Update on TRACY-2 documentation

Michael Böge, Michael.Boege@psi.ch $Log: tracy.sgml,v $ Revision 1.2 1999/06/02 12:05:46 boege header section cernlib.h and corr.h added Revision 1.1 1998/07/28 07:03:40 boege Initial revision

This document gives a summary of all changes made since the release of the original PASCAL version by J. Bengtsson.

Contents

1 Introduction

The main difference between the PASCAL version and the C version of TRACY-2 manifests itself in the fact that the C version is a library. In contrary the PASCAL version relied on the PASCAL-S compiler/interpreter developed by N. Wirth. This means that the PASCAL-S input file is replaced by a C program which is then linked with the TRACY library.

1.1 The PASCAL to C conversion

The conversion was performed utilizing the GNU `p2c` and `f2c` utility. The conversion went more or less smoothly (p2c did a good job !). The main work was spent on the reorganization of routines and data structures. Most of the routines available in the PASCAL version are still there, although some of them have slightly changed parameter lists. They are documented in section "Changed C library routines" (refer to 4.1 ()). Some routines were rewritten or added. New routines are documented in section "Added C library routines" (refer to 4.2 ()). Note: some of the routines namely the wiggler routines and the SVD routine taken from *LAPACK*) have been translated using `f2c`.

2 Installation

In order to compile the TRACY package successfully you have to provide recent versions of:

- `gcc`
- `p2c`
- `f2c`

If you have a LINUX system (lucky you !) this preriquisite should be already fulfilled.

You will also need the

- *Numerical Recipes* library `nrecipes-1p0.tar.gz`

which is available via *http://slsbd.psi.ch/tools/nrecipes/c/distrib/*.

The documentation ist provided in SGML format. To generate `tracy.ps` (*Source*), `tracy.html` (*Source*) and `tracy.txt` (*Source*) from `tracy.sgml` (*Source*) you will need

- `sgml-tools`

The TRACY package is distributed as gzip'ed tape archive file

- `tracy-x.xy.tar.gz`

You can get it from *http://slsbd.psi.ch/tools/tracy/distrib/*. Make sure that you download the most recent stable version. `Latest.tar.gz` points to the latest stable version.

Move the file to a directory of your choice preferably something like `~/tracy/` and unpack the archive using the command `gzip -cd tracy-version.tar.gz | tar xf -`. Browse through the README file and edit the Makefile (*Source*) in order to make changes to the compiler flags. Eventually define the variable `SYSNAME` if you want to work simultaneously on various platforms. Set `SYSNAME` to the output of `uname`.

Type `make all` to start the compilation. This will create the library `libtracy.a` with respect to `./$SYSNAME/lib/` and the documentation within `./doc/`. Browse through the documentation before you continue.

Type `make -f Makefile.example test` to compile and link the test program `tracytest.c`. Have a look at the example Makefile (*Source*). It allows you to keep your extensions to TRACY at a different place as the distribution.

The holy source ist under RCS (Revision Control System) [1] control. The RCS files are not part of the distribution.

3 Getting started

Have a quick look at the test program `tracytest.c` (*Source*):

```
#include <math.h>

#include "tracy.h"
#include "datatyp.h"
#include "physlib.h"

main() {
  boolean chroma;
  double dP;
  globvalrec globv;
  long lastpos;

  t2init();

  fi=fopen("sls.lat","r"); /* open the lattice input file  */
  fo=fopen("sls.out","w"); /* open the lattice output file */

  if (Lattice_Read(&fi, &fo)) { /* read lattice */
    Cell_Init(); /* initialize cell structure */

    getglobv_(&globv);

    /* define x/y aperture limits */
    for (i = 0; i <= globval.Cell_nLoc; i++)
      globv.maxampl[i][0] = globv.maxampl[i][1] = 1e0;

    globv.MatMeth = false; globv.Cavity_on = false;
    globv.radiation = false; globv.emittance = false;
    globv.pathlength = false; globv.CODimax = 15;
    globv.bpm = ElemIndex("mon");

    putglobv_(&globv);

    chroma=true;
    dP=0.0;

    getglobv_(&globv);
    printglob();

    Ring_GetTwiss(chroma, dP); /* get Twiss parameters */

    printglob(); /* print parameter list */
    printlatt(); /* dump linear lattice functions into "linlat.dat" */

    getcod(0.0, &lastpos); /* determine closed orbit */
    printcod(); /* dump closed orbit into "cod.dat" */
  }
}
```

This C program corresponds to the PASCAL `inp` file which was interpreted by the PASCAL-S interpreter which was itself written in PASCAL and part of TRACY. The `inc` files which were included at runtime by the PASCAL-S interpreter have been replaced by C headers. The master header file is `tracy.h`. It contains all definitions of data structures and routines contained in the TRACY library. As in the case of the PASCAL `inc` files there are high level header files containing routines which make use of the more basic library functions. `physlib.h` is a prominent example of such a header file (refer to 5.1 () for details).

