

# **WRF-Fire User's Guide**

**May 2009**

This document describes the use of the fire scheme in the Advanced Research Weather Research Forecasting Model (ARW) version 3.1.

This document is complementary to

- the WRF ARW Technical Note, which describes the equations and numerics in WRF, available at [http://www.mmm.ucar.edu/wrf/users/docs/arw\\_v3.pdf](http://www.mmm.ucar.edu/wrf/users/docs/arw_v3.pdf)
- the WRF Fire Scheme document which is destined to become a chapter in the WRF ARW Technical Note at some point, and available from the authors at <http://math.ucdenver.edu/~jmandel/fires/wrf-fire-doc.pdf>,
- the ARW Version 3 Modeling System User's Guide, where a version of this document is destined to become a chapter at some point. ***All references to chapter numbers in this document refer to chapter numbers in the ARW Guide.*** The ARW Guide is available at [http://www.mmm.ucar.edu/wrf/users/docs/user\\_guide\\_V3/contents.html](http://www.mmm.ucar.edu/wrf/users/docs/user_guide_V3/contents.html)

The WRF-Fire code is released under the public domain notice and disclaimer located at <http://www.mmm.ucar.edu/wrf/users/public.html>

If you find this code useful please cite the article

**Mandel, J., J. D. Beezley, J. L. Coen, and M. Kim: *Data assimilation for wildland fires: Ensemble Kalman filters in coupled atmosphere-surface models.* IEEE Control Systems Magazine, to appear in June 2009, <http://arxiv.org/abs/0712.3965>**

even if the journal is an unusual venue for atmospheric science.

For support please subscribe to the wrf-fire mailing list at NCAR at

<http://mailman.ucar.edu/mailman/listinfo/wrf-fire>

The current version of this document is available from

<http://math.ucdenver.edu/~jmandel/fires/wrf-fire-guide.pdf>

For further information see the Open Wildland Fire Modeling E-Community

<http://openwfm.org>

***Contributors to this guide:*** Jonathan Beezley, Janice Coen, Jan Mandel

***Contributors to the WRF-Fire code:*** Jonathan Beezley, Janice Coen, Jan Mandel, John Michalakes, Net Patton

# Table of Contents

<b>1. INTRODUCTION .....</b>	<b>4</b>
<b>2. GETTING THE SOFTWARE .....</b>	<b>4</b>
<b>3. BUILDING THE WRF-FIRE CODE.....</b>	<b>5</b>
<b>4. RUNNING THE WRF-FIRE CODE (IDEAL DATA).....</b>	<b>5</b>
4.1 SHELL COMMANDS.....	5
4.2 INITIALIZATION.....	6
4.3 NAMELIST VARIABLES.....	6
<b>5. RUNNING THE WRF-FIRE CODE (REAL DATA).....</b>	<b>7</b>
5.1 NAMELIST.WPS .....	8
5.2 GEOGRID.....	9
5.3 UNGRIB AND METGRID .....	10
5.4 REAL AND WRF.....	11
<b>6. GEOGRID DATA SCRIPTS.....</b>	<b>13</b>
6.1 FETCH_DATA.PY .....	13
6.2 GEOGRID.PY.....	13
6.3 GEOGRID_WRAPPER.PY .....	15
<b>7. FIRE STATE VARIABLES .....</b>	<b>15</b>
7.1 FIRE STATE VARIABLES FOR POSTPROCESSING .....	15
7.2 OTHER STATE VARIABLES OF INTEREST .....	16
<b>8. SOFTWARE STRUCTURE .....</b>	<b>16</b>
8.1 CODING CONVENTIONS .....	16
8.2 PARALLEL EXECUTION .....	17
8.3 SOFTWARE LAYERS.....	18

## 1. Introduction

The fire scheme is included in WRF as a physics package. The fire code itself consists of several files in the `phys` directory, with names containing the string `sfire`, and modifications in the registry code to support 2D refined grids on the Earth's surface.

