CSR

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

REVISION LOG

TITLE: Software Detailed Design
DEA Device

DOC. NO. 36-53211

Revision	Date (mm/dd/yy)	ECO No.	Page(s) Affected	Reason	Approval
—	3/28/95	—	—	Initial version for design walkthrough	
A	5/4/95	36-234	all	Incorporate review comments	5/22/95

11.0 Detector Electronics Assembly Device (36-53211 A)

11.1 Purpose

The purpose of the Detector Electronics Assembly (DEA) Device is to provide access to Back End's DEA interface logic and registers.

NOTE: The responsibility for forming command words, and interpreting reply words is lies with the ***Protocols*** layer, specifically, by classes operating under the direction of the **DeaManager**. See Section TBD.

11.2 Uses

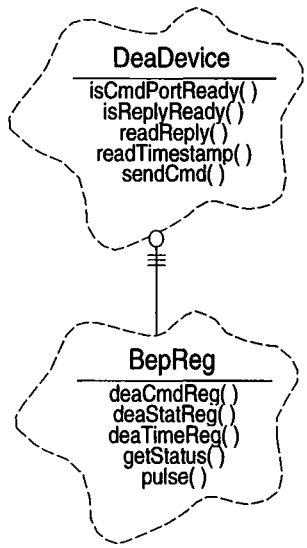
The DEA Device class provides the following features:

Use 1:: Send a command to the DEA
Use 2:: Read a response from the DEA
Use 3:: Read the timestamp latched by the previous command sent to the DEA

11.3 Organization

Figure 24 illustrates the relationships used by the **DeaDevice** class.

FIGURE 24. DEA Device Class Relationships



DeaDevice- This class is responsible for providing access to the Back End Processor's Detector Electronics Assembly command and reply port hardware, and the microsecond science timestamp. This class provides functions to test the status of the command and reply ports (`isCmdPortReady()`, `isReplyPortReady()`), write commands to and read replies from the DEA (`sendCmd()`, `readReply()`), and read the microsecond

timestamp value, latched when a command is sent to the DEA (`readTimestamp()`). This class uses the **`BepReg`** class to get access to the DEA interface register addresses (`deaCmdReg()`, `deaStatReg()`, `deaTimeReg()`), to read the command port and reply port status (`getStatus()`), and to reset the reply port ready bit (`pulse()`).

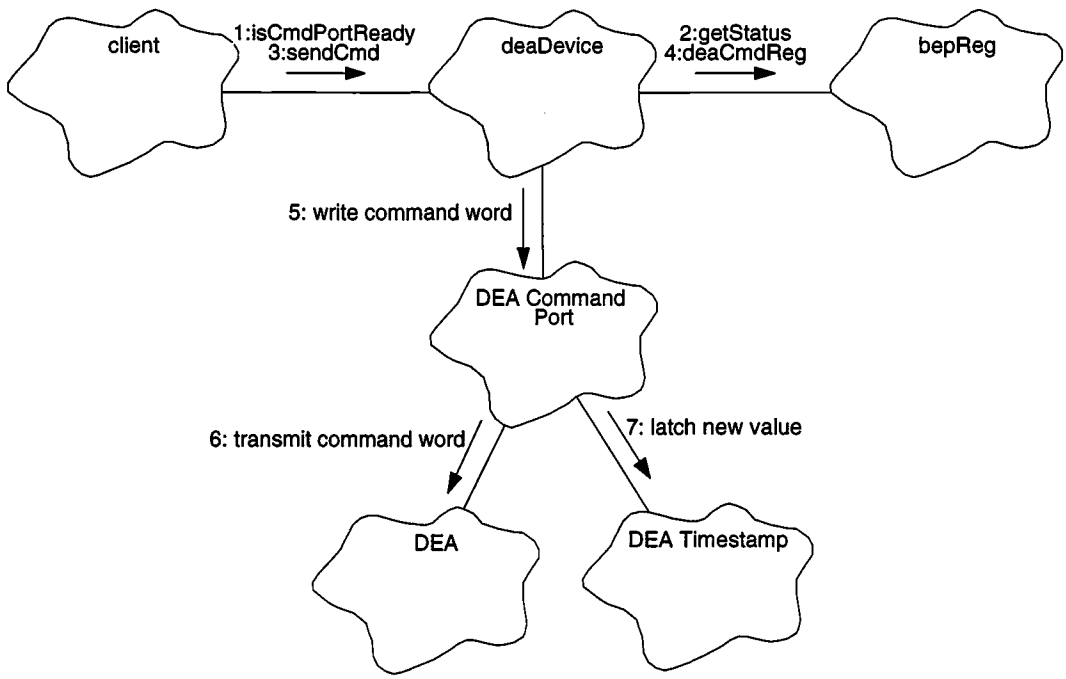
`BepReg`- This class represents the lowest level hardware access to the features provided by the Back End hardware control, status, and pulse registers. For more detail, see Section 5.0 .