4 TRACY C library routines

4.1 Changed C library routines

This section contains a description of routines which were changed since TRACY's translation from PASCAL to C. For a description of the routines which were left untouched refer to the Tracy-2 manual by J. Bengtsson [2].

- `gemit` in `physlib.h`

  ```
  Void gemit();
  ```

  Calculates radiation integrals (emittances), damping times, fractional tunes and generalized sigma matrices ala CHAO. Requires DA mode and lattice with cavity to run. The global 3D array `globval.ElemMat` contains the 6x6 sigma matrices for all positions. See also `printsigma`.

- `gcmat` in `lsoc.h`

  ```
  Void gcmat(long bpm, long bpmdis[mnp], long corr, long corrdis[mnp], long plane,
                  double (*A)[mnp], double (*U)[mnp], double (*V)[mnp], double *w,
                  long wdis[mnp], double (*Ai)[mnp])
  ```

  Determine the corrector-bpm correlation matrix `A`:

  ```
             -----------
            \/beta  beta
                  i     j
  A    = ------------- cos(nu pi - 2 pi|nu  - nu |)
   ij    2 sin(pi nu)                    i     j
  ```

  and perform Singular Value Decomposition (SVD) of `A` for plane `plane` with `A^-1 = Ai = V*w*U^T`. Please refer to `gcmatprint` for the description of the remaining parameters.

- `lsoc` in `physlib.h`

  ```
  Static Void lsoc(long niter, long bpm, long bpmdis[mnp], long corr, long corrdis[mnp],
                   double maxkick, long maxbits, long plane,
                   (*U)[mnp], (*V)[mnp], double *w);
  ```

  Perform an orbit correction based on the SVD algorithm. Iterate `niter` times for plane `plane`. Please refer to `gcmatprint` for the description of the remaining parameters. See *Numerical Recipes chapter 2.6* for an introduction to the SVD algorithm.

- `GirSet` in `bcosys.h`

  ```
  Void GirSet(double gfrac, double jfrac, double efrac, double bfrac);
  ```

  The rms girder alignment error defined in input file "bcosys.dat" ist multiplied with `bfrac`. The rms girder joints error ist multiplied with `jfrac` and the rms element error ist multiplied with `efrac`. If `bfrac` is != 0 `GirSet` simulates a simple beam-based alignment (BBA) procedure of qudrupoles with respect to adjacent monitors (`method=1`) or sextupoles with respect to adjacent monitors (`method=2`).

It simply correlates monitors-quadrupole (sextupole) combinations assuming a certain rms error of the aligment procedure. `bfrac`=0.1 would correspond to a BBA error which is 10 times smaller than the rms element error. The BBA settings are dumped to the file "bbaset.dat". The variable `method` is a local variable to `GirSet`. Angle errors of elements are now included for girders and elements. The rms angle is defined in "bcosys.dat" in micro degrees !-)

4.2 Added C library routines

This section contains a description of routines which were added since TRACY's translation from PASCAL to C. For a description of the routines which were left untouched refer to the Tracy-2 manual by J. Bengtsson [2].

- `getbumprec4a` in `t2bump.c`

  ```
  Void getbumprec4a (long ncorr, long *corr, long plane,
                     long corr1, long corr2, long corr3, long corr4, bumprec *bump);
  ```

  calculate asymmetric closed orbit bump for 4 correctors `corr1-4` and store kick ratios in `bump`

- `getbumprec4s` in `t2bump.c`

  ```
  Void getbumprec4s (long ncorr, long *corr, long plane,
                     long corr1, long corr2, long corr3, long corr4, bumprec *bump);
  ```

  calculate symmetric closed orbit bump for 4 correctors `corr1-4` and store kick ratios in `bump`

- `SetUpBump4a` in `t2bump.c`

  ```
  Void SetUpBump4a (long ncorr, long *corr, double dnumin, long plane);
  ```

  initialize `ncorr` asymmetric closed orbit bumps utilizing 4 adjacent coils around the machine with a minimal phase difference larger than `dnumin` for the plane `plane`.`corr` contains element numbers of the correction coils (See `inibump` in `physlib.h` for details).

