman table header contains its length, thus it defines itself. A full table with header occupies 8193 thirty-two bit words. Truncated tables may contain fewer than 100 entries.

The storage of both the tables and their reference in I-Cache anticipates survival through CPU resets even when the tables have been modified using write memory commands.

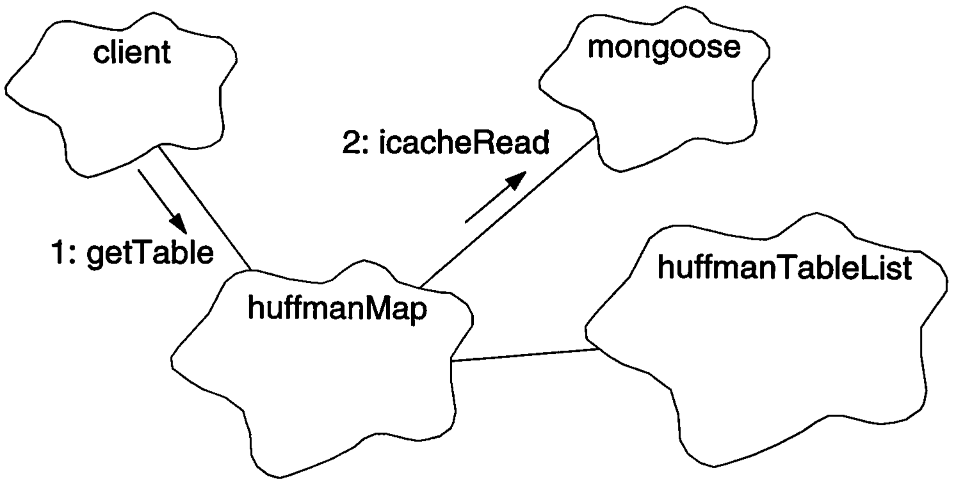
4.5 Scenario

The constructor having been called with the location of the Huffman Map, the client may use `getTable` specifying a slot number to obtain the address of the desired HuffmanTable which will be loaded into D-Cache. Reference to a nonexistent or inactive table element will return zero; consequently the data will be packed without compression.

4.6 Use 1: Obtaining a specific Huffman Table address in I-cache

Figure 2 illustrates the Huffman Map procedure to obtain a Huffman table address.

FIGURE 2. HuffmanMap referencing a HuffmanTable address



1. When the client function needs to load a HuffmanTable into D-Cache, it uses ***HuffmanMap***::`getTable` passing it a slot number in order to acquire the address of that table.
2. `getTable` will reference the I-Cache storage structure *tableBase*, pointing to the HuffmanMap structure *HuffTableList*. Then use `icacheRead` to obtain the offset stored in the array *offset*. With a legitimate offset value, it will provide the I-Cache memory address, otherwise it will return zero.

4.7 Class HuffmanMap

Documentation

This class represents the collection of compression tables contained within I-cache. It is responsible for mapping compression table codes into the corresponding I-cache address containing the table.

Export Control:	Public
Cardinality:	1

Hierarchy

Superclasses:	**`none`**

Public Interface

Operations:	HuffmanMap().
	getTable().

Private Interface:

Has-A Relationship:

`const HuffTableList` * *tableBase*: Structure containing the *offset* array.

`unsigned` *offset[32]*: The array of offsets to each Huffman Table. The offset origin is the first word after this structure. An offset value of 0xFFFFFFFF indicates an empty table entry.

Concurrency:	Synchronous
Persistence:	Persistent

4.7.1 HuffmanMap()

Public member of:	**`HuffmanMap`**
Return Class:	**`void`**
Arguments:	**`const HuffTableList`** `*` *tableBase*

Documentation:

This is the constructor for the Huffman compression table map. *tableBase* is the starting address of the structure containing an array of offsets for each Huffman compression table contained within I-cache. This structure shall precede the HuffmanTable storage.

Preconditions

The HuffTableList and set of associated Huffman Tables exist in I-Cache.

Concurrency: Sequential

4.7.2 getTable()

Public member of: **`HuffmanMap`**

Return Class: **`const unsigned *`**

Arguments:

`unsigned` *tableSlot*

Documentation

This function retrieves the I-cache address of the compression table indicated by *tableSlot*. If no corresponding table exists, or if *tableSlot* contains 0xFFFFFFFF, the function returns 0.

Concurrency: Synchronous