JLAB-TN-02-045

# **tdbbu** 1.6:
# *Another Tool for Estimating Beam Breakup due to Higher Order Modes*

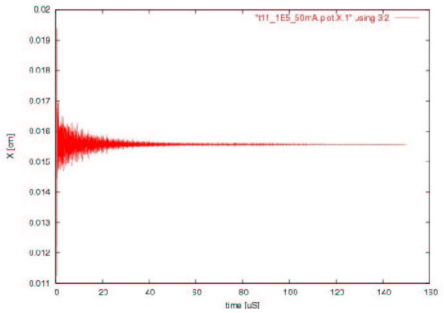
K.B.Beard, L.Merminga, B.Yunn, TJNAF

29 August 2003

The `tdbbu` program was written by G.Krafft and B.Yunn of TJNAF and CASA some time ago, later modified by L.Merminga, and then by K.Beard. Its purpose is to predict the transverse beam breakup instability thresholds in recirculating linacs.

Very simply, `tdbbu` reads in an input file describing the system of interest as drift spaces, lenses, and cavities (in terms of their **h**igher **o**rder **m**odes, or HOMs) and a recirculation matrix. The internal physics[1][2][3] is the same as that used by `matbbu`[4], as is the input file, which is very column specific and is described in detail in another note.[5] The HOMs on the X and Y axes are treated entirely independently.

From that input, the program generates matrices representing the particle transport and coupling to the HOMs. An initial bunch is put through off axis and subsequent bunches on axis. The output is used to make plots showing the transverse motion of the beam as a function of time. After an initial period of settling, the motion may grow, stay constant, or dampen; in an unstable situation the transverse motion grows without bound (Fig. 1).

Stable: 

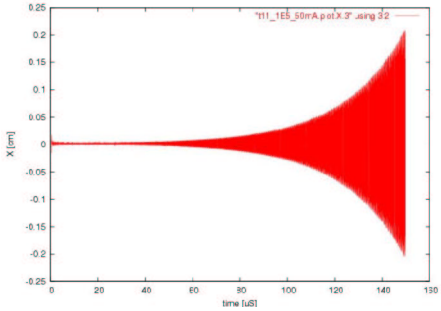

Unstable: 


Fig1. Stable and Unstable motion.

The time to run the simulation should exceed about 10 times the characteristic time, which is about $t \sim 2*Q/\omega$, so for f=2 GHz, $Q=10^5$, T~150 uS, and for $Q=10^6$, T~1500 uS. A rough estimate can be had from `tdbbu` using the -G or +G options. For example,

---

1 G.A.Krafft, J.J. Bisognano and S.Laubach, Jlab Tech Note JLAB-TN-01-011
2 J.J. Bisognano and R.L. Gluckstern, Proc. of 1988 Linear Accelerator Conf., 388 (1988)
3 G.A. Krafft and J.J. Bisognano, 1987 PAC Proceedings, 1356 (1987)
4 JLAB-TN-02-044, *matbbu 2.4: A Tool for Estimating Beam Breakup due to Higher Order Modes* K.B.Beard, L.Merminga, B.Yunn
5 JLAB-TN-02-043, *TDBBU and MATBBU Input File Format*, K.B.Beard, L.Merminga, B.Yunn

**$> tdbbu --DIMAD -i t11_1E5_50mA +G**
**tdbbuk: estimate J[mA]= 0.36827E+05 T[uS]= 0.15113E+04**
**tdbbuk: estimate avgJ[mA]= 0.92066E+03 T[uS]= 0.15113E+04**

# tdbbu Example

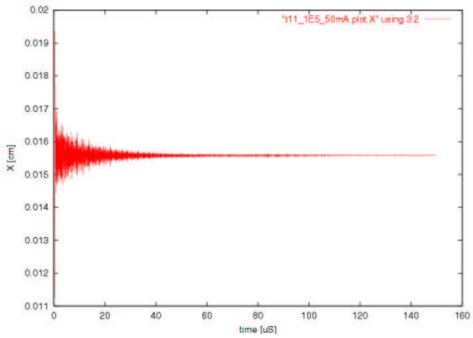
The input data file format is described elsewhere.[6] The beam current is specified in the input file's CMPNT line or via the `-j` $I_{mA}$ command line option. **The specified current there is the harmonic from the BEAM line *times* the beam current of interest.** For example, if the harmonic number was 40 and the real beam current of interest was 10 mA, the specified current would be 400 mA (40 * 10 mA). The **average** beam current may be set on the command line using the `-J` $I_{mA}$ option.

The "time to turn beam on", "time to turn beam off", and "time to run" in the REF line are the points to start the beam, stop the beam, and stop the simulation. Typically, the first is zero and the next two the same and sufficiently long; both may be overridden by the `-r` $T_{uS}$ command line option.

The simplest way to run `tdbbu` is to run the file and look at the output; note that the units of the recirculation matrix must be specified on the command line (`--DIMAD`) and the request be made that the output names be generated automatically (`-a`):

**$> tdbbu -i t11_1E5_50mA -a --DIMAD**

The output files are `t11_1E5_50mA.tdlog`, `t11_1E5_50mA.plot.X`, and `t11_1E5_50mA.plot.Y`. They were plotted using `gnuplot`.[7]

X: 

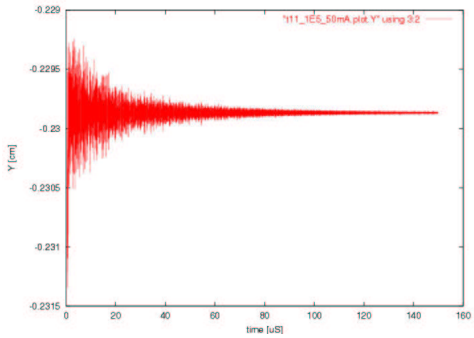

Y: 


Fig.2 X position, Y position as a function of time

The user can then either change the current by editing the input file (line CMPNT ) or by using the `-j` $I_{mA}$ option and repeating the process. This takes some time, in this example 150 uS of simulation required 172 CPU seconds on a 2GHz Pentium III.