- `SetUpBump4s` in `t2bump.c`

  ```
  Void SetUpBump4s (long ncorr, long *corr, double dnumin, long plane);
  ```

  initialize `ncorr` symmetric closed orbit bumps utilizing 4 adjacent coils around the machine with a minimal phase difference larger than `dnumin` for the plane `plane`.`corr` contains element numbers of the correction coils (See `inibump` in `physlib.h` for details).

- `InitBUMP4a` in `t2bump.c`

  ```
  Void InitBUMP4a (long *ncorr, long *hcorr, long *vcorr, double dnuhmin, double dnuvmin);
  ```

  initialize `ncorr[0]` horizontal and `ncorr[1]` vertical asymmetric closed orbit bumps utilizing 4 adjacent coils around the machine with a minimal phase difference larger than `dnuhmin` and `dnuvmin` for the horizontal and the vertical plane respectively. `hcorr` and `vcorr` contain element numbers of the correction coils (See `inibump` in `physlib.h` for details).

- `InitBUMP4s` in `t2bump.c`

  ```
  Void InitBUMP4s (long *ncorr, long *hcorr, long *vcorr, double dnuhmin, double dnuvmin);
  ```

  initialize `ncorr[0]` horizontal and `ncorr[1]` vertical symmetric closed orbit bumps utilizing 4 adjacent coils around the machine with a minimal phase difference larger than `dnuhmin` and `dnuvmin` for the horizontal and the vertical plane respectively. `hcorr` and `vcorr` contain element numbers of the correction coils (See `inibump` in `physlib.h` for details).

- `EigenVal` in `fit.c`

  ```
  Void EigenVal (double (*Ai)[mnp], long n, double *wr, double *wi);
  ```

  calculate real `wr` and imaginary `wi` part of the eigenvalues for the matrix `Ai`. Uses routines from the Numerical Recipes library.

- `QuadFit` in `fit.c`

  ```
  Void QuadFit (double *X, double *Y, long ndata, double *A, double Covar[][3],
                double *Chisq);
  ```

  perform a least square quadratic fit to (`X`,`Y`) using `ndata` samples. `A` contains the coefficients of the fit and `Chisq` the 3x3 covariance matrix. Uses routines from the Numerical Recipes library.

- `NormEigenVec` in `eigenv.c`

  ```
  Void NormEigenVec (vector *Vr, vector *Vi, double *wr, double *wi, vector *t6a);
  ```

  sort and normalize complex eigenvectors (`Vr`,`Vi`) with eigenvalues (`wr`,`wi`) and store the result in `t6a`. Used for the calculation of generalized sigma matrices ala CHAO in `emit`.

- `ElemIndex` in `t2lat.c`

  ```
  long ElemIndex(Char *name);
  ```

  return element family index. Note: in the PASCAL version the element family index could be comfortably accessed using the element name. This is no longer possible because we gave up on the interpretive PASCAL-S.

- `Mpole_GetdT` in `t2elem.c`

  ```
  double Mpole_GetdT(long Fnum1, long Knum1);
  ```

  return total roll angle of the element `Cell[ElemFam[Fnum1 - 1].KidList[Knum1 - 1]].Elem` which is a sum of a design ,a systematic error and a random error part.

- `Mpole_DefdTpar` in `t2elem.c`

  ```
  double Mpole_DefdTpar(long Fnum1, long Knum1, double PdTpar);
  ```

  Set design roll angle to `PdTpar` degrees.

- `Mpole_DefdTsys` in `t2elem.c`

  ```
  double Mpole_DefdTsys(long Fnum1, long Knum1, double PdTsys);
  ```

  Set systematic roll angle error to `PdTsys` degrees.

- `fft_double` in `fourierd.c`

  ```
  void fft_double
           ( unsigned NumSamples,         /* must be a power of 2 */
             int    InverseTransform,     /* 0=forward FFT, 1=inverse FFT */
             double *RealIn,              /* array of input's real samples */
             double *ImaginaryIn,         /* array of input's imag samples */
             double *RealOut,             /* array of output's reals */
             double *ImaginaryOut );      /* array of output's imaginaries */
  ```

  performs a onedimensional fourier transform. Is an alternative to `FFT`.

- `Circumference` in `physlib.h`

```
double Circumference();
```

returns the length of the ring.

