A software state machine for computing astronomical coordinates

Jeffrey W. Percival

Space Astronomy Laboratory, 1150 University Avenue, Madison, WI 53706

Abstract. We consider the common problem of computing apparent and topocentric places of stars for the purpose of pointing a telescope. Detailed algorithmic descriptions exist (see, for example, Kaplan et al. AJ 1989, 97, 1197). In addition, several software packages such as NOVAS (Kaplan, 1990) and Starlink's SLALIB by Patrick Wallace considerably ease the burden in building specific application programs. A few problems remain, however. Portability can be a problem, in that some real-time platforms have grudging or non-existent support for Fortran, which is the language of implementation for NOVAS and SLALIB (SLALIB in C is now available). Also, efficiency can be a problem if the subroutines try to do too much, not allowing the programmer to fragment the calculation as needed. SLALIB offers many convenient entry points, which avoids this problem, but the programmer is still left to weave the subroutines together to achieve a desired result.

We have designed a portable software state machine, written in C, for use in the WIYN Telescope Control System. The state machine is in the form of a graph, with the nodes representing coordinate states (heliocentric FK4, topocentric apparent, or galactic, for example) and the edges representing the calculations required to move between states. The programmer provides a starting state and coordinate state vector, and a desired ending state. Using the current state and desired end state, the machine marches through the graph, performing the transitions in the proper order.

This approach has several advantages. First, not only are the calculations well-defined, as they are in existing subroutine libraries, but their order of execution is embedded in the machine, rather than merely specified in documentation, removing a source of programming error. Second, each transition can be implemented exactly once, in exactly one place, while the state machine dynamically changes the order of events according to the state transition table. Third, transitions can be implemented in both directions, allowing both production and reduction of coordinates.

We present a brief overview of the state machine, including the state diagram and transition table.

1. Overview

The Telescope Pointing Machine is a software state machine that produces or reduces astronomical coordinates.

- It is coded entirely in C.
- It can perform a single calculation (i.e. precession) or it can perform a whole coordinate flow (i.e. galactic to refracted Az/El).
- It performs its calculations entirely in a 6-dimensional rectangular coordinate space (position and velocity). Spherical projections do not enter in at any point, nor is spherical trigonometry used at any point. No CPU cycles are used going back and forth between rectangular and spherical coordinate representations.

The state transition table is a two-dimensional matrix of entries, indexed by the current and destination state. Each entry specifies two data: the next transition to execute, and the state that will result. In this way, for example, precession can be specified at many points in the table, each one with a different beginning and ending state.

Note that a state table entry does *not* specify the whole calculation between the start state (row) and the destination state (column). This would require common calculations, such as precession, to appear many times in the state table and in the state machine. Rather, each state table entry specifies only the *next* calculation, as well as where to go in the table to find the next step. In this way, the state machine uses each basic subroutine only once, in one place.