---

6 JLAB-TN-02-043, *TDBBU and MATBBU Input File Format,* K.B.Beard, L.Merminga, B.Yunn

7 gnuplot 3.7, `http://www.gnuplot.info/`

An alternative is to use the new seek feature (+S or -S *lo:hi*) of tdbbu; the time is the same, but doesn't require the user to examine the output, edit the file, and rerun each cycle:

**$> tdbbu -i t11_1E5_50mA -a +S -v**

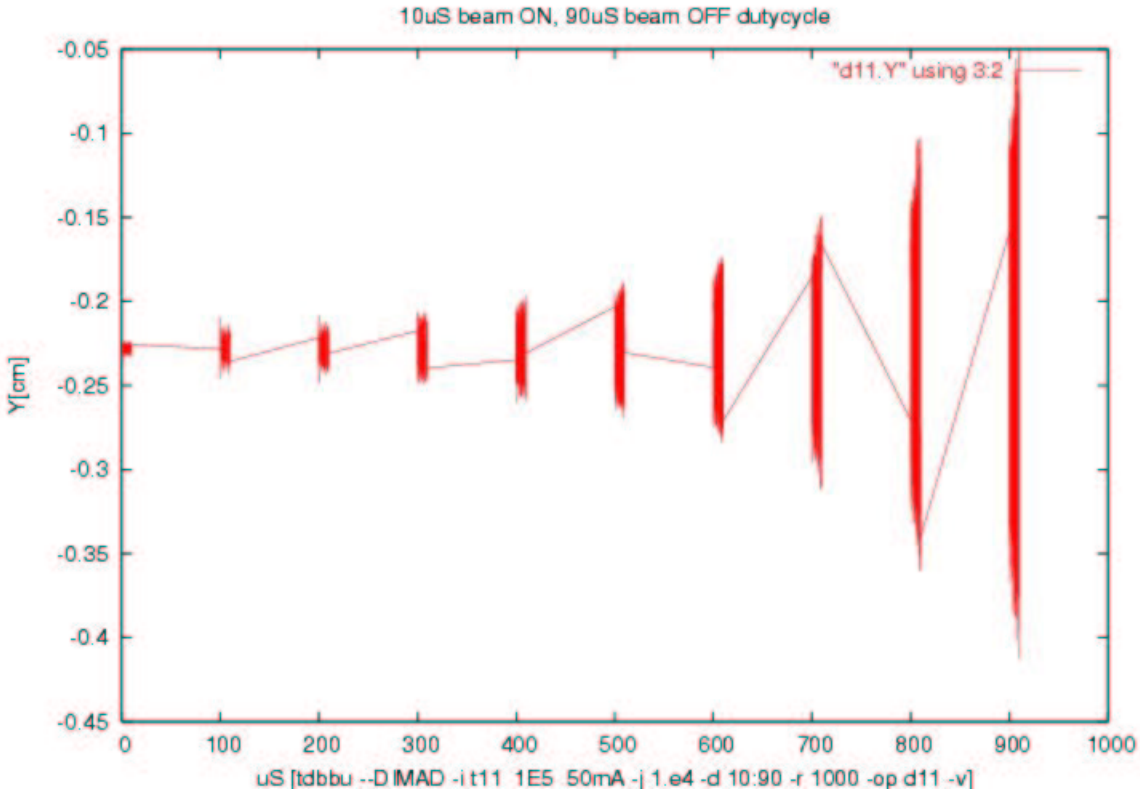
```
tdbbu:tdbbuk: opened gnuplot command file "t11_1E5_50mA.gp" OK
tdbbu:tdbbuk: opened input "t11_1E5_50mA" OK
tdbbu:tdbbuk: opened log "t11_1E5_50mA.tdlog.1" OK
tdbbu:tdbbuk: opened plot "t11_1E5_50mA.plot.X.1" OK
tdbbu:tdbbuk: opened plot "t11_1E5_50mA.plot.Y.1" OK
#tdbbu:t11_1E5_50mA:J_nochange:T_nochange: X= -0.37116E-03 Y= -0.24977E-03

#j=2000.000mA X stable:(1)2000.000  dead:none unstable:none
#j=2000.000mA Y stable:(1)2000.000  dead:none unstable:none
#   iteration      current[mA]    Xslp-status-#pts          Yslp-status-#pts
#guessshow:   1   0.20000E+04    -0.37116E-03 S 0.11E+05    -0.24977E-03 S 0.11E+05

  2000.000 ==> 4000.000
...

  4000.000 ==> 8000.000
tdbbu:tdbbuk: opened input "t11_1E5_50mA" OK
tdbbu:tdbbuk: opened log "t11_1E5_50mA.tdlog.3" OK
tdbbu:tdbbuk: opened plot "t11_1E5_50mA.plot.X.3" OK
tdbbu:tdbbuk: opened plot "t11_1E5_50mA.plot.Y.3" OK
#tdbbu:t11_1E5_50mA:J_8000.000:T_nochange: X= 0.35537E-03 Y= 0.34551E-03

#j=8000.000mA X stable:(2)4000.000  dead:none unstable:(3)8000.000
#j=8000.000mA Y stable:(2)4000.000  dead:none unstable:(3)8000.000
#   iteration      current[mA]    Xslp-status-#pts          Yslp-status-#pts
#guessshow:   1   0.20000E+04    -0.37116E-03 S 0.11E+05    -0.24977E-03 S 0.11E+05
#guessshow:   2   0.40000E+04    -0.18310E-03 S 0.11E+05    -0.44339E-04 S 0.11E+05
#guessshow:   3   0.80000E+04     0.35537E-03 U 0.11E+05     0.34551E-03 U 0.11E+05

  8000.000 ==> 5657.000
...

#j=4177.000mA X stable:(7)4968.000  dead:none unstable:(6)5188.000
#j=4177.000mA Y stable:(9)4177.000  dead:none unstable:(8)4362.000
#   iteration      current[mA]    Xslp-status-#pts          Yslp-status-#pts
#guessshow:   1   0.20000E+04    -0.37116E-03 S 0.11E+05    -0.24977E-03 S 0.11E+05
#guessshow:   2   0.40000E+04    -0.18310E-03 S 0.11E+05    -0.44339E-04 S 0.11E+05
#guessshow:   3   0.80000E+04     0.35537E-03 U 0.11E+05     0.34551E-03 U 0.11E+05
#guessshow:   4   0.56570E+04     0.96858E-04 U 0.11E+05     0.17957E-03 U 0.11E+05
#guessshow:   5   0.47570E+04    -0.60946E-04 S 0.11E+05     0.64215E-04 U 0.11E+05
#guessshow:   6   0.51880E+04     0.15179E-04 U 0.11E+05     0.12255E-03 U 0.11E+05
#guessshow:   7   0.49680E+04    -0.23874E-04 S 0.11E+05     0.93361E-04 U 0.11E+05
#guessshow:   8   0.43620E+04    -0.12746E-03 S 0.11E+05     0.77308E-05 U 0.11E+05
#guessshow:   9   0.41770E+04    -0.15665E-03 S 0.11E+05    -0.18994E-04 S 0.11E+05

#tdbbu: Xthr= 4968.0000- 5188.0000 mA Ythr= 4177.0000- 4362.0000 mA after 9 iterations
#tdbbu: avgXthr= 124.2000- 129.7000 mA avgYthr= 104.4250- 109.0500 mA after 9 iterations
```

