In the following, a general format for the input file of MITHRA is presented. The red icons or groups can be repeated in the text.

```
MESH
{
  length-scale              = < double | METER | DECIMETER | CENTIMETER | MILLIMETER | MICROMETER |
                                NANOMETER | ANGSTROM
  time-scale                = < double | SECOND | MILLISECOND | MICROSECOND | NANOSECOND |
                                PICOSECOND | FEMTOSECOND | ATTOSECOND >
  mesh-lengths              = < ( double, double, double ) >
  mesh-resolution           = < ( double, double, double ) >
  mesh-center               = < ( double, double, double ) >
  total-time                = < double >
  bunch-time-step           = < double >
  bunch-time-start          = < double >
  mesh-truncation-order     = < 1 | 2 >
  space-charge              = < true | false >
}

BUNCH
{
bunch-initialization
  {
    type                    = < manual | ellipsoid | 3D-crystal | file >
    distribution            = < uniform | gaussian >
    charge                  = < double >
    number-of-particles     = < int >
    gamma                   = < double >
    beta                    = < double >
    direction  = < ( double, double, double ) >
    position  = < ( double, double, double ) >
    sigma-position  = < ( double, double, double ) >
    sigma-momentum  = < ( double, double, double ) >
    transverse-truncation  = < double >
    longitudinal-truncation  = < double >
    bunching-factor  = < double between zero and one >
  }

bunch-sampling
{
sample  = < true | false >
directory  = < address according to UNIX convention >
base-name  = < name of the file >
rhythm  = < double >
}

bunch-visualization
{
sample  = < true | false >
directory  = < address according to UNIX convention >
base-name  = < name of the file >
rhythm  = < double >
}

bunch-profile
{
sample  = < true | false >
directory  = < address according to UNIX convention >
base-name  = < name of the file >
time  = < double >
rhythm  = < double >
}
}

FIELD
{
field-initialization
{
type  = < plane-wave | confined-plane-wave | gaussian-beam >
position  = < ( double , double , double ) >
direction  = < ( double , double , double ) >
polarization  = < ( double , double , double ) >
radius-parallel  = < double >
radius-perpendicular  = < double >
signal-type  = < neumann | gaussian | secant-hyperbolic | flat-top >
strength-parameter  = < double >
offset  = < double >
variance  = < double >
wavelength  = < double >
CEP  = < double >
}

field-sampling
{
sample  = < true | false >
type  = < over-line | at-point >
field  = < Ex | Ey | Ez | Bx | By | Bz | Ax | Ay | Az | Jx | Jy| Jz | F | Q >
directory  = < address according to UNIX convention >
base-name  = < name of the file >
rhythm  = < double >
position  = < ( double , double , double ) >
line-begin  = < ( double , double , double ) >
line-end  = < ( double , double , double ) >
resolution  = < double >
}

field-visualization
{
sample  = < true | false >
field  = < Ex | Ey | Ez | Bx | By | Bz | Ax | Ay | Az | Jx | Jy| Jz | F | Q >
directory  = < address according to UNIX convention >
base-name  = < name of the file >
rhythm  = < double >
}

field-profile
{
sample  = < true | false >
field  = < Ex | Ey | Ez | Bx | By | Bz | Ax | Ay | Az | Jx | Jy| Jz | F | Q >
directory  = < address according to UNIX convention >
base-name  = < name of the file >
rhythm  = < double >
time  = < double >
}
}

UNDULATOR
{
static-undulator
{
undulator-parameter  = < double >
period  = < double >
length  = < int >
polarization-angle  = < double >
offset  = < double >
}

static-undulator-array
{
undulator-parameter  = < double >
period  = < double >
length  = < int >
polarization-angle  = < double >
gap  = < double >
number  = < int >
tapering-parameter  = < double >
}

optical-undulator
{
beam-type  = < plane-wave | confined-plane-wave | gaussian-beam >
position  = < ( double , double , double ) >
direction  = < ( double , double , double ) >
polarization  = < ( double , double , double ) >
radius-parallel  = < double >
radius-perpendicular  = < double >
signal-type  = < neumann | gaussian | secant-hyperbolic | flat-top >
strength-parameter  = < double >
offset  = < double >
variance  = < double >
wavelength  = < double >
CEP  = < double >
}
```

```
}

EXTERNAL-FIELD
{
electromagnetic-wave
{
type  = < plane-wave | confined-plane-wave | gaussian-beam >
position  = < ( double , double , double ) >
direction  = < ( double , double , double ) >
polarization  = < ( double , double , double ) >
radius-parallel  = < double >
radius-perpendicular  = < double >
signal-type  = < neumann | gaussian | secant-hyperbolic | flat-top >
strength-parameter  = < double >
offset  = < double >
variance  = < double >
wavelength  = < double >
CEP  = < double >
}
}

FEL-OUTPUT
{
radiation-power
{
sample  = < false | true >
type  = < at-point | over-line >
directory  = < address according to UNIX convention >
base-name  = < name of the file >
plane-position  = < double >
line-begin  = < double >
line-end  = < double >
resolution  = < double >
normalized-frequency  = < double >
minimum-normalized-frequency  = < double >
maximum-normalized-frequency  = < double >
normalized-frequency-resolution  = < double >
}
}
```