## 2. Getting the software

The current version of the code is available at <http://github.com/jbeezley/wrf-fire> as a tar file or using git. git is a source control system such as CVS or SVN and allows easy management of branches and coordination of many developers; it is the source control system used for the Linux kernel.

We recommend to download the software by git because you can get updates easily while you can keep and track your changes. You may need to ask your system group to install git or install it from sources. Then download the code by

```
git clone git://github.com/jbeezley/wrf-fire.git
```

This will create directory `wrf-fire` and you will have a local copy of the complete repository in a hidden subdirectory `.git`. By default, your local copy will have checked out the master branch. The master branch should be always the latest stable and tested version. We do not use release numbers. Instead, the version of the code is identified by a commit hash (a long string of hexadecimal numbers), which you can see at the top of the listing from

```
git log
```

You can leave your copy of the code on the master branch and use

```
git commit -a; git pull origin master:master
```

when you want to update the code to the current version and keep your changes, i.e., merge them into the new version.

In this simplest setup, you work in your own local (possibly modified) copy of the master branch. However, we strongly recommend that you learn a bit more about git and make your own branch.

Please contact us for instructions if you wish to contribute changes. The access to the code described here is read only. Developers with write access use a different download location and must work in their own branch.

See file `doc/README_git.txt` for more information.

### 3. Building the WRF-Fire code

Download NetCDF and install. The current version is 4.0. Set the environment variables `CC FC F90` to your compilers, then configure for your NetCDF install location by something like

```
./configure --prefix=/opt/netcdf
```

(of course, use your own location) and proceed with installation by

```
make
```

```
make install
```

Set the environment variable to the top level NETCDF directory like:

```
setenv NETCDF /opt/netcdf
```

(Of course, replace the path by your location of NETCDF above.) ***You must use the same compilers to build NETCDF as you use to build WRF-Fire.*** See Chapter 1 for more information.

Then,

```
cd WRFV3
```

```
./configure
```

Select an option for your computer architecture and compilers. If you are trying to get this working on different compilers, you can try copying a section close to your case in `WRFV3/arch/configure_new.defaults` to the end of that file and modifying to suit your needs. However, this is not supported and may not work.

Note that currently `gfortran` cannot compile WRF 3.1 while `g95` works fine.

```
./compile em_fire >& compile.log
```

(Assuming you are using `csh` as your shell.) This will take a while. Make sure that the file `compile.log` contains no errors. ("`grep Error compile.log`" shouldn't return anything.)

See Chapter 2 for more information.

### 4. Running the WRF-Fire code (ideal data)

#### 4.1 Shell commands

To run the provided example,

```
cd WRFV3/test/em_fire/small
```

```
./ideal.exe
```

This creates files like `wrfinput_d01` and needs to be done only when the file `input_sounding` or the domain definitions change. Then,

```
./wrf.exe
```

This runs what is known as an ideal test case. Ideal runs are distinguished from real runs in that they require no initialization data and their domains are not a representation of a section of the earth's surface. The program `ideal.exe` creates the inputs for WRF that would otherwise need to be obtained elsewhere.

## 4.2 Initialization

The initial state of the atmosphere is given in the file `input_sounding`; see Chapter 4. The mesh sizes and ignition are given in the file `namelist.input`, see "Description of namelist variables" below and Chapter 5.

## 4.3 Namelist variables

The following variables have been added to the file `namelist.input` for the fire model. See Chapter 6 for the description of other namelist variables.

Variable names	Value	Description
<code>&amp;domains</code>		Domain definition
<code>sr_x</code>	10	Fire mesh is 10 times finer than the innermost atmospheric mesh in the x direction, must be even
<code>sr_y</code>	10	Fire mesh is 10 times finer than the innermost atmospheric mesh in the y direction, must be even
<code>&amp;fire</code>		Fire control
<code>Ifire</code>	0	The fire model skipped
	2	The fire model runs
<code>fire_fuel_read</code>	0	How to set the fuel data
	-1	real data from WPS
	0	set to <code>fire_fuel_cat</code> everywhere
	1	vegetation by altitude
<code>fire_num_ignitions</code>	3	Number of ignition lines, max. 5 allowed
<code>fire_ignition_start_x1</code>	1000.	x coordinate of the start point of the ignition line 1. All ignition coordinates are given in m from the lower left corner of the innermost domain