Remember that the `X,Ythr` current is the subharmonic (40) times the average `avgX,Ythr` current. A `gnuplot` command file, `t11_1E5_50mA.gp`, was written to display all plots sequentially:

$> **gnuplot t11_1E5_50mA.gp**

A duty cycle may be specified on the command line; taking the same system, but with a 10% duty cycle (10uS beam on, 90uS beam off), with a current of 1000mA (40 x 25 mA) with a total runtime of 1000 uS:

$> tdbbu --DIMAD -i t11_1E5_50mA -j 1.e4 -d 10:90 -r 1000 -op d11 -v




## `tdbbu` hunting algorithm

The hunt within `tdbbu` keeps a small database of each attempt, and decides if X and Y were `STABLE` [**S**], `UNSTABLE` [**U**], or `CANT_SAY` [**?**]. After each attempt, the points from the X and Y plots are used to generate a single quantity for each axis in routine `RMS2SLOPE`.

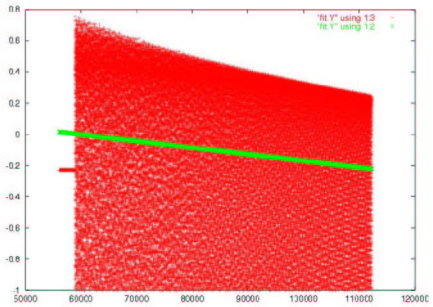
In the case of continuous beam, this quantity is the slope fitted to the RMS (root-mean-square) values of a moving window on upper half of the data. In the case of pulsed beam, it is the RMS values of all the data while the beam is on. If it is negative, the RMS

values are getting smaller, if positive, they are getting larger. The decision on whether the results are stable or not and what current to try next is made in routine `GUESSNEXT` (Fig. 3).

There is a dead zone where it is difficult to say whether the RMS values are growing or shrinking. For continuous beam, the size of the dead zone is from `DEADBAND(Lo)` to `DEADBAND(Hi)`, where the units are the fractional change in normalized RMS/point, and only the upper half of the data is considered. The default values are $-5*10^{-7}$ and $5*10^{-7}$ and can be changed using option `--dead` *`Lo:Hi`*.

In the case of pulsed beam, if the RMS slope is less than `DutyLooksStable`, the beam is presumed to be stable and if greater, unstable. The default value is $10^{-3}$, but can be changed using option `--looksstable` *`Hi`*.

The result for each current and axis is stored in a database; the next current is chosen based on the previous results.. The search continues attempting to find the "best" values until they are less than a `CLOSE_ENOUGH_FRACTION` (setable using `-c` *`frac,`* the default is 5%) apart, or until both edges of the dead zone are defined to within `CLOSE_ENOUGH_FRACTION/2`. The threshold is always expressed as a range; each iteration attempts to decrease the range. The X axis is done first, and then the Y axis, if required. The options `-X` and `-Y` will restrict the search to only that axis.

Stable: 

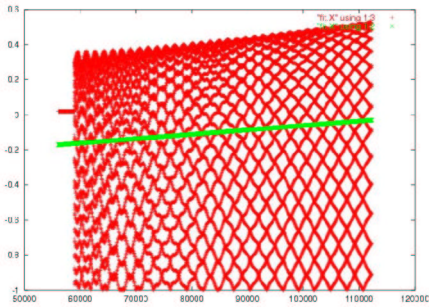
Unstable: 

Fig. 3. Stable and Unstable variation in RMS position.

To save time, if the transverse motion exceeds a reasonable limit (`TooFar=`$10^6$ um), the case is ruled `UNSTABLE` by routine `RMS2SLOPE` and the calculation aborted.

The history of the hunt is appended to the end of the `*.gp` file:

**$> tail -20 t11_1E5_50mA.gp**

```
plot "t11_1E5_50mA.plot.X.9" using 3:2
pause 3
set title "guess# 9 current= 4177.000 mA STABLE"
set ylabel "Y(cm)"
plot "t11_1E5_50mA.plot.Y.9" using 3:2
pause 3
```

```
#j=4177.000mA X stable:(7)4968.000  dead:none unstable:(6)5188.000
#j=4177.000mA Y stable:(9)4177.000  dead:none unstable:(8)4362.000
#  iteration     current[mA]    Xslp-status-#pts          Yslp-status-#pts
#guessshow:  1  0.20000E+04   -0.37116E-03 S 0.11E+05   -0.24977E-03 S 0.11E+05
#guessshow:  2  0.40000E+04   -0.18310E-03 S 0.11E+05   -0.44339E-04 S 0.11E+05
#guessshow:  3  0.80000E+04    0.35537E-03 U 0.11E+05    0.34551E-03 U 0.11E+05
#guessshow:  4  0.56570E+04    0.96858E-04 U 0.11E+05    0.17957E-03 U 0.11E+05
#guessshow:  5  0.47570E+04   -0.60946E-04 S 0.11E+05    0.64215E-04 U 0.11E+05
#guessshow:  6  0.51880E+04    0.15179E-04 U 0.11E+05    0.12255E-03 U 0.11E+05
#guessshow:  7  0.49680E+04   -0.23874E-04 S 0.11E+05    0.93361E-04 U 0.11E+05
#guessshow:  8  0.43620E+04   -0.12746E-03 S 0.11E+05    0.77308E-05 U 0.11E+05
#guessshow:  9  0.41770E+04   -0.15665E-03 S 0.11E+05   -0.18994E-04 S 0.11E+05
```

The hunt stopped when the upper and lower limits were close enough (in this case, the default 5% value).

# Comparison with Other Codes

Estimates of the 10KW FEL HOMs was made using both `tdbbu` and `matbbu,` and the results compared in other notes.[8] In general, `tdbbu` is a good choice when the characteristic time is short or the number modes is large, and a poor choice when the characteristic time is very long.