11.4 Scenarios

11.4.1 Use 1: Send a command to the DEA

Figure 25 illustrates the steps used to send a command to the DEA.

FIGURE 25. Send command to DEA



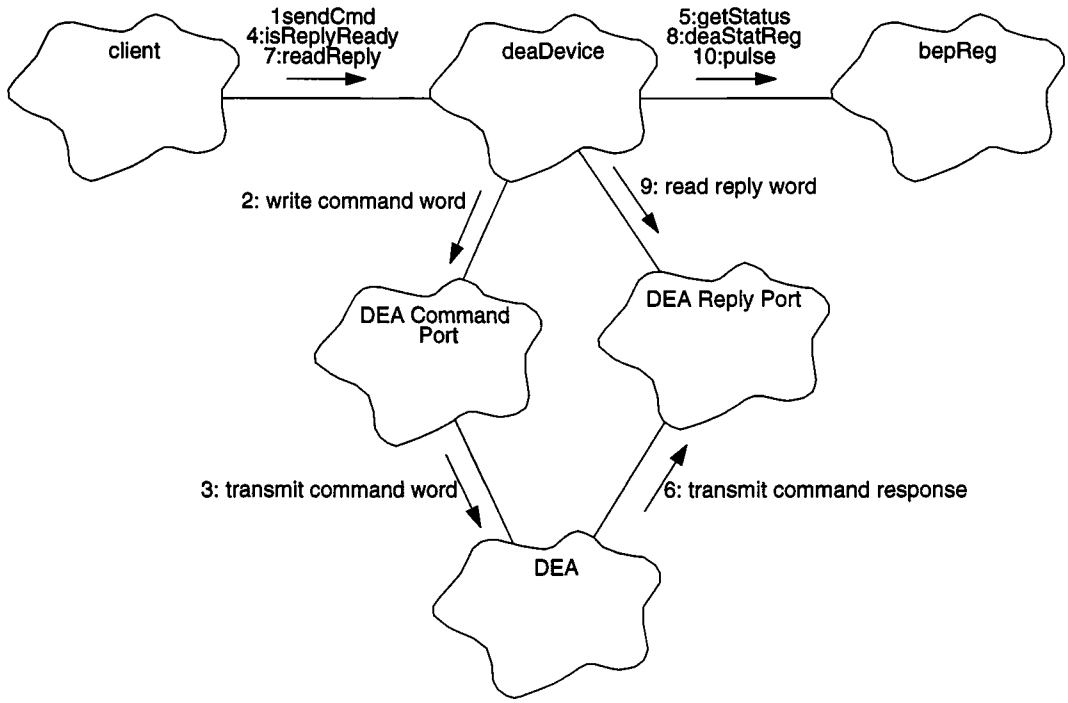
1. *`client`* polls the DEA Device until the command port becomes available, using *`deaDevice`*`.isCmdPortReady()`. NOTE: It is *`client`*'s responsibility to detect and handle time-outs on the DEA interface.
2. `isCmdPortReady()` uses *`bepReg`*`.getStatus()` to read the BEP's Status Register. It tests the DEA Command Port Available bit in the returned status word, and returns whether or not the port is busy.
3. Once the port is ready, *`client`* tells the DEA Device to send a command word to the DEA, using *`deaDevice`*`.sendCmd()`.