<code>fire_ignition_start_y1</code>	500.	x coordinate of the start point of the ignition line 1
<code>fire_ignition_end_x1</code>	1000.	Y coordinate of the end point of the ignition line 1. Point ignition (actually a small circle) is obtained by specifying the end point the same as the start point.
<code>fire_ignition_end_y1</code>	1900.	y coordinate of the end point of the ignition line 1
<code>fire_ignition_radius1</code>	18.	Everything within 18 meters from the ignition time will be ignited.
<code>fire_ignition_time1</code>	2.	Time of ignition in s since the start of the run
<code>fire_ignition_start_x2</code> ...		Up to 5 ignition lines may be given. Ignition parameters with the number higher than <code>fire_num_ignitions</code> are ignored.
<code>fire_ignition_time5</code>		
<code>fire_print_msg</code>	1	0: no messages from the fire scheme 1: progress messages from the fire scheme
<code>fire_print_file</code>	0	0: no files written (leave as is) 1: fire model state written every 10 s into files that can be read in Matlab. See <code>wrf/doc/README_vis.txt</code> in the developers' version.

There are several more variables in the namelist for developers' use only to further develop and tune the numerical methods. *Leave as is* unless directed by the developers.

## 5. Running the WRF-Fire code (real data)

Running WRF with real data is a complicated process of converting data formats and interpolating to the model grid. This process is simplified by the WRF Preprocessing System (WPS). The purpose of this section is to summarize the use of this system and to highlight the differences in its use between fire and ordinary atmospheric simulations. For more complete documentation of WPS, see Chapter 3 of the WRF-ARW user's guide.

WPS consists of three utility programs: `geogrid.exe`, `ungrib.exe`, and `metgrid.exe`. Each program is designed to take existing data sets and convert/interpolate them into an intermediate format. The build system for WPS is similar to that of WRF. NetCDF must be installed and the environment variable

NETCDF should be set to the installation prefix. Run the configure script in the main WPS directory, pick a configuration option from the list, and then run compile. Note that WRF itself must be built prior to compiling WPS. In addition, the build process assumes that WRF exists in `../WRFV3/`. WRF should be configured as described in Section 3 and compiled with the command

```
./compile em_real >& compile.log
```

The WPS can be configured from inside the top level directory `wrf-fire/WPS` with the command

```
./configure
```

and compiled in the same directory with the command

```
./compile >& compile.log
```

Upon successful completion the three binaries listed above should exist in the current directory.

Because the WPS programs are, for the most part, not processor intensive, it is not generally necessary to compile these programs for parallel execution, even if they do support it. Typical usage of WRF with real data involves doing all of the preprocessing work either locally on a workstation or on the head node of a supercomputer. The intermediate files are all architecture independent, so they can be transferred between computers safely. If you intend on using a super computer for the main simulation, it is advisable to generate the WPS output locally and transfer the `met_em` files to the computer you will be using for WRF-Fire. The `met_em` files are much smaller than the `wrfinput` and `wrfbdy` files and can be transported easily. This also eases the process of dealing with the dependencies of the python scripts described below because it may not be easy or even possible to meet these requirements on a shared parallel computer.

## 5.1 namelist.wps

The simulation domain is described in the file `namelist.wps`. This namelist contains four sections one for each of the main binaries created in WPS and one shared among them all. This file, as distributed with WRF-Fire, is set up for a test case useful for testing, but in general one needs to modify it for each simulation domain. The following table lists namelist options that can be modified. Other options in this file are generally not necessary to change for WRF-Fire simulations. See the WRF-ARW user's guide for more information.