8 JLAB-TN-02-042, *Estimates of the Beam Breakup Thresholds in the 10KW FEL due to HOMs*,K.B.Beard, L.Merminga, B.Yunn

# Appendix 1. Help

The `tdbbu` program is run from the command line; the options must be separated by white space (`-h -V` is **not** equivalent to `-hV`) . It has an internal help (either `-h` or `--help`) and some examples (`--examples`). The recirculation units must be specified (version 1.6 on) using `--DIMAD` (meter-radian) or `--STANDARD` (cm-MeV/c). The most commonly used options may be listed:

### $> **tdbbu --help**

```
  tdbbu 1.6h2f2  22aug2003, G.Kraft, B.Yunn, L.Merminga, K.Beard, TJNAF

 Looks at transverse beam displacement due to HOMs in a recirculating linac

form:
$> tdbbu {option [value]} {option [value]} ... { <input } { >output }
-h  --help               - print brief list of options & quit [default]
+h  ++help               - print longer list of options & quit [default]
-V  --version            - print version info & quit
-v  --verbose            - print informational messages
    --DIMAD              - specify recirculation matrix as DIMAD (m-radian) units
    --STANDARD           - specify recirculation matrix as STANDARD (cm-MeV/c) units
-i  --input        FILE  - input file [-]
-op --outplot      BASE  - log and plot files basename (->BASE.tdlog,BASE.X,BASE.Y,BASE.
gp)
-a  --auto               - create plot and log filenames from input filename
-j  --current      VALUE - set beam current (mA*subharmonic#)
-J  --avgcurrent   VALUE - set average beam current (mA)
-G  --Guestimate         - guess at beam current and runtime
+G  ++Guestimate         - only guess at beam current and runtime and exit
+S  ++Seek               - seek thresholds automatically
    --examples           - print some examples

 also see: http://casa.jlab.org/internal/code_library/code_library.shtml
```

A full list of options may also be listed:

### $> **tdbbu ++help**

```
tdbbu 1.6h2f2  22aug2003, G.Kraft, B.Yunn, L.Merminga, K.Beard, TJNAF

 Looks at transverse beam displacement due to HOMs in a recirculating linac

form:
$> tdbbu {option [value]} {option [value]} ... { <input } { >output }
-h  --help               - print brief list of options & quit [default]
+h  ++help               - print longer list of options & quit [default]
-V  --version            - print version info & quit
+V  ++version            - print longer version info & quit
-v  --verbose            - print informational messages
    --DIMAD              - specify recirculation matrix as DIMAD (m-radian) units
    --STANDARD           - specify recirculation matrix as STANDARD (cm-MeV/c) units
-i  --input        FILE  - input file [-]
    ++DIMAD              - cavity matricies in DIMAD (m-radian) units
    ++STANDARD           - cavity matricies in STANDARD (cm-MeV/c) units
-C  --Cavmat       FILE  - cavity description file [ ]
    +-DIMAD              - both recirculation and cavity matricies in DIMAD (m-radian)
units
    +-STANDARD           - both recirculation and cavity matricies in STANDARD (cm-MeV/c)
units
-o  --output       FILE  - output log file [-]
+o  ++output             - no log file
-p  --plot         BASE  - plot files basename (->BASE.X,BASE.Y) [pfile]
-op --outplot      BASE  - log and plot files basename (-
>BASE.tdlog,BASE.X,BASE.Y,BASE.gp)
-a  --auto               - create plot and log filenames from input filename
-r  --runtime      TIME  - set input runtime (uS)
-d  --dutycycle   On:Off - set input beam on, off dutycycle (uS:uS)
+d  ++dutycycle          - force beam to be continuous
-j  --current      VALUE - set beam current (mA*subharmonic#)
-J  --avgcurrent   VALUE - set average beam current (mA)
-G  --Guestimate         - guess at beam current and runtime
+G  ++Guestimate         - only guess at beam current and runtime and exit
```

```
 -s  --scanplot     BASE  - just scan BASE.X and BASE.Y files
 -S  --Seek         j1:j2 - seek thresholds, starting with 1st:2nd (mA*subharmonic#)
current values
 +S  ++Seek               - seek thresholds automatically
     --dead         Lo:Hi - set dead band limits [-0.500E-06:0.500E-06]
     --fluc         CM    - set fluctuation beneath notice [0.100E-06]
     --lookstable   X     - set normalized max RMS variation for a stable pulsed beam
[0.100E-02]
     --aperature    CM    - set aperature for beam loss [0.200E+02] cm
 -D  --Discard      FRAC  - initial fraction of results to discard [0.0100]
 -c  --closeEnough  FRAC  - stop hunting when within fraction [0.0500]
 -fi --flushinterval   N  - output flush interval [250000]
 -m  --maximum         N  - quit after N total attempts [20]
 -X  --Xonly              - confine seeking to X axis
 -Y  --Yonly              - confine seeking to Y axis
     --adjustprint        - automatically adjust printing interval, if required
     ++adjustprint        - do not automatically adjust printing interval*
     --seed            N  - set random number seed
     --limits             - list compiled size limits and exit
     --examples           - print some examples

  also see: http://casa.jlab.org/internal/code_library/code_library.shtml
```

The output files may become very large as the run times become long; even if the available disk space is sufficient, a filesystem size limit of 2 Gb per file may become a problem. The `--adjustprint` option will reduce the printing interval automatically to avoid reaching this limit.

The compiled size limitations may be displayed:

$> tdbbu --limits

```
 tdbbu: compiled size ("tdbbu.cmn") limits:
    2000 MELM - max. total # of elements
    2000 MMOD - max. # of current loading modes
      28 MPAS - max. # of passes
   250000 MAX_plotdata - max # of points to fit/axis
```

Tho change these limits, the file `tdbbu.cmn` must be edited and the program recompiled.

# Appendix 2. Obtaining tdbbu

The tdbbu source code, example files, and executable binaries for a number of UNIX platforms (Linux, SunOS, HP-UX, AIX, ...) are available both onthe Jlab CUE filesystem[9] at `/group/casa/SUPPORTED/tdbbu` and from the CASA Code Library site http://casa.jlab.org/internal/code_library/code_library.shtml.

To run tdbbu from a CUE machine, the appropriate directory, `/group/casa/SUPPORTED/tdbbu/EXE.`*ostype*, must be in the user's UNIX PATH, where *ostype* is Linux, SunOS, HP-UX, and so forth .

The standard version of tdbbu typically requires **~70 Mb** of memory, but this may be reduced (for smaller systems of interest) by recompiling and relinking the code (refer to the `--limits` option in Appendix 1 and the recompilation instructions in Appendix 3).

9 http://cc.jlab.org/cue/

# Appendix 3. tdbbu code

tdbbu is an old code developed with little regard for maintenance, but considerable progress has been made toward making it easier to maintain, modify and use. The current version is 1.6h2f2.