- `printsigma()` in `physlib.h`

```
Void printsigma();
```

dump beam ellipse sigmas and twists calculated from generalized sigma matrices ala CHAO at every element to the file "sigma.dat".

- `digitize` in `physlib.h`

```
double digitize(double x, double maxkick, double maxsamp);
```

map `x` onto the integer interval (`-maxsamp` ... `maxsamp`) where `maxsamp` corresponds `maxkick`.

- `Dis_In` in `physlib.h`

```
Void Dis_In(long *bpmdis, long *vcorrdis, long *hcorrdis, long *wvdis, long *whdis)
```

Read a number of flag arrays from a plain text file named "*dis.dat*". The first line is always treated as a comment line. The first element on the following lines specifies the number of flags of type integer following on the same line. A space separated comment at the end of each line is allowed. `bpmdis` and `v(h)corrdis` are flag arrays marking bad bpm's and vertical (horizontal) correctors (0=ok, 1=faulty). `wv(h)dis` defines which SVD `w` values belonging to the vertical (horizontal) motion are explicitly set to zero (0=keep value, 1=set value to zero) in order to reduce the influence of bpm noise.

- `gcmatprint` in `lsoc.h`

```
Void gcmatprint(long bpm, long bpmdis[mnp], long corr, long corrdis[mnp], long plane,
                double (*A)[mnp], double (*U)[mnp], double (*V)[mnp], double *w,
                long wdis[mnp], double (*Ai)[mnp])
```

dump result of Singular Value Decomposition (SVD) of the corrector-bpm correlation matrix `A` for plane `plane` performed by `dsvdc` with `A^-1 = Ai = V*w*U^T` to the file "svd.dat". `bpm` and `corr` are element family indices (see `ElemIndex`) of bpm's and correctors. `bpmdis` and `corrdis` are flag arrays marking bad bpm's and correctors (0=ok, 1=faulty). `wdis` defines which `w` values are set to zero (0=keep value, 1=set value to zero) in order to reduce the influence of bpm noise.

- `minsq` in `cernlib.h`

```
Void minsq(long mfun, long nvar, double *fm, double *xn, double *en, long iprint,
           long niter, long cov, double xstep)
```

Interface to the cernlib routine MINSQ which minimizes a sum of squares of functions:

```
mfun   = number of functions
nvar   = number of variables
fm     = one-dim array for f1,f2,...,fm
xn     = one-dim array for x1,x2,...,xn
en     = starting incremint of xs
iprint = print after iprint iterations 0=final print only
ninter = number of iterations
cov    = =1 approx. error matrix printed, =0 otherwise
xstep  = initial step size searching for minimum along the line
         xstep <=0.5 recommended
```

A one-dim dummy array `wf[NW]` with `NW=200` is declared within `minsq`. Make sure that `nvar+mfun(nvar+1)+3nvar(nvar+1)/2` does not exceed `NW=200`. The user has to define the following routine:

```
Void fcn(long mfun, long nvar, double *fm, double *xn, long iflag);
```

`iflag` is =1 for first entry, =4 for normal entry and =3 for the final entry after the minimum has been found. A maximum of `MFUN=10` functions of `NVAR=10` variables is supported. For detailed information refer to the cernlib writeups.

- `minvar` in `cernlib.h`

```
Void minvar(double *x,double *y, double *r, double *eps, double *step, long maxf,
            double a, double b, char *f)
```

Interface to the cernlib routine MINVAR which calculates to an attempted specified accuracy the abscissa of a local minimum of real-valued function `f(x)` lying in a given interval `a,b` together with the value of the function at the minimum:

```
x    = estimate of the abscissa on exit estimate
y    = on exit f(x)
r    = on exit |x-x0|<r x0 x0=exact abscissa of a minimum of f
eps  = accuracy |x-x0|/(1+|x|)<=eps
step = initial search step
a,b  = search interval
f    = function f(x) to be minimized
```

The user has to define a function `f` and parse a pointer to it to `minvar`:

```
float f(float *x, long *iflag)
```

`iflag` is =0 for first entry, and =1 for subsequent entries. For detailed information refer to the cernlib writeups.