Variable names	Description
<code>&amp;share</code>	Shared namelist options
<code>max_dom</code>	Number of nested domains to use

start_date/end_date	Starting/ending date and time to process atmospheric data in the format YYYY-MM-DD_hh:mm:ss. These times should coincide with reanalysis cycles for your atmospheric data (hours 00,03,06,09,12, etc. for 3 hour NARR data). The simulation window in which you are interested in running must be inside this interval.
subgrid_ratio_[xy]	The refinement ratio from the atmospheric grid to the fire grid.
interval_seconds	Number of seconds between each atmospheric dataset. (10800 for 3 hour NARR data)
&geogrid	Domain specifications
parent_id	When using nested grids, the parent of the current grid, or 0 if it is the highest level.
parent_grid_ratio	The refinement ratio from the parent grid (ignored for top level grid) (only 3 or 5 is supported by WRF)
[ij]_parent_start	The indices on the parent grid of the lower left corner of the current grid (ignored for top-level grid)
e_we/e_sn	The size of the grid in the x/y axis
dx/dy	Resolution of the grid in the x/y axis
map_proj, true_lat[12], stand_lon	Projection specifications. Lambert is typically used for central latitudes such as the continental US. For small domains, the projection used doesn't matter much.
ref_x/ref_y	Grid index of a reference point with known geographic location. Defaults to the center of the domain.
ref_lon/ref_lat	The location (longitude/latitude) of the reference point.
geog_data_path	Absolute or relative path to geogrid data released with WPS ( <a href="http://www.mmm.ucar.edu/wrf/src/wps_files/geog_v3.1.tar.gz">http://www.mmm.ucar.edu/wrf/src/wps_files/geog_v3.1.tar.gz</a> )

## 5.2 Geogrid

The geogrid executable acts exclusively on static datasets (those that don't change from day to day) such as surface elevation and land use. Because these datasets are static, they can be obtained as a single tar ball from the main WPS distribution website in resolutions of 10 minutes, 2 minutes, and 30 seconds. The geogrid executable extracts from these global data sets what it needs for the current domain. While resolutions of this magnitude are acceptable for ordinary atmospheric simulations, these datasets are too coarse for a high-resolution fire simulation.

Because geogrid was not designed with datasets of a meter scale resolutions in mind, small modifications to the standard procedure have been necessary.

Generally, it is not possible to release a global dataset at resolutions of less than one second; the download would be prohibitively large (potentially 100's of terabytes or more). It is necessary to obtain a separate data set for each spatial location being simulated. High resolution static data can be obtained freely from sources such as the USGS website for the continental US; however, the process of converting this to a format readable by geogrid can be cumbersome, requiring specialized knowledge of low-level numeric representation and geographic projections. In order to ease this process, several python scripts are included in the WRF-Fire distribution that obtain this data from the USGS server and convert it automatically. Because they fetch data from USGS servers, they only work on domains within the continental US. These scripts are located in `other/convert_to_geogrid/`. Among these files are the main scripts `fetch_data.py`, `geogrid.py`, and `geogrid_wrapper.py`. They can be executed by the python interpreter as `python fetch_data.py`, etc. They all support a flags: `-h` prints out a basic usage statement and `-v` prints more verbose output. These commands are described in detail in Section 6. These scripts require a number of external programs to be installed:

- Python, <http://www.python.org>
- Twill, <http://twill.idyll.org>
- NumPy, <http://numpy.scipy.org>
- GDAL, <http://www.gdal.org>

Consult the homepages for installation instructions for your platform.

If all of the dependencies have been installed correctly and the standard WPS static data has been extracted somewhere on you local filesystem with the namelist variable `geog_data_path` set to its location, then the following command should download any other static data necessary for a WRF-Fire real data run:

```
./geogrid_wrapper.sh
```

The output of this command will indicate if everything was successful. Once completed there will be new files, `geo_em.d?? .nc`, one for each nested grid. The python scripts that this procedure relies on are experimental at best. Let us know if you experience any problems with them.