A notable feature of tdbbu is that it uses C for both the main routine (shown in lower case in Figure 4), while the bulk of the code is in FORTRAN (shown in upper case in Figure 4). This was done to allow the code to read options from the command line and to provide platform independence.

Rebuilding the code requires the author's KBB[10] and KWRAP[11] libraries available from the same site as tdbbu (their purpose is to provide platform independence). It is strongly suggested that the author's organizational guide[12] [13] be read before attempting to rebuild the program, as the Makefiles depend on several (KBB_*) environmental variables to describe local idiosyncrasies. Once the files have been unpacked, the variables set for a supported platform, and the paths to the appropriate libraries set in the Makefile, it is only necessary to run the Makefile.

**$> make help**

```
make -f Makefile.Linux help
make[1]: Entering directory `/ERL/FEL/tdbbu/1.6'
#
# make <command>
#
# all            - update everything
# help           - print this help
# show           - print current value of required symbols and their definitions
# clean          - delete all binaries for this platform
# distclean      - delete all binaries for all platforms
# create_os      - create all binary directories for this platform
# destroy_os     - delete all binary directories for this platform
# destroy_all_os - delete all binary directories for all platforms
# archiveall     - create a compressed tar backup of everything
# archive        - create a compressed tar backup for export
#
make[1]: Leaving directory `/ERL/FEL/tdbbu/1.6'
```

**$> make show**

```
make -f Makefile.Linux show
make[1]: Entering directory `/ERL/FEL/tdbbu/1.6'
# --------- required symbols ------------
printenv KBB_DEV        #- location of KBB related libraries
/home/beard/DEVELOPMENT
printenv KBB_OSTYPE     #- single keyword for current operating system
Linux
printenv KBB_CC         #- C compiler
gcc
printenv KBB_F77        #- FORTRAN compiler
g77
printenv KBB_F2C_RULE   #- suffix on F77 objects when linking C+F77
_
printenv KBB_F2C_CASE   #- name case on F77 objects when linking C+F77
LOWERCASE
```

10 KBB 7.5g Library, K.Beard, http://casa.jlab.org/internal/code_library/casa_lib/KBB/DOC/

11 KWRAP: Kevin's Wrapper 0.1h, K.Beard, http://casa.jlab.org/internal/code_library/casa_lib/KWRAP/DOC/

12 http://casa.jlab.org/internal/code_library/casa_lib/KBB/DOC/bs.html

13 JLAB-TN-03-001, *My Code Development Style, Organization, amd Platform Dependency Guide*, K.B.Beard

```
# ------------------------------------
make[1]: Leaving directory `/ERL/FEL/tdbbu/1.6'
```

**$> make destroy_all_os;  make_create_os;  make**

| | | | | | | |
|---|---|---|---|---|---|---|
| main | TDBBUK | FLUSH | | | | |
| | | RMS2SLOPE | FLUSH | | | |
| | | TDBBU | WIPE_ALL | | | |
| | | | CLOCKK | | | |
| | | | PRECAL | FLUSH | | |
| | | | | ARRAYCHECK | WHEREAT | WHEREINDEXMAP |
| | | | | | FLUSH | |
| | | | FLUSH | | | |
| | | | ROUGH_ESTIMATE | FLUSH | | |
| | | | VERIFY_SPACE | FLUSH | | |
| | | | STEP0 | RANF | | |
| | | | | ARRAYCHECK | WHEREAT | WHEREINDEXMAP |
| | | | | | FLUSH | |
| | | | | FLUSH | | |
| | | | | LET_HOMS_DECAY | FLUSH | |
| | | | | EMTNCE | | |
| | | | RMS2SLOPE | FLUSH | | |
| | | | SECOND | | | |
| | | VERIFY_SPACE | FLUSH | | | |
| | | GUESSNEXT | GUESSREPORT | | | |
| | | | FLUSH | | | |
| | | GUESSSHOW | FLUSH | | | |

Key: [$c_{routine}$] [$FORTRAN_{routine}$] [**recursive**]  *...linking: FORTRAN:* XXXX(...) => *C:* xxxx_(...)

Figure 4. `tdbbu` structure

Figure 4 shows the dependencies of the tdbbu routines generated by splitcf [14]; a hyperlinked version of the above is in the documentation included with the distribution. The main routine is in the upper left corner; it calls the routines in the next column, and those routines call the ones next to them in the column after that, and so forth.  Each routine is briefly described below:

[ *SRC/arraycheck.f* ]
  [ *tdbbu_io.par* ]

```
SUBROUTINE ARRAYCHECK( LABEL, ARRAY, SIZE, SUM, PRODUCT )

      CHARACTER*(*) LABEL - !(input) debugging label for array
      REAL*8 ARRAY(*) - !(input) any size array
      INTEGER*8 SIZE - !(input) number of elements within array
      REAL*8 SUM - !(output) sum of all elements
      REAL*8 PRODUCT - !(output) product of all elements

      *
      *   A simple routine to look for corruption in an
```

14 http://casa.jlab.org/internal/code_library/casa_lib/SPLITCF/DOC/

```
*   array; returns the sum of all the elements and
*   the product of all the elements
*
```

*called by:* PRECAL , STEP0

*calls:* WHEREAT , FLUSH

[ SRC/clock.f ]

```
SUBROUTINE CLOCK( STRING )
```

CHARACTER*(*) *STRING* - !(output) hh:mm:ss

```
*
*  Returns the time as "hh:mm:ss" ("18:12:55" for example)
*  A kludge for the system call, uses the KBB library.
*  KBB 7/22/02
*
```

[ SRC/clockk.f ]

```
SUBROUTINE CLOCKK( STRING )
```

CHARACTER*(*) *STRING* - !(output) hh:mm:ss

```
*
*  Returns the time as "hh:mm:ss" ("18:12:55" for example)
*  A kludge for the CLOCK system call, uses the KBB library.
*  KBB 7/22/02, modified 8/18/03
*
```

*called by:* TDBBU

[ SRC/date.f ]

```
SUBROUTINE DATE( STRING )
```

CHARACTER*(*) *STRING* - !(output) ddMonYY

```
*
*  Returns the date as "ddMmmyy" ("22Jul02" for example)
*  A kludge for the system call, uses the KBB library.
*  KBB 7/22/02
*
```

[ SRC/emtnce.f ]
[ tdbbu.cmn ]

```
SUBROUTINE EMTNCE( N, X, PX, XAVE, PXAVE, SIGX, SIGPX, E )
```