- `SkewCorr` in `corr.h`

```
Void SkewCorr(long niter, double *xn, double *fm, double *rn, double eps, double step,
              long maxf, double a, double b)
```

Minimizes the emittance coupling based on `minvar` utilizing 3 families of skew quadrupoles in the long straight sections of the SLS (`dsl11[p,m]`, `dsl12[p,m]`, `dsl13[p,m]`). Starting skew values may be assigned before `SkewCorr` is invoked. The families are assumed to be orthogonal. In case of a small nonorthogonality the minimzation can be repeated `niter` times in order find the minimum:

```
niter    = number of iterations
xn[MFUN] = estimate of the abscissa on exit estimate
fm[MFUN] = on exit f(x)
rn[MFUN] = on exit |x-x0|<r x0 x0=exact abscissa of a minimum of f
eps      = accuracy |x-x0|/(1+|x|)<=eps
step     = initial search step
a,b      = search interval
```

5 TRACY C library files

5.1 C library header files

This section ontains a summary of all TRACY header files located `./include`.

5.1.1 Main header files

tracy.h (*Source*)

```
RCS file: ./include/RCS/tracy.h,v
Working file: ./include/tracy.h
head: 1.2
total revisions: 3
description:
Main include file
```

datatyp.h (*Source*)

```
RCS file: ./include/RCS/datatyp.h,v
Working file: ./include/datatyp.h
head: 1.1
total revisions: 2
description:
magnet type definitions
```

physlib.h (*Source*)

```
RCS file: ./include/RCS/physlib.h,v
Working file: ./include/physlib.h
head: 1.2
total revisions: 3
description:
General physics routines
```

bcosys.h (*Source*)

```
RCS file: ./include/RCS/bcosys.h,v
Working file: ./include/bcosys.h
head: 1.2
total revisions: 3
description:
Girder and magnet misalignments
```

lsoc.h (*Source*)

```
RCS file: ./include/RCS/lsoc.h,v
Working file: ./include/lsoc.h
head: 1.2
total revisions: 3
description:
Orbit correction with SVD
```

mperr.h (*Source*)

```
RCS file: ./include/RCS/mperr.h,v
Working file: ./include/mperr.h
```

```
head: 1.2
total revisions: 3
description:
Set multipole errors
```

dynacc.h (*Source*)

```
RCS file: ./include/RCS/dynacc.h,v
Working file: ./include/dynacc.h
head: 1.1
total revisions: 2
description:
Dynamic Aperture routines
```

cernlib.h (*Source*)

```
RCS file: ./include/RCS/cernlib.h,v
Working file: ./include/cernlib.h
head: 1.1
total revisions: 1
description:
wrappers for CERNLIB routines
```

corr.h (*Source*)

```
RCS file: ./include/RCS/corr.h,v
Working file: ./include/corr.h
head: 1.2
total revisions: 2
description:
routines for coupling correction
```

5.1.2 FFT header files

This subsection contains header files of the FFT package (see *http://www.intersrv.com/˜dcross/fft.html* for details) which is a C subroutine library for computing the Discrete Fourier Transform (DFT) in one dimension. The `.h` files are located in `./include/fft/`.

fourier.h (*Source*)

```
RCS file: ./include/fft/RCS/fourier.h,v
Working file: ./include/fft/fourier.h
head: 1.1
total revisions: 2
description:
Contains definitions for doing Fourier transforms
       and inverse Fourier transforms(belongs to Don Cross FFT package)
```

audfile.h (*Source*)

```
RCS file: ./include/fft/RCS/audfile.h,v
Working file: ./include/fft/audfile.h
head: 1.1
total revisions: 2
description:
A class for reading and writing AUD files (belongs to Don Cross FFT package)
```

ddc.h (*Source*)

```
RCS file: ./include/fft/RCS/ddc.h,v
Working file: ./include/fft/ddc.h
head: 1.1
total revisions: 2
description:
Generic ddclib stuff (belongs to Don Cross FFT package)
```

ddcmath.h (*Source*)

```
RCS file: ./include/fft/RCS/ddcmath.h,v
Working file: ./include/fft/ddcmath.h
head: 1.1
total revisions: 2
description:
Contains useful math stuff (belongs to Don Cross FFT package)
```

5.2 C library files

5.2.1 Core "T2" C library files

t2lat.c (*Source*)

```
RCS file: RCS/t2lat.c,v
Working file: t2lat.c
head: 1.2
total revisions: 3
description:
lattice file parser
```

t2ring.c (*Source*)

```
RCS file: RCS/t2ring.c,v
Working file: t2ring.c
head: 1.1
total revisions: 2
description:
Routines for closed beam lines
```

t2cell.c (*Source*)

```
RCS file: RCS/t2cell.c,v
Working file: t2cell.c
head: 1.2
total revisions: 3
description:
Routines for Cell manipulation
```

t2bump.c (*Source*)

```
RCS file: RCS/t2bump.c,v
Working file: t2bump.c
head: 1.1
total revisions: 2
description:
Routines for sliding bump orbit correction
```

t2elem.c (*Source*)

```
RCS file: RCS/t2elem.c,v
Working file: t2elem.c
head: 1.1
total revisions: 2
description:
Routines for element definition and manipulation
```

t2common.c (*Source*)

```
RCS file: RCS/t2common.c,v
Working file: t2common.c
head: 1.1
total revisions: 2
description:
Common global variables (relict from the old PASCAL days)
```