### 5.3 Ungrib and Metgrid

The `ungrib` executable performs initial processing on atmospheric data. There are many different datasets that can be used as input to `ungrib`. One must obtain this data manually for a given simulation. Because fire simulations will be at a much higher resolution than most atmospheric simulations, it is advisable to get as high resolution data as possible. The 32 km resolution data from the North American

Regional Reanalysis (NARR) is likely a good choice. This data is available freely from [https://dss.ucar.edu/datazone/dsszone/ds608.0/NARR/3HRLY\\_TAR/](https://dss.ucar.edu/datazone/dsszone/ds608.0/NARR/3HRLY_TAR/). For real data WRF runs, three individual datasets from this website are required: 3d, flx, and sfc. To use them, download the files for the appropriate date/time and extract them somewhere on your filesystem. The files have the naming convention, NARR3D\_200903\_0103.tar. NARR indicates it comes from the NARR model, 3D indicates that it is the atmospheric data fields, and 200903\_0103 indicates that it contains data from March 1<sup>st</sup> through 3<sup>rd</sup> of 2009. Once these files are extracted, they must be linked into the main WPS directory with the command `link_grib.csh`. It takes as arguments all of the files extracted from the dataset. For example, if you extracted these files to `/home/joe/mydata`, then you would issue the command:

```
./link_grib.csh /home/joe/mydata/*
```

into the top level WPS directory. Each atmospheric dataset requires a descriptor table known as a variable table to be present. WPS comes with several variable tables that work with most major data sources. These files reside in `WPS/ungrib/Variable_Tables/`. The appropriate file must be symlinked into the top level WPS directory as the file `Vtable`. For NARR data, type:

```
ln -sf ungrid/Variable_Tables/Vtable.NARR Vtable
```

Once this has been done, everything should be set up properly to run the `ungrib` command:

```
./ungrib.exe
```

Finally, the program `metgrid` combines the output of `ungrib` and `geogrid` to create a series of files, which can be read by WRF's `real.exe`. This is accomplished by

```
./metgrid.exe
```

Assuming everything completed successfully, you should now have a number of files named something like `met_em.d01.2009-03-01_12:00:00.nc`. These should be copied or linked to your `WRFV3/test/em_real` directory. If any errors occur during execution of `ungrib` or `metgrid`, then make sure you have downloaded all of the necessary atmospheric data and that the variable table and namelist are configured properly. See the WRF-ARW user's guide and the user's forum at <http://forum.wrfforum.com/> for many helpful hints at using various datasets.

## 5.4 Real and Wrf

First copy or link the `met_em` files generated by `metgrid` into `test/em_real`. If the simulation is being done locally, this can be accomplished by running in `wrf-fire/WRFV3/test/em_real`

```
ln -sf ../../../../WPS/met_em* .
```

The namelist for WRF in the file `namelist.input` must now be edited to reflect the domain configured in WPS. In addition to the fire specific settings listed in Section 4.3 regarding the ideal simulation a number of other settings must be considered as listed below. See WRF-ARW user's guide for more details on these settings.

Variable	Description
<code>&amp;time_control</code>	
<code>start_xxx/end_xxx</code>	These describe the starting and ending date and time of the simulation. They must coincide with the <code>start_date/end_date</code> given in <code>namelist.wps</code> .
<code>run_xxx</code>	The amount of time to run the simulation.
<code>interval_seconds</code>	Must coincide with interval seconds from <code>namelist.wps</code> .
<code>restart_interval</code>	A restart file will be generated every x minutes. The simulation can begin from a restart file rather than <code>wrfinput</code> . This is controlled by the namelist variable 'restart'.
<code>&amp;domains</code>	All grid settings must match those given in the <code>geogrid</code> section of <code>namelist.wps</code> .
<code>num_metgrid_levels</code>	The number of vertical levels of the atmospheric data being used. This can be determined from the <code>met_em</code> files:  <code>ncdump -h met_em*   grep 'num_metgrid_levels ='</code>
<code>sr_x/sr_y</code>	Fire grid refinement. Must match that given in <code>namelist.wps</code> as <code>subgrid_ratio_x/subgrid_ratio_y</code> .
<code>p_top_requested</code>	The default is 5000, but may need to be edited if there is an error executing <code>real</code> . If so, just set this to whatever it tells you in the error message.

