CSR	MASSACHUSETTS INSTITUTE OF TECHNOLOGY CENTER FOR SPACE RESEARCH CAMBRIDGE, MASSACHUSETTS 02139	
REVISION LOG	TITLE: Gram Library	DOC. NO. 36-53241

Revision	Date (mm/dd/yy)	ECO No.	Page(s) Affected	Reason	Approval
A	02/29/96	36.521	ALL	Initial version	[illegible] 4/11/96

4.0 Sram Library (36-53241 A)

4.1 Purpose

The SRAM Library provides a set of primitives which direct CCD pixel charge manipulation in support of specific predetermined observational goals.

4.2 Uses

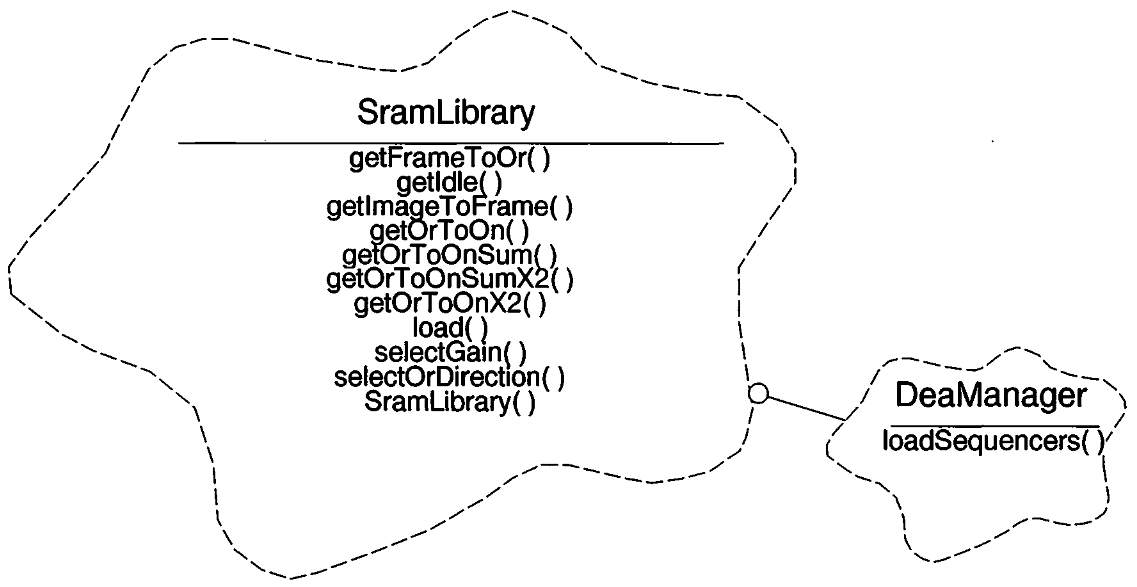
The following lists the uses of the SRAM Library.

Use 1:: Facilitates loading of the SRAM Library by the DEA Manager
Use 2:: Accepts and retains a set of decision directives
Use 3:: Directs access to one or more blocks containing a primitive command set

4.3 Organization

Figure 1 illustrates the relationship used by the Sram Library.

FIGURE 1. Sram Library Relationships



The Sram Library class provides accesses to a catalog of primatives. The primitives provide a set of directives which are implemented by the CCD controller to initiate each step of each action of a series of exposures which make up an observation. These are the directives which cause the shifting of CCD pixel charge out of the CCD for processing. This class uses the services of the DEA Manager to load the library.