5.2.2 Wiggler C library files

etwigg.c (*Source*)

```
RCS file: RCS/etwigg.c,v
Working file: etwigg.c
head: 1.1
total revisions: 2
description:
first order symplectic integrator for wiggler using expanded Hamiltonian
```

gfwigg.c (*Source*)

```
RCS file: RCS/gfwigg.c,v
Working file: gfwigg.c
```

```
head: 1.1
total revisions: 2
description:
Symplectic integrator for wiggler based on generating function
```

b2perp.c (*Source*)

```
RCS file: RCS/b2perp.c,v
Working file: b2perp.c
head: 1.1
total revisions: 2
description:
                  -   - 2        -
    Calculates |B x e| , where e is a unit vector in the direction of
    propagation
```

5.2.3 Math C library files

fit.c (*Source*)

```
RCS file: RCS/fit.c,v
Working file: fit.c
head: 1.1
total revisions: 2
description:
Interface to fit routines from the Numerical Recipes Library
```

fft.c (*Source*)

```
RCS file: RCS/fft.c,v
Working file: fft.c
head: 1.1
total revisions: 2
description:
FFT routine
```

svd.c (*Source*)

```
RCS file: RCS/svd.c,v
Working file: svd.c
head: 1.1
total revisions: 2
description:
Least-square closed orbit correction by singular value decomposition
```

dsvdc.c (*Source*)

```
RCS file: RCS/dsvdc.c,v
Working file: dsvdc.c
head: 1.1
total revisions: 2
description:
Linpack SVD routine
```

eigenv.c (*Source*)

```
RCS file: RCS/eigenv.c,v
Working file: eigenv.c
head: 1.1
total revisions: 2
description:
Eigenvalue routines
```

ety.c (*Source*)

```
RCS file: RCS/ety.c,v
Working file: ety.c
head: 1.1
total revisions: 2
description:
given a real general matrix, this subroutine
reduces a submatrix situated in rows and columns
low through igh to upper hessenberg form by
orthogonal similarity transformations.
```

dab.c (*Source*)

```
RCS file: RCS/dab.c,v
Working file: dab.c
head: 1.1
total revisions: 2
description:
Differential Algebra routines
```

mathlib.c (*Source*)

```
RCS file: RCS/mathlib.c,v
Working file: mathlib.c
head: 1.2
total revisions: 3
description:
Math routines
```

5.2.4 FFT C library files

This subsection contains C library files of the FFT package (see *http://www.intersrv.com/˜dcross/fft.html* for details) which is a C subroutine library for computing the Discrete Fourier Transform (DFT) in one dimension.

fourierd.c (*Source*)

```
RCS file: RCS/fourierd.c,v
Working file: fourierd.c
head: 1.1
total revisions: 2
description:
Contains definitions for doing Fourier transforms with Don Cross package
```

fftmisc.c (*Source*)

```
RCS file: RCS/fftmisc.c,v
Working file: fftmisc.c
head: 1.1
total revisions: 2
description:
 Helper routines for Fast Fourier Transform implementation by Don Cross
```

5.2.5 Miscellaneous C library files

dummy.c (*Source*)

```
RCS file: RCS/dummy.c,v
Working file: dummy.c
head: 1.1
total revisions: 2
description:
 dummy routine for f2c library
```

stringlib.c (*Source*)

```
RCS file: RCS/stringlib.c,v
Working file: stringlib.c
head: 1.1
total revisions: 2
description:
Routines for string manipulation (relict from the old PASCAL days)
```

pascalio.c (*Source*)

```
RCS file: RCS/pascalio.c,v
Working file: pascalio.c
head: 1.1
total revisions: 2
description:
PASCAL i/o routines and initialization routines (relict from the old PASCAL days)
```

6 References

1. W.F. Tichy, RCS–A System for Version Control, Software–Practice & Experience 15, 7 (July 1985), 637-654.

2. J. Bengtsson, Tracy-2 User's Manual, Feb 1997.