Once the namelist is properly configured, run the `real` executable:

```
./real.exe
```

and then run `wrf`:

```
./wrf.exe
```

Note: It is required that the simulation start time on the top level grid coincide with the start time of the atmospheric data source. However, it is possible to start lower level domains at an arbitrary time after the top level grid starts.

## 6. Geogrid Data Scripts

### 6.1 fetch\_data.py

```
fetch_data.py -d <dataset> -- <north> <south> <east> <west>
```

Downloads a dataset from the USGS server.

**<dataset>**: The dataset to download. Currently, it supports either NED for the National Elevation Dataset (1/3 second resolution) obtained from <http://seamless.usgs.gov/index.php> or LANDFIRE13 for 13 Anderson Fire Behavior Fuel Models (1 second resolution) from <http://landfire.cr.usgs.gov/viewer/>.

**<north> <south> <east> <west>**: The geographic bounds of the data requested in degrees latitude/longitude. These arguments should each be a decimal number (north hemisphere has positive latitude, western hemisphere has negative longitude). The USGS server limits the size of a single request to approximately 2 gigabytes. One should limit the bounds to as small an area as possible for a given simulation.

This script works by reverse engineering the USGS server and is extremely sensitive to any changes to the website that may occur in the future. If any problems occur during the execution of this script, we recommend you go to the USGS server and download what you need manually.

**DEPENDENCIES**: This script depends on the presence of python version 2.5+ and twill version 1.9+ (lower versions will probably work as well, but are untested). Twill is an html parser with a python API and can be obtained from <http://twill.idyll.org/>. Consult its homepage for installation instructions.

### 6.2 geogrid.py

```
geogrid.py [options] <output> <input>
```

Options:

**-c, --continuous** indicates continuous data [default]

**-a, --categorical** indicates categorical data

**-w WORDSIZE, --wordsize=WORDSIZE**

size (in bytes) of each number in the output file

[default: 2]

**-m MISSING, --missing=MISSING**

value (int) to set any missing data in the source data

[default: 65535]

`-s SCALE, --scale=SCALE`

geogrid stores data as integers scaled by some  
constant value [default: 1.0]

`-d DESC, --description=DESC`

description of the data set [default: determined from  
source data]

`-u UNITS, --units=UNITS`

physical units of the data [default: determined from  
source data]

`-f, --force`

force creation of output files even if they already  
exist [default: False]

`-S, --signed` output data is signed [default: False]

`<input>`: File containing the input data to be converted. This file can be formatted in most standard geotagged data formats, but geotiff is recommended. For a full list of supported formats on your system, see the output of `gdal-config --formats`.

`<output>`: Directory to put the converted data. This directory will contain two files, one is an index file containing the meta data as text, and one is a binary file containing the actual data. The path to this directory should be included in the GEOGRID.TBL file as an absolute path for the given dataset. See the WRF-ARW user's guide for details of this data format and the syntax of the GEOGRID.TBL file.

The options for this command default to reasonable values; however, they may require tuning for practical use.

DEPENDENCIES: This script requires python, NumPy, and GDAL with python bindings. NumPy is a numeric library for python and can be obtained from <http://numpy.scipy.org/>. GDAL is a geospatial data library used for reading geotagged files and can be obtained from <http://www.gdal.org/>.

LIMITATIONS: Currently this script can only process datasets with less than 100,000 data points in either the x or y axes. Attempting to convert a larger dataset will result in the script exiting with an error message. Also, the source data must exist as a single data set. The USGS server splits dataset larger than 250 megabytes

into multiple downloads. These files must be merged prior to use with this script. The GDAL installation comes with a utility called `gdal_merge.py` that can perform this if necessary. One should crop the source datasets only to what is necessary for a single simulation for optimal use of this script.