INTEGER*8 *N* - !(input) THE NUMBER OF POINTS
REAL*8 *X(*)* - !(input) AN ARRAY CONTAINING THE POSITIONS
REAL*8 *PX(*)* - !(input) AN ARRAY CONTAINING THE MOMENTA
REAL*8 *XAVE* - !(output) THE AVERAGE POSITION
REAL*8 *PXAVE* - !(output) THE AVERAGE MOMENTUM
REAL*8 *SIGX* - !(output) THE RMS SPREAD IN POSITION
REAL*8 *SIGPX* - !(output) THE RMS SPREAD IN MOMENTUM
REAL*8 *E* - !(output) THE RMS EMITTANCE

```
C
C ************************************************
C ***                                          ***
```

```
C ***     RMS EMITTANCE CALCULATING SUBROUTINE     ***
C ***                                              ***
C ****************************************************
C
C     THE FOLLOWING SUBROUTINE COMPUTES THE EMITTANCE, POSITION
C     AND POSITION SPREAD, MOMENTUM AND MOMENTUM SPREAD, FOR
C     A SET OF DATA. THE INPUTS ARE
C           N       THE NUMBER OF POINTS
C           X(N)    AN ARRAY CONTAINING THE POSITIONS
C           PX(N)   AN ARRAY CONTAINING THE MOMENTA
C     AND THE OUTPUTS ARE
C           XAVE    THE AVERAGE POSITION
C           PXAVE   THE AVERAGE MOMENTUM
C           SIGX    THE RMS SPREAD IN POSITION
C           SIGPX   THE RMS SPREAD IN MOMENTUM
C           E       THE RMS EMITTANCE
C
*
*
*
```

*called by:* STEP0

[ *SRC/flush.f* ]

```
SUBROUTINE FLUSH( IOCHANNEL )

      INTEGER IOCHANNEL -

*
*  A kludge for AIX to emulate the
*  I/O FLUSH of other systems
*
```

*called by:* ARRAYCHECK , GUESSNEXT , GUESSSHOW , LET_HOMS_DECAY , PRECAL , RMS2SLOPE , ROUGH_ESTIMATE , STEP0 , TDBBU , TDBBUK , VERIFY_SPACE

[ *SRC/guessnext.f* ]
[ *tdbbu.par* ] [ *tdbbu.cmn* ] [ *tdbbu_io.par* ] [ *tdbbu_hunt.cmn* ] [ *tdbbu_enough.cmn* ] [ *tdbbu_axes.cmn* ] [ *tdbbu_duty.cmn* ]

```
SUBROUTINE GUESSNEXT( IO, JO, NXPTS, XSLP, NYPTS, YSLP, JNEXT, KEEPON,
THRX, THRY, CMTX, CMTY )
```

```
      INTEGER  IO - !(input) output channel for messages (if>0)
      REAL*8 JO - !(input) recent current [mA]
      INTEGER*8 NXPTS - !(input) number of X points
      REAL*8 XSLP - !(input) recent X result
      INTEGER*8 NYPTS - !(input) number of Y points
      REAL*8 YSLP - !(input) recent Y result
      REAL*8 JNEXT - !(output) next current [mA]
      LOGICAL  KEEPON - !(output) whether to continue or quit
      REAL*8 THRX(LO:HI) - !(output) best X threshold [mA]
      REAL*8 THRY(LO:HI) - !(output) best Y threshold [mA]
      CHARACTER*(*)  CMTX - !(output) comment on X axis
      CHARACTER*(*)  CMTY - !(output) comment on Y axis

*
*  A simple routine to choose whether to continue and
*  what current to try next.
*
```

*called by:* TDBBUK

*calls:* GUESSREPORT , FLUSH

[ *SRC/guessreport.f* ]
[ *tdbbu.par* ] [ *tdbbu_io.par* ] [ *tdbbu_hunt.cmn* ]
SUBROUTINE GUESSREPORT( *JO*, *AXIS*, *DED*, *BST*, *REPORT* )

REAL*8 *JO* - !(input) current J [mA]
INTEGER*4 *AXIS* - !(input) either X or Y axis
INTEGER*4 *DED(LO:HI,X:Y)* - !(input) current dead zone
INTEGER*4 *BST(LO:HI,X:Y)* - !(input) current best limits
CHARACTER*(*) *REPORT* - !(output) line showing the regions

```
*
* Writes the current status of the dead and
* best regions into "report" for an axis (X or Y)
*
*ex:  "X stable:(1)1.000 dead:(2)2.000-(3)2.500 unstable:(4)3.000"
*
```

*called by:* GUESSNEXT

[ *SRC/guessshow.f* ]
[ *tdbbu.par* ] [ *tdbbu.cmn* ] [ *tdbbu_io.par* ] [ *tdbbu_hunt.cmn* ] [ *tdbbu_duty.cmn* ]
SUBROUTINE GUESSSHOW( *PREFIX*, *O*, *SUFFIX* )

CHARACTER*(*) *PREFIX* - !(input) optional prefix to preceed each line
INTEGER *O* - !(input) already opened channel
CHARACTER*(*) *SUFFIX* - !(input) optional suffix to follow each line

```
*
*  Just write the guess status to IO
*
```

*called by:* TDBBUK

*calls:* FLUSH

[ *SRC/let_homs_decay.f* ]
[ *tdbbu.cmn* ] [ *tdbbu_tm.cmn* ] [ *tdbbu_io.par* ] [ *tdbbu_duty.cmn* ]
SUBROUTINE LET_HOMS_DECAY( *TIME_US* )

REAL *TIME_US* -

```
*
*   Let all HOMs decay by time_uS uSec.
*   (modified from "CAVITY UPDATE" in STEP0)
*
*
```

*called by:* STEP0

*calls:* FLUSH

[ *SRC/precal.f* ]
[ *tdbbu.cmn* ] [ *tdbbu_tm.cmn* ] [ *tdbbu_bug.cmn* ] [ *tdbbu_io.par* ] [ *tdbbu_est.cmn* ] [ *tdbbu_duty.cmn* ]
SUBROUTINE PRECAL( *IN*, *O*, *NEW_CURRENT*, *NEW_RUNTIME* )