4. `sendCmd()` uses *bepReg*.`deaCmdReg()` to obtain the address of the DEA Command Port.
5. `sendCmd()` then writes the command word into the port.
6. The command port hardware then proceeds to serially clock the word out to the DEA.
7. Once the command word has been clocked to the DEA, the interface hardware latches the current microsecond counter into the DEA Timestamp register.

11.4.2 Use 2: Read a response from the DEA

Figure 26 illustrates the steps used to read a reply word, sent by the DEA in response to a command.

FIGURE 26. Read command reply



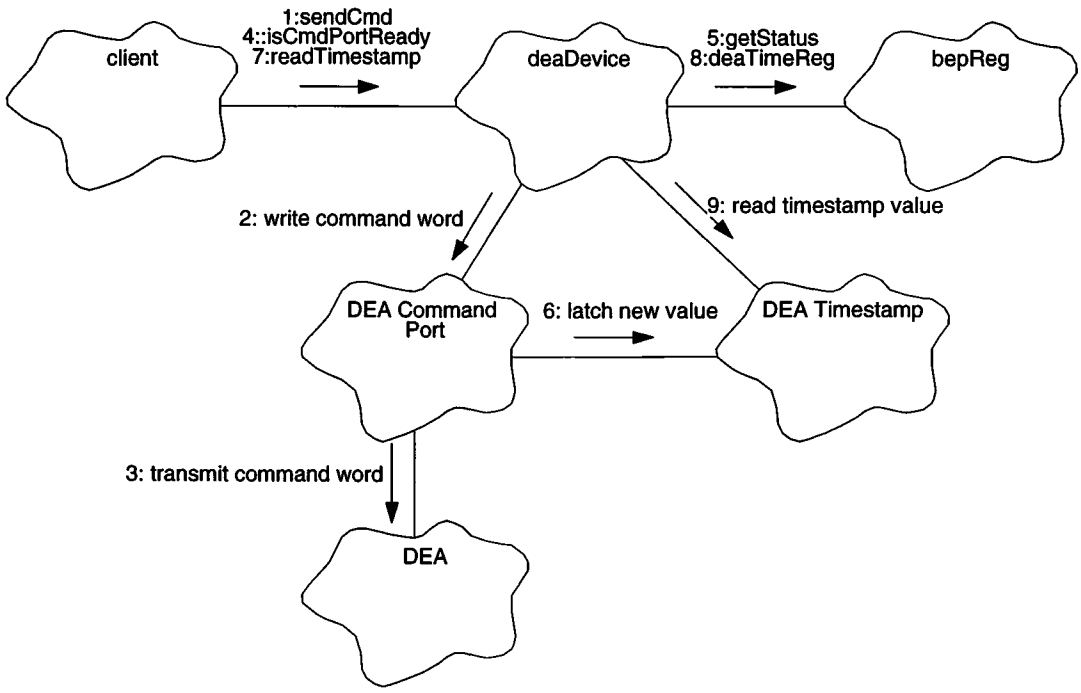
1. *client* sends a command to the DEA (which is intended to illicit a response) using *deaDevice*.`sendCmd()` (see Section 11.4.1 for more detail on this action).
2. `sendCmd()` writes the command word to the DEA Command Port.
3. The DEA command hardware transmits the command word to the DEA
4. Meanwhile the *client* polls the reply status using *deaDevice*.`isReplyReady()`. NOTE: It is *client*'s responsibility to detect and handle time-outs on the DEA interface.
5. `isReplyReady()` uses *bepReg*.`getStatus()` to read the BEP's Status Register. It then tests the DEA Reply Ready bit, and returns whether or not a response is in the DEA's Reply Port.

6. Eventually, the commanded DEA board generates a response to the sent command, and sends it back to the BEP's DEA Reply Port.
7. Once the reply is received, `isReplyReady()` indicates that a response is available. *`client`* then reads the reply word using *`deaDevice`*`.readReply()`.
8. `readReply()` uses *`bepReg`*`.deaStatReg()` to get the address of the DEA Reply Port.
9. `readReply()` reads the value from the reply port.
10. `readReply()` then resets the Reply Ready bit in the BEP's Status Register, using *`bepReg`*`.pulse()`. It then returns the read reply word to *`client`*.