4.4 SRAM Load Table Format

This section illustrates the format of an SRAM library table, used to load SRAM and determine where the various SRAM primitives reside once loaded.

```
struct SramEntry
{
    unsigned  index;                          // Location
    unsigned  count;                          // Blocks
};

struct SramHeader // Catalog of entries     accessor function
{
    struct SramEntry idle;                    // getIdle

    struct SramEntry imageFrame;              // getImageToFrame
    struct SramEntry frameOReg;               // get FrameToOr

    struct SramEntry oRegONode;               // getOrToOn
    struct SramEntry oRegONode_High;          // getOrToOn
    struct SramEntry oRegONodeX2;             // getOrToOnX2
    struct SramEntry oRegONodeX2_High;        // getOrToOnX2
    struct SramEntry oRegONode_Rev;           // getOrToOn
    struct SramEntry oRegONode_High_Rev;      // getOrToOn
    struct SramEntry oRegONodeX2_Rev;         // getOrToOnX2
    struct SramEntry oRegONodeX2_High_Rev;    // getOrToOnX2

    struct SramEntry oRegONode_Dis;           // getOrToOn
    struct SramEntry oRegONodeX2_Dis;         // getOrToOnX2
    struct SramEntry oRegONode_Dis_Rev;       // getOrToOn
    struct SramEntry oRegONodeX2_Dis_Rev;     // getOrToOnX2

    struct SramEntry oRegONodeSum;            // getOrToOnSum
    struct SramEntry oRegONodeSum_High;       // getOrToOnSum
    struct SramEntry oRegONodeSum_Rev;        // getOrToOnSum
    struct SramEntry oRegONodeSum_High_Rev;   // getOrToOnSum

    struct SramEntry oRegONodeSumX2;          // getOrToOnSumX2
    struct SramEntry oRegONodeSumX2_Rev;      // getOrToOnSumX2
};

struct SramTable
{
    struct SramHeader sramHeader;
    struct DeaSequenceLoad seqLdBlock;        // from cmdif.H
};
```

4.5 Scenario

The library itself consists of a catalog of the CCD control primitives and the primitive sets themselves. When setting up an observation, the selected Sram Library address is provided to the constructor, `SramLibrary()`. The catalog structure delineates the location of each of the primatives. Each primitive is contained in one or more contiguous data blocks. The library functions, in conjunction with the decision directives, are used in determining the location and length of the primitives which will control each step of an observation in a science run.

Use 1:

The client uses `load()` which directs the **`DeaManager`**`.loadSequencers()` to load the sets of primitives into preselected memory locations. The catalog, having been properly constructed to indicate the location of the loaded primitives, will be called upon to direct the client to the desired primitive for sequencing of the CCDs.

Use 2:

The two functions, `selectGain()` and `selectOrDirection()` provide boolean decision indicators which enable four functions to select and deliver each of eighteen primitives. The default settings are: low gain and forward direction.

Use 3:

The function names are indicative of the primitive they provide. Or and On are output register and output node. X2 indicates a shift of charge from two pixels (default is one pixel). Sum denotes a accumulation of charge at the node. Image to Frame and Frame to Output Register refer to transfer of a row of pixel charge. Idle provides for CCD exposure time.

The client will access the catalog functions to obtain the location and block count of the primitives to be invoked in controlling the CCD output.

4.6 Class SramLibrary

Documentation:

This class supplies a collection of SRAM blocks which perform specific operations. The member functions of this class return the index and block count of an SRAM primitive which performs a particular operation.

Export Control:	Public
Cardinality:	n

Hierarchy:

Superclasses:	**none**

Implementation Uses:

DeaManager

Public Interface:

Operations:

```
SramLibrary()
getFrameToOr()
getIdle()
getImageToFrame()
getOrToOn()
getOrToOnSum()
getOrToOnSumX2()
getOrToOnX2()
load()
selectGain()
selectOrDirection()
```

Private Interface:

Has-A Relationships:

Boolean *useHighGain*: This specifies whether or not to use high-gain OR to ON clocking. If BoolFalse, use the 1:1 gain selection. If BoolTrue, use the 4:1 gain selection.

Boolean *useReverse*: This specifies the direction in which to clock the output registers. If BoolFalse, clock in the normal clocking direction (i.e. Full or AC). If BoolTrue, clock in the reverse direction (BD or Diagnostic).

const SramTable* *catalog*: This points to the table passed as the argument to the instance's constructor.

Concurrency: Sequential

Persistence: Transient

4.6.1 SramLibrary()

Public member of: **SramLibrary**

Arguments:

SramTable* *catalogAddr*

Documentation:

SramLibrary This constructor initializes the SramLibrary instance, where *catalogAddr* points to the library table to use.

