

## Console Library for the Beam Line Tuners

Thomas S. Meyer  
BD/RFI

In the attempt to make the Labview Beam Line Tuner more usable by console application programmers a library of routines has been created and included in the CLIB ul\_cbsaux library. The following is the CLIB help files that describe the functions available for use.

### **blt\_get\_data (ul\_cbsaux)**

```
status.i4.v = blt_get_data(machine.i4.v, mode.i4.v, batch.i4.v,
                          BLT_DATA *Request, [buffer.i4.v] )
```

This routine accepts an integer indicating the machine to be viewed, an integer indicating the mode of the machine (BLT\_PROTON or BLT\_PBAR), an integer indicating the bunch or batch (TeV or MI) to be viewed and a data structure to be filled with BLT data. An optional argument can be passed as an integer indicating the buffer to be used in offset calculations.

machine	BLT_TEVETRON or BLT_MAIN_INJECTOR as defined in ul_cbsaux:bltuti.h
mode	BLT_PROTON or BLT_PBAR as previously defined.
batch	batch to be viewed this number must be between 0 and BLT_MAX_BATCH which at this time is valued at 35.
Request	reference to a structure of type BLT_DATA to be filled with the requested data
[buffer]	optional buffer designation in which to retrieve the data.

It makes the appropriate calls, fills the structure and returns.

This function returns ACNET status values as follows:

1 if positions were collected without error.

0 if positions were collected with errors.

This function requires the following include files:

ul\_cbsaux:bltuti, acnet\_errors

Related functions: blt\_get\_pos, blt\_activate\_spec, blt\_is\_fresh, blt\_get\_tag

C usage:

```
BLT_DATA dataSet;
int status;
int batch;
int buffer;
```

```
status = blt_get_data(BLT_TEVATRON, BLT_PROTON, batch, &dataSet,
                    buffer);
```

----- end of blt\_get\_data -----

### **blt\_get\_pos (ul\_cbsaux)**

```
status.i4.v = blt_get_pos(machine.i4.v, batch.i4.v, type.i4.v,
                        size.i4.v, BLT_POS *Request, )
```

This routine accepts an integer indicating the machine to be viewed, an integer indicating the bunch or batch (TeV or MI) to be viewed, an integer indicating whether the positions should be raw or filtered (BLT\_POS\_RAW or BLT\_POS\_FILTERED), an integer indicating the number of datapoints to return with a maximum of 1024, and a data structure to be filled with BLT positions.

machine	BLT_TEVETRON or BLT_MAIN_INJECTOR as defined in ul_cbsaux:bltuti.h
batch	batch to be viewed this number must be between 0 and BLT_MAX_BATCH which at this time is valued at 35.
type	BLT_POS_RAW or BLT_POS_FILTERED
size	0 to 1024 position data points to be retrieved
Request	reference to a structure of type BLT_POS to be filled with the requested data

It makes the appropriate calls, fills the structure and returns.

This function returns ACNET status values as follows:

1 if positions were collected without error.  
0 if positions were collected with errors.

This function requires the following include files:

ul\_cbsaux:bltuti, acnet\_errors

Related functions: blt\_get\_data, blt\_activate\_spec, blt\_is\_fresh, blt\_get\_tag

C usage:

```
BLT_POS positionSet;
int status;
int batch;
int size;
```

```
status = blt_get_pos(BLT_TEVATRON, batch, BLT_POS_RAW, size,
&dataSet);
```

----- end of blt\_get\_pos -----

### **blt\_get\_tag (ul\_cbsaux)**

```
status.i4.v = blt_get_tag(machine.i4.v, buffer.i4.v, BLT_TAG *Request )
```

This routine accepts an integer indicating the machine to be viewed, an integer indicating the buffer to be viewed and a data structure to be filled with Tag data.

machine	BLT_TEVETRON or BLT_MAIN_INJECTOR as defined in ul_cbsaux:bltuti.h
buffer	buffer designation between 0 and BLT_MAX_BUFFER in which to retrieve the data.
Request	reference to a structure of type BLT_POS to be filled with the requested data

It makes the appropriate calls, fills the structure and returns.

This function returns ACNET status values as follows:

1 if positions were collected without error.

0 if positions were collected with errors.

This function requires the following include files:

ul\_cbsaux:bltuti, acnet\_errors

Related functions: blt\_get\_data, blt\_get\_pos, blt\_activate\_spec, blt\_is\_fresh

C usage:

```
BLT_TAG tagSet;
```

```
int status;
```

```
int buffer;
```

```
status = blt_get_tag(BLT_TEVATRON, buffer, &tagSet);
```

----- end of blt\_get\_tag -----

### **blt\_activate\_spec (ul\_cbsaux)**

```
status.i4.v = blt_activate_spec(machine.i4.v, spec_number.i4.v)
```

This selects the appropriate machine then activates the BLT Measurement Specification indicated by the value in spec\_number.

machine	BLT_TEVETRON or BLT_MAIN_INJECTOR as defined in ul_cbsaux:bltuti.h
---------	--

spec_number	measurement specification number to activate
-------------	--

This function returns ACNET status values as follows:  
 1 if the Specification was activated without error.  
 0 if an error occurred.

This function requires the following include files:

ul\_cbsaux:bltuti, acnet\_errors

Related functions: blt\_get\_data, blt\_get\_pos, blt\_get\_tag, blt\_is\_fresh

C usage:

```
int status;
int spec_num;
```

```
status = blt_activate_spec(BLT_TEVATRON, spec_num );
```

----- end of blt\_activate\_spec -----

### **blt\_is\_fresh (ul\_cbsaux)**

```
status.i4.v = blt_is_fresh(machine.i4.v, setCheck.i4.v)
```

This will either set the BLT freshness number and exit, or check that the number has or has not changed. It then goes into a loop until either the freshness number changes, indicating new data, or a 10 second timeout is reached.

```
machine      BLT_TEVETRON or BLT_MAIN_INJECTOR as defined in
              ul_cbsaux:bltuti.h
setCheck     BLT_UPDATE_SET or BLT_UPDATE_CHECK
```

This function returns ACNET status values as follows:  
 1 if the data is fresh or the original value was set.  
 0 if the 10 second timeout was reached.

This function requires the following include files:

ul\_cbsaux:bltuti, acnet\_errors

Related functions: blt\_get\_data, blt\_get\_pos, blt\_get\_tag,  
 blt\_activate\_spec

C usage:

```
int freshness;
int results;
int event;
time_t time1;
```

```
freshness = blt_is_fresh(BLT_TEVATRON, BLT_UPDATE_SET);
```

```
abort_init();
results = waitev_c( event, -1, 19, WMNGR_CENTER );
abort_cancel();
time1 = time(NULL);

if( results == FALSE ){
    error_message_c(timeout_message,
ERR_SIMPLE_DISPLAY,RED,FALSE);
    return;
}

if( blt_is_fresh(BLT_TEVATRON, BLT_UPDATE_CHECK) != 1){
    error_message_c("Frontend Error: Data not fresh",
ERR_SIMPLE_DISPLAY,RED,FALSE);
}

----- end of blt_is_fresh -----
```