11.4.3 Use 3: Read the latched timestamp

Figure 27 illustrates the steps used to read the latched DEA command timestamp.

FIGURE 27. Read DEA Command Timestamp



1. *`client`* sends a command to the DEA using *`deaDevice`*`.sendCmd()`.
2. `sendCmd()` writes the command word to the DEA Command Port
3. The command port hardware transmits the command word to the DEA
4. Meanwhile, *`client`* polls the command port status, using *`deaDevice`*`.isCmdPortReady()`. Once the port becomes available, the command word will have been completely sent, and the timestamp value will be valid.

5. `isCmdPortReady()` uses *bepReg*`.getStatus()` to obtain the command port status.
6. Eventually, the command word transmission will be complete, and the DEA interface hardware will latch the current microsecond counter into the DEA Timestamp register.
7. Once `isCmdPortReady()` indicates that the command has been completely sent, *client* calls *deaDevice*`.readTimestamp()` to read the latched microsecond timestamp value. NOTE: It is the caller's responsibility to detect and handle time-outs on the DEA interface.
8. `readTimestamp()` uses *bepReg*`.deaTimeReg()` to get the address of the timestamp register.
9. `readTimestamp()` then reads the contents of the timestamp register, and returns the read value to *client*.

11.5 Class DeaDevice

Documentation:

This class is responsible for sending commands to the Detector Electronics Assembly (DEA) and providing status information send by the DEA.

Export Control: Public

Cardinality: 1

Hierarchy:

Superclasses: **none**

Implementation Uses:

BepReg

Public Interface:

Operations:
```
isCmdPortReady()
isReplyReady()
readReply()
readTimestamp()
sendCmd()
```

Concurrency: Guarded

Persistence: Persistent

11.5.1 isCmdPortReady()

Public member of: **DeaDevice**

Return Class: **Boolean**

Documentation:

This function tests the current status of the DEA command port (via **BepReg**::`getStatus`()). It returns *`BoolTrue`* if the DEA is able to accept commands and *`BoolFalse`* if the DEA is not ready to accept commands (i.e. the interface port is busy).

Concurrency: Synchronous

11.5.2 isReplyReady()

Public member of: **DeaDevice**

Return Class: **Boolean**

Documentation:

This function tests the BEP status register (accessed via **BepReg**::`getStatus`()) to determine the DEA status register contains a reply word supplied by one of the DEA boards. This function returns *`BoolTrue`* if there is a DEA reply word ready for reading, and *`BoolFalse`* if a reply has not yet been received.

Concurrency: Synchronous

11.5.3 readReply()

Public member of: **DeaDevice**

Return Class: **unsigned**

Documentation:

This function reads and returns the contents of the DEA Status Register (whose address is determined using **BepReg**::`deaStatReg`()), and clears the DEA Reply Ready bit using **BepReg**::`pulse`().

Preconditions:

The caller must ensure that a command which requests a reply has been sent at least TBD microseconds prior to invoking this function. See `sendCmd`() and `isReplyReady`(). This function DOES NOT CHECK to ensure that a reply is in the status register. If no reply is present, garbage may be returned to the caller.

Postconditions:

The DEA reply ready status will remain de-asserted until another command requesting information is sent.

Concurrency: Guarded

11.5.4 readTimestamp()

Public member of: **DeaDevice**

Return Class: **unsigned**

Documentation:

This function reads and returns the contents of the microsecond timestamp, latched when the last command was sent to the DEA. This function uses **BepReg**::`deaTimeReg`() to get the address of the timestamp register.

Preconditions:

The client must ensure that the timestamp is read after the command has been sent to the DEA, using `isCmdPortReady`().

Concurrency: Guarded

11.5.5 sendCmd()

Public member of: **DeaDevice**

Return Class: **void**

Arguments:

unsigned *cmdWord*

Documentation:

This function writes *cmdWord* to the DEA's Command Interface port.

Preconditions:

The command port must be ready for use (see `isCmdPortReady()`). If not, a previous command may be corrupted by *cmdWord*.

Postconditions:

If the command requested status information, this information will be available after TBD microseconds (see `isReplyReady()`).

Concurrency: Guarded