INTEGER *IN* - !(input) already opened input channel
INTEGER *O* - !(input) already opened logging channel (O>0)
REAL*8 *NEW_CURRENT* - !(input&output) if>0, set new current[mA]
REAL*8 *NEW_RUNTIME* - !(input&output) if>0, new runtime

```
C=======================================================================
C       REVISED VERSION OF HELM'S INPUT SUBROUTINE
C
C     ADDED AN ELEMENT MATRIX AND AN ELEMENT RECIRC WITH GENERAL TRANSPORT
C     MATRIX OPERATIONS
C
C     REINJECTION POINT MARKED BY > IN MACHINE LATTICE (CAVITY,LENS,DRIFT
C     OR MATRIX)
C
C=======================================================================
* KBB: 7/23/02 - g77 disapproves of using integers for characters; changed
declarations
* KBB: 4/7/03 - enhanced CAVMAT auxillary file handling
*
* The CAVMAT file describes the transfer matrix for each cavity; each
* entry corresponds to a CECAV card in the input file.
*
************************************************************************
*
*
```

*called by:* TDBBU

*calls:* FLUSH , ARRAYCHECK

[ *SRC/ranf.f* ]
REAL*8 FUNCTION RANF( )

```
*
* A kludge to substitute for the system call;
* returns a random number (native size).
*
* KBB 7/22/02
*
```

*called by:* STEP0

[ *SRC/rms2slope.f* ]
[ *tdbbu.par* ] [ *tdbbu.cmn* ] [ *tdbbu_io.par* ] [ *tdbbu_hunt.cmn* ] [ *tdbbu_enough.cmn* ] [ *tdbbu_blowup.cmn* ] [ *tdbbu_duty.cmn* ]
SUBROUTINE RMS2SLOPE( *O*, *N*, *VALUE*, *TIMES*, *SLOPE* )

INTEGER *O* - !(input) output channel for messages if>0
INTEGER*8 *N* - !(input) number of plotted value
REAL*8 *VALUE(*)* - !(input) plotted value
REAL*8 *TIMES(*)* - !(input) times of plotted value
REAL*8 *SLOPE* - !(output) slope/channel of RMS in upper half

```
*
* Given the number of and X or Y values that appear in
* the TDBBU plot file, fits a slope to the
* running RMS of the 2nd half for continous beam, and
* for non-continous beam, fits the extremums for each
* beam ON/OFF cycle.
*
* Both destroy the input value list. - KBB
*
*
```

*called by:* TDBBU , TDBBUK

*calls:* FLUSH

[ *SRC/rough_estimate.f* ]
[ *tdbbu_est.cmn* ] [ *tdbbu_bug.cmn* ]
SUBROUTINE ROUGH_ESTIMATE( *J_MA*, *T_US* )

REAL*8 *J_MA* - !(output) estimate of current [mA]
REAL*8 *T_US* - !(output) estimate of required runtime [uS]

```
*************************************************************************
*
*  Given the HOM values and the energy exiting the HOMs,
*  returns a rough estimate of the relevant current J and
*  time required...
*
*  KBB 5/29/03
*
*************************************************************************
*
*
*
*
*
*
*
```

*called by:* TDBBU

*calls:* FLUSH

[ *SRC/second.f* ]
REAL FUNCTION SECOND( *T0* )

REAL *T0* -

```
*
*  A kludge for the system call that
*  returns the number of seconds since
*  midnight minus t0
*
```

*called by:* TDBBU

[ *SRC/step0.f* ]
[ *tdbbu.cmn* ] [ *tdbbu_tm.cmn* ] [ *tdbbu_bug.cmn* ] [ *tdbbu_io.par* ] [ *tdbbu_blowup.cmn* ] [ *tdbbu_axes.cmn* ] [ *tdbbu_duty.cmn* ]
SUBROUTINE STEP0( *O*, *X_PLT*, *Y_PLT*, *NXRSLT*, *XRSLT*, *TXRSLT*, *NYRSLT*, *YRSLT*, *TYRSLT*, *MAXRSLT* )

INTEGER *O* - !(input) already opened logging channel (O>0) [native size]
INTEGER *X_PLT* - !(input) already opened plotting channels [native size]
INTEGER *Y_PLT* - !(input) already opened plotting channels [native size]
INTEGER*8 *NXRSLT* - !(output) number of X results
REAL*8 *XRSLT(*)* - !(output) X results ([cm] for XPRNT, then [MeV/c] for PXPRNT)
REAL*8 *TXRSLT(*)* - !(output) time of X results (uS)
INTEGER*8 *NYRSLT* - !(output) number of Y results
REAL*8 *YRSLT(*)* - !(output) Y results ([cm] for YPRNT, then [MeV/c] for PYPRNT)
REAL*8 *TYRSLT(*)* - !(output) time of Y results (uS)
INTEGER*8 *MAXRSLT* - !(input) maximum number of X,Y results - keep only most recent results

```
C=====================================================================
C          VECTORIZED TWO DIMENSIONAL BUNCH PUSHING SUBROUTINE
C          ALSO PERFORMS A PRELIMINARY LOAD BASED ON THE VALUE OF KSTART
C          AND A CAVITY AMPLITUDE UPDATE AT EACH TIME STEP
C=====================================================================
* KBB: 7/23/02 - g77 disapproves of using integers for characters; changed
declarations
* KBB: 8/15/02 - added much debugging, write during processing, cleaned up code
somewhat
* KBB: 5/28/03 - added time column to output
* KBB: 7/16/03 - use a circular buffer internally; keep only last MAX_plotdata
points
*               internally and export only last MAXrslt values
*
```

*called by:* TDBBU

*calls:* RANF , ARRAYCHECK , FLUSH , LET_HOMS_DECAY , EMTNCE

[ *SRC/tdbbu.f* ]
[ *tdbbu.cmn* ] [ *tdbbu_io.par* ] [ *tdbbu_duty.cmn* ]

SUBROUTINE TDBBU( *IN*, *OT*, *X_PLT*, *Y_PLT*, *NEW_J*, *NEW_RUNT*, *NXS*, *XSLP*, *NYS*, *YSLP*, *SCAN* )