Concurrency: Sequential

4.6.2 getFrameToOr()

Public member of: **SramLibrary**

Return Class: **void**

Arguments:

unsigned& *index*
unsigned& *count*

Documentation:

This function supplies a set of adjacent SRAM blocks which clock one row from the framestore into the output registers. Upon return, *index* contain the index of the first SRAM block, and *count* returns the number of adjacent blocks to use.

Concurrency: Guarded

4.6.3 getIdle()

Public member of:	`SramLibrary`
Return Class:	`void`
Arguments:	`unsigned&` *index*

Documentation:

This function returns the *index* of an SRAM block which is used to integrate the CCDs.

Concurrency: Guarded

4.6.4 getImageToFrame()

Public member of:	`SramLibrary`
Return Class:	`void`
Arguments:	`unsigned&` *index* `unsigned&` *count*

Documentation:

getImageToFrame will get a set of adjacent SRAM blocks which transfer one row from the image array to the framestore. On return, *index* contains the SRAM index to the first block, and *count* returns the number of adjacent blocks to use.

Concurrency: Guarded

4.6.5 getOrToOn()

Public member of:	`SramLibrary`
Return Class:	`void`
Arguments:	`Boolean` *discard* `unsigned&` *index*

Documentation:

getOrToOn will get an SRAM block which transfers one pixel from the output register to the output node. If *discard* is BoolFalse, the caller intends to sample the pixels at the output node, and if *discard* is BoolTrue, the caller intends to discard the pixel. Upon return, *index* will contain the SRAM index of the block. The supplied block will clock the output registers in the direction indicated by the most recent call to `selectDirection()` and with the gain specified by the most recent call to `selectGain()`.

Concurrency: Guarded

4.6.6 getOrToOnSum()

Public member of:	`SramLibrary`
Return Class:	`void`
Arguments:	`unsigned&` *index*

Documentation:

This function supplies an SRAM block which clocks and sums one pixel from the output register into the output node. Upon return, *index* contains the selected SRAM block. The generated block will use the gain and direction last selected by `selectGain()` and `selectOrDirection()`, respectively.

Concurrency: Guarded

4.6.7 getOrToOnSumX2()

Public member of:	`SramLibrary`
Return Class:	`void`
Arguments:	
	`unsigned&` *`index`*

Documentation:

This function returns an SRAM block which clocks and sums two output register pixels into the output node. Upon returning, *`index`* contains the SRAM address of the desired block.

Concurrency: Guarded

4.6.8 getOrToOnX2()

Public member of:	`SramLibrary`
Return Class:	`void`
Arguments:	
	`Boolean` *`discard`*
	`unsigned&` *`index`*

Documentation:

This function supplies an SRAM block which clocks two pixels from the output register to the output node (possibly in one SRAM cycle). *`discard`* indicates that the caller wants the clocked pixels to be discarded (either at the output node, or via the FEP). Upon return, *`index`* contains the SRAM index of the desired block. The supplied block will clock in the direction last specified using `selectDirection`(), and is timed to generate the gain last chosen using `sellectGain`().

Concurrency: Guarded

4.6.9 load()

Public member of:	`SramLibrary`
Return Class:	`Boolean`

Documentation:

This function loads the contents of the SRAM library into the DEA controllers selected by the library's load block, using the DEA Manager. If the load succeeds the function returns BoolTrue, otherwise, it returns BoolFalse.

Concurrency: Guarded

4.6.10 selectGain()

Public member of:	`SramLibrary`
Return Class:	`void`
Arguments:	
	`Boolean` *high*

Documentation:

The selectGain function selects whether or not to use SRAM primitives which return high gain from the output registers. If high is BoolFalse, then use primitives which supply a gain of 1:1. If high is BoolTrue, return use blocks which supply a gain of 4:1 at the output node (sacrificing pulse-height resolution).

Concurrency: Guarded

4.6.11 selectOrDirection()

Public member of:	`SramLibrary`
Return Class:	`void`
Arguments:	`Boolean` *`reverse`*

Documentation:

The `selectOrDirection` function selects the clocking direction of the output registers. If *`reverse`* is BoolFalse, then clock each register half toward their respective output nodes (i.e. use for Full and AC modes). If *`reverse`* is BoolTrue, clock each register in the opposite direction (i.e. use for BD and Diagnostic modes).

Concurrency:	Guarded