### 6.3 `geogrid_wrapper.py`

`geogrid_wrapper.py [-f ]:`

This script functions as a wrapper to the `geogrid` binary and the previous two python scripts. It should be run inside the WPS directory containing a compiled `geogrid.exe`. This particular script is only compatible with the version of WPS distributed with WRF-Fire. The `geogrid` source code in this version is modified only slightly to provide additional output when source data is missing. This output is parsed by the `geogrid wrapper` script to determine what additional data (if any) is needed to successfully complete the execution. It determines the extent of the domain from this output and fetches the required data using `fetch_data.py`. It then extracts the gzipped tar files that are downloaded and runs `geogrid.py` to convert the data to the `geogrid` format. It will then modify the `GEOGRID.TBL` file to point `geogrid` to the new data directory and re-execute `geogrid`. If additional data is missing, it will attempt to fetch and convert again.

Presently, this script is limited to fetching only two datasets automatically: `highres_elev` (which is obtained from the National Elevation Dataset) and `nfuel_cat` (which is obtained from LANDFIRE). If any other data is missing or a different error is encountered, the wrapper script will print the entire `geogrid` output and exit with an error.

This script will not overwrite any existing dataset (from a previous simulation) without permission. Either move the old dataset or use the `-f` flag to overwrite it.

## 7. Fire state variables

A number of array variables was added in the registry to the WRF state in order to support the fire model. They are available in the `wrfout*` files created when running WRF. All fire array variables are based at the centers of the fire grid cells. Their values in the strips at the upper end of width `sr_x` in the  $x$  direction and `sr_y` in the  $y$  direction are unused and are set to zero by WRF.

### 7.1 Fire state variables for postprocessing

The following variables can be used to interpret the fire model output.

LFN - level set function. Node  $(i,j)$  is on fire if  $LFN(i,j) \leq 0$ .

FXLONG, FXLAT - longitude and latitude of the nodes

FGRNHFX - ground heat flux from the fire ( $W/m^2$ ), averaged over the cell

FGRNQFX - ground moisture (latent heat) ( $W/m^2$ ), averaged over the cell

ZSF - elevation above sea level (m)

UF,VF - surface wind

FIRE\_AREA – approx. part of the area of the cell that is on fire, between 0 and 1

## 7.2 Other state variables of interest

U, V – horizontal component of wind velocity on the atmosphere grid, used as input in the fire model. Note that the winds are defined on a staggered grid not at cell centers. See *WRF ARW Technical Note* for version 3, pages 58-60.

## 8. Software structure

### 8.1 Coding conventions

The fire model resides in WRF physics layer. Physics layer code needs to conform to *WRF Coding Conventions*, which can be found at [http://www.mmm.ucar.edu/wrf/WG2/WRF\\_conventions.html](http://www.mmm.ucar.edu/wrf/WG2/WRF_conventions.html)

The purpose of the conventions is to produce a transparent, fast, and maintainable code that runs in parallel without any effort on the side of the programmer of the physics layer routines.

The fire code maintains the conventions as they apply to on atmospheric grids, adapts them to 2D surface-based computations, and follows analogous conventions on the fire grid.

In particular, the fire code may not maintain any variables or arrays that persist between between calls, and may not use common blocks, allocatable variables, or pointer variables. Work arrays with variable bounds may be declared only as automatic; thus, they are freed between on exit from the subroutine where they are declared. All grid-sized arrays that should persist between calls to the fire code must be created in WRF through the registry mechanism, and passed to the fire code as arguments.

In addition, the fire code may not call any WRF routines directly but only through a utility layer. This is so that the fire code can be easily run standalone or coupled with another weather code.

All variables in the fire code are based at grid centers.

Grid dimensions are passed in argument lists as

```
ifds,ifde,jfds,jfde, & ! fire domain dims  
ifms,ifme,jfms,jfme, & ! fire memory dims  
ifps,ifpe,jfps,jfpe, & ! fire patch dims (may be omitted)  
ifts,ifte,jfts,jfte, & ! fire tile dims
```

Atmosphere grid 2D variables are declared `dimension(ims:ime, jms:jme)`.

Fire grid variables are declared `dimension(ifms:ifme, jfms:jfme)`.