INTEGER *IN* - !(input) already opened input file
INTEGER *OT* - !(input) if>0, already opened logging IO channel
INTEGER *X_PLT* - !(input) already opened plot IO channels
INTEGER *Y_PLT* - !(input) already opened plot IO channels
REAL*8 *NEW_J* - !(input&output) if>0, use as new beam current; if Scan, returns estimate
REAL*8 *NEW_RUNT* - !(input&output) if>0, use as new runtime; if Scan, returns estimate
INTEGER*8 *NXS* - !(output) number of X data points
REAL*8 *XSLP* - !(output) slope of X-RMS
INTEGER*8 *NYS* - !(output) number of Y data points
REAL*8 *YSLP* - !(output) slope of Y-RMS
LOGICAL*4 *SCAN* - !(input) whether to only scan input file for rough estimate & return

```
C       A VECTORIZED TWO DIMENSIONAL BEAM BREAKUP SIMULATION CODE
C=====================================================================
C MULTIPASS BBU with ENERGY RECOVERY - Updated for UNICOS at NERSC
C *** Recirculation matrix now in DIMAD units
C *** THICK LENS VERSION
C *** To accomodate energy recovery scheme, specify the PASS NUMBER at
c     which ENERGY RECOVERY starts by using the variable 'IRECOV' in aprtr.
c     also, BUNCHING FREQUENCY should be changed to TWO TIMES OF
C     RF FREQUENCY to enable 180 degree phase slip from acceleration
c     to deceleration in cavities.
C *** THRESHOLD CURRENT is CURR(mA) divided by IPDL(bunching subharmonic)
C=====================================================================
C *** KSTART= 1 START OUT BY FILLING MACHINE WITH ONE BEAM IN STRUCTURE,
C     THEN TWO, ETC UP TO NPASS BEAMS IN STRUCTURE KSTART= 2 START OUT
C     WITH MACHINE HAVING TWO BEAMS IN STRUCTURE AND THEN BUILDING UP
C     KSTART MUST OBVIOUSLY BE L.E. NPASS
C *** X WIGGLE =AMPX*COS(2*PI*FREQX*TIME)*RANDOM
C *** PX WIGGLE = AMPPX*COS(2*PI*FREQPX*TIME)*RANDOM
C *** IRANDX OR IRANDPX = [0],1 [NOT] RANDOM ABOVE WIGGLES ARE ADDITIVE
C     TO HELMS INPUT X,PX
C *** RECIRCULATION MAP X=R11*X +R12*PX,PX=R21*X+R22*PX MAP OCCURS THEN
C     WIGGLE ADDED THEN REINJECTION
C *** CEBAF CAVITY TRANSFER MATRIX CAN BE INCLUDED WITH 'CECAV' CARD.
C     FIRST, RUN 'CEBAFCAVITY.FOR' ONLY WITH 'CECAV' CARDS TO CREATE
C     'CAVMAT' WHICH CONTAINS TRANSFER MATRIX. THEN RUN TDBBU WITH 'CAVMAT'.
C
C       G. A. KRAFFT 17-MAY-86
C
*
```

```
*
```

*called by:* TDBBUK

*calls:* WIPE_ALL , CLOCKK , PRECAL , FLUSH , ROUGH_ESTIMATE , VERIFY_SPACE , STEP0 , RMS2SLOPE , SECOND

[ *SRC/tdbbuk.f* ]
[ *tdbbu.par* ] [ *tdbbu.cmn* ] [ *tdbbu_io.par* ] [ *tdbbu_enough.cmn* ] [ *tdbbu_bug.cmn* ] [ *tdbbu_axes.cmn* ] [ *tdbbu_est.cmn* ] [ *tdbbu_duty.cmn* ] [ *tdbbu_blowup.cmn* ]

```
INTEGER FUNCTION TDBBUK( )

*
* A simple interface to the TDBBU program.  KBB 25jul02
*
```

*called by:* main

*calls:* FLUSH , RMS2SLOPE , TDBBU , VERIFY_SPACE , GUESSNEXT , GUESSSHOW

[ *SRC/verify_space.f* ]
[ *tdbbu.par* ] [ *tdbbu.cmn* ] [ *tdbbu_io.par* ] [ *tdbbu_duty.cmn* ]

```
SUBROUTINE VERIFY_SPACE( RUNTIME_US, ENOUGHSPACE, ERR )
```

```
REAL*8 RUNTIME_US - !(input) runtime in uS
LOGICAL*4 ENOUGHSPACE - !(output) whether there is enough memory
CHARACTER*(*) ERR - !(output) return a comment, if required
```

```
*
*  Verifies that there is sufficient space allocated
*  to run the simulation as specified
*
*
```

*called by:* TDBBU , TDBBUK

*calls:* FLUSH

[ *SRC/whereat.f* ]

```
SUBROUTINE WHEREAT( DECLARED_AS, WHERE, STRING, LSTRING )
```

```
CHARACTER*(*) DECLARED_AS - !(input) variable declaration
INTEGER*4 WHERE - !(input) 1-dimensional location
CHARACTER*(*) STRING - !(output) conventional string NAME(i,...,k)
INTEGER*4 LSTRING - !(output) length of string
```

```
*
*     Given a FORTRAN variable declaration string and
*     its 1D index, returns a string in the conventional
*     form.
*
*     For example declared_as="V(10,3,-1,1)" and
*     where=5 --> string="(5,1,-1)"
*
*
*
```

*called by:* ARRAYCHECK

*calls:* WHEREINDEXMAP

[ *SRC/whereindexmap.f* ]

```
SUBROUTINE WHEREINDEXMAP( NDIM, DIMLO, DIMHI, WHERE, OK, STR )
```

INTEGER*4 *NDIM* - !(input) number of dimensions
INTEGER*4 *DIMLO(*)* - !(input) low end for each dimension
INTEGER*4 *DIMHI(*)* - !(input) high end for each dimension
INTEGER*4 *WHERE* - !(input) 1-dimensional location
LOGICAL*4 *OK* - !(output) within boundries
CHARACTER*(*) *STR* - !(output) conventional string(i,...,k)

```
*
*     Given a variable's FORTRAN declared dimensionality
*     and its 1D index, returns a string str in the
*     conventional form - KBB
*
*     For example, if V(10,3,-1:1) -> Ndim=3
*                     dimLO(1)=1,dimLO(2)=1,dimLO(3)=-1
*                     dimHI(1)=10,dimHI(2)=3,dimHI(3)=1
*     and where= 5 --> str="(5,1,-1)"
*
*
*
```

*called by:* WHEREAT

[ *SRC/wipe_all.f* ]
[ *tdbbu.cmn* ] [ *tdbbu_tm.cmn* ] [ *tdbbu_duty.cmn* ]

```
SUBROUTINE WIPE_ALL( )
```

```
*
*  Zero out the DLOAD, R, PRNT, T_M, and
*  general_input_info COMMONs.
*
*
```

*called by:* TDBBU

[ *tdbbuk_main.c* ]

```
int main( int argc, char *argv[] )
```

```
int argc -
char *argv[] -
```

*Kevin's WRAPper*
*http://wims3.larc.nasa.gov/~beard/KWRAP/*
*defined during compilation*

*calls:* TDBBUK

---