Documentation for `QuickFiber` Input and Output Files

Jaime E. Forero-Romero

July 1, 2015

Contents

1 Important definitions 1

2 `QuickFiber` inputs 2

3 `QuickFiber` inputs data-structure 2

4 `QuickFiber` outputs 4

4.1 File naming convention 4

4.2 File type and structure 4

4.3 Data-structure 4

4.4 Example of potentialtargetid construction 5

y This documents describes the format of the files generated by the `QuickFiber`

software.

1 Important definitions

Taken from `https://desi.lbl.gov/trac/wiki/Pipeline/FormatsAndNumbering`

- Tile: pre-defined locations on the sky. Tiles are pre-defined with a 6-digit ID. 900000 and above are reserved for calibration, commissioning, test, ancillary, and other non-DESI key project tiles, which may not even appear in a DESI tile list.
- Pointing: A specific selection of targets within a tile indicating both a pointing of the telescope and the fiber positioners.

This means that a tile may be observed multiple times with different pointings. A pointing may be observed multiple times with different exposures. If even one target changes or any target: fiber mapping changes it becomes a new pointing. A slightly different positioner location of the same targets on the same fibers is the same pointing.

2 QuickFiber inputs

The necessary inputs of QuickFiber are

- TargetDB. Database containing the information of all targets on the sky.
- ObsDB. Database constructed from the ongoing DESI observations after the data has been processed by the Spectroscopic pipeline. This DB has not been designed yet.
- SurveyTiles. File containing the positions of all the tiles to be observed. Currently this ASCII file `https://desi.lbl.gov/trac/browser/code/survey/surveyplan/trunk/data/desi-tiles-full.par`
- FiberPositions. File containing the positions of all fibers on the focal plane. Currently this ASCII file `https://desi.lbl.gov/trac/browser/code/desimodel/trunk/data/focalplane/fiberpos.txt`
- PlateScale. File containing the mapping between the radius from the center of the focal plane and the radial angle. Currently a fit from the data obtained in this ASCII file `https://desi.lbl.gov/trac/browser/code/desimodel/trunk/data/focalplane/platescale.txt`

3 QuickFiber inputs data-structure

The table of interest in the TargetDB `/project/projectdirs/desi/db` is `Target` with the following columns

- id: primary key
- cand_ID [0-]
- ra: degrees [0-360]
- dec: degrees [-90 -+90]
- priority: [0-] Larger positive numbers indicate higher priority.
- nobs: [0-] Desired number of spectroscopic observations in DESI.
- objtype: ELG, LRG, QSO, SKY, STDSTAR, GAL, OTHER

Columns of interest to QuickFiber from ObsDB will be

- cand_ID [0-]
- ra: degrees [0-360]
- dec: degrees [-90 -+90]
- fiber_ID [0:]
- positioner_ID [0:]
- spectrograph_ID [0:]
- tile_ID [1:28810]
- signaltonoise
- confirmedobjtype: ELG, LRG, QSO, SKY, STDSTAR, GAL, OTHER

SurveyTiles is an ASCII file including the following fields

- tile_ID [1:28810]
- ra: degree [0-360]
- dec: degrees [-90-+90]
- layer [1-5]
- in_desi [0,1]

FiberPositions is an ASCII file including the following fields.

- fiber_ID [0:]
- positioner_ID [0:]
- spectrograph_ID [0:]
- x: mm, position on focal plane
- y: mm, position on focal plane
- z: mm, position on focal plane

PlateScale is an ASCII file including the following fields.

- Radius: mm. Radius from the center of the focal plane.
- Theta: degrees. Radial angle.

4 QuickFiber outputs

The principal outputs of `QuickFiber` are

- FiberMap. A pointing defining which targets are on which fiber.
- PotentialFiberMap. The set of targets that can be reached by a set of fibers in a tile.

The previous two sets will be stored in the same FITS file for each tile and pointing. This means that two different assignment algorithms running on the same tile producing different pointings will store the results in different files.

4.1 File naming convention

Primary output files from `QuickFiber` will follow the naming scheme

`QuickFiber_TTTTTT_PPPPPPPPPPPP.fits`

where `TTTTTT` is the 6-digit Tile ID and `PPPPPPPPPPPP` is a 12-digit pointing ID. The pointing ID is hash value computed by adding the IDs of all the targets in the pointing and taking the last 12 digits.

4.2 File type and structure

The outputs are uncompressed FITS files with all relevant information about the code in the primary HDU header, a BIN table describing the pointing, a BIN table listing the potential targets.

HDU0	NULL	Empty
HDU1	Binary FITS table	FiberMap information
HDU2	Binary FITS table	PotentialFiberMap information

4.3 Data-structure

FiberMap is a table with the following columns.

- fiber: [0-4999]
- positioner: [0-4999]
- numtarget: [0-] Number of potential target IDs associated to each fiber.
- objtype: ELG, LRG, QSO, SKY, STDSTAR, GAL, OTHER.
- targetid: unique target identifier to get back to target selection info. Corresponds to the cand_ID in the TargetDB.
- desi_target0: 64 bit mask of targeting info.
- ra: degrees [0-360].

- dec: degrees [-90 - +90].
- xfocal_design: mm from center in positioner coordinate system
- yfocal_deisgn: mm from center in positioner coordinate system

PotentialFiberMap with only one column.

- potentialtargetid: unique target identifier to get back to target selection info. This contains all the targets that can be reached by a fiber. Corresponds to the cand_ID in the TargetDB. This concatenates the sets of available targets for each fiber.

All the arrays have the same size as the number of fibers, except potentialtargetid which has a variable size.

4.4 Example of potentialtargetid construction

Let's assume for simplicity that we have 4 fibers with IDs `[0,1,2,3]`. There are in turn 4 different sets of targets that can be reached by each fiber. These sets are `[980,203]`, `[736,102,304]`, `[0]` and `[234]` for fibers `0,1,2` and `3`, respectiveley. The index `-1` for fiber `3` means that this fiber cannot reach any target.

In this case fiber, numtarget and potentialtargetid are as follows.

- fiber: contains the array `[0,1,2,3]`.
- numtarget: contains the array `[2,3,0,1]`.
- potentialtargetid: contains the array `[980,203,736,102,304,234]`.