Loops on the fire grid are always over a tile. The index variable names, the order of the loops, and the bounds are required exactly as in the code fragment below.

```
do j=jfts,jfte
  do i=ifts,ifte
    fire_variable(i,j)=..
```

In loops that need to index more than one grid at the same time (such as computations on a submesh, or interpolation between atmosphere and fire) the index variable names must always begin with `i j`.

## 8.2 Parallel execution

In the fire code, all computational subroutines are called from a thread that services a single tile. There is no code running on a patch. Loops may update only array entries within in the tile but they may read other array entries in adjacent tiles, for example for interpolation or finite differences. The values of arrays that may be read between adjacent tiles are synchronized outside of the computational routines. Consequently, the values of a variable that was just updated may be used from an adjacent tile only in the next call to the computational subroutines, after the required synchronization was done outside.

Synchronization within a patch is by exiting the OpenMP loop. Synchronization of the values between patches is by explicit HALO calls on the required variables and with the required width. HALOs are provided by the WRF infrastructure and specified in the registry.

The overall structure of the parallelism is spread over multiple software layers, subroutines and source files. The computation is organized in stages, controlled by the value of `ifun`.

```
! the code executes on a single patch
! if distributed memory, we are one of the MPI processes

do ifun=ifun_start,ifun_end ! what to do

  if(ifun.eq.1)then ! this HALO needed before stage ifun=1
    #include "SOME_HALO.inc" ! communicate between patches
  endif

  ...
!$OMP PARALLEL DO
  do ij=1,num_tiles ! parallel loop over tiles

    if(ifun.eq.1)then ! one of the initialization stages
```

```

        call some_atmosphere_to_fire_interpolation(...)
    endif
    ...
    call sfire_model(...,ifun,...) ! call the actual model
    ! for some values of ifun, sfire_model may do nothing

    if(ifun.eq.6)then    ! fire step done
        call some_fire_to_atmosphere_computation(...)
    endif

    enddo ! end parallel loop over tiles
    ! array variables are synchronized between tiles now

endo ! end ifun loop

```

### 8.3 Software layers

The fire code is called from WRF file `dyn_em/module_first_rk_step_part1`. The output of the fire code (the heat and moisture tendencies) are stored on exit from the fire code and added to the tendencies in WRF later in a call to `update_phy_ten` from `dyn_em/module_first_rk_step_part2`

The fire code itself consists from the following files in the `phys` directory, each constituting a distinct software layer:

`module_fr_sfire_driver.F` **Fire driver** layer: Subroutines called directly from WRF. All parallelism is contained here. The rest of the code runs is called on a single tile.

`module_fr_sfire_atm.F` **Atmosphere-fire interaction** layer: routines to interface fire and the atmosphere, interpolate between fire and atmosphere.

`module_fr_sfire_model.F` **Fire model** layer: The fire model itself, callable independently of WRF. Calls the core and the physics layers. Formulated in terms of the fire grid only. Intended to be independent of particular mathematical methods used in the core layer.

`module_fr_sfire_core.F` **Core** layer: Numerical algorithms for fire propagation and fuel decay calculation. Dimensionless. Calls the physics layer for the fire spread rate.

`module_fr_sfire_phys.F` **Fire physics** layer: Physical fire spread model and associated initialization.

`module_fr_sfir_util.F` **Utilities** layer: Used by all other layers. Declares scalar switches and parameters. Contains all interpolation and other service routines that may be general in nature and could be conceivably used for multiple purposes, and interface to WRF routines such as messages and error exits. To maintain independence on WRF, this is the only layer that may call any WRF routines.

`fr_sfir_params_args.h` Include file for subroutine argument lists to pass through all arguments that are needed in the fire spread rate routine in the physics layer. Necessary to write this long argument list only once given the WRF requirement that arrays may be passed as arguments only, and not shared globally, say, as pointers. Also, the include maintains the independence of the core layer on the physics layer and the independence of the fire code on WRF.

`fr_sfir_params_decl.h` Include file with the matching declarations.

The dependencies (allowed direction of subroutine and function calls) between the layers and WRF are in the following graph.

### WRF-Fire Software Layers and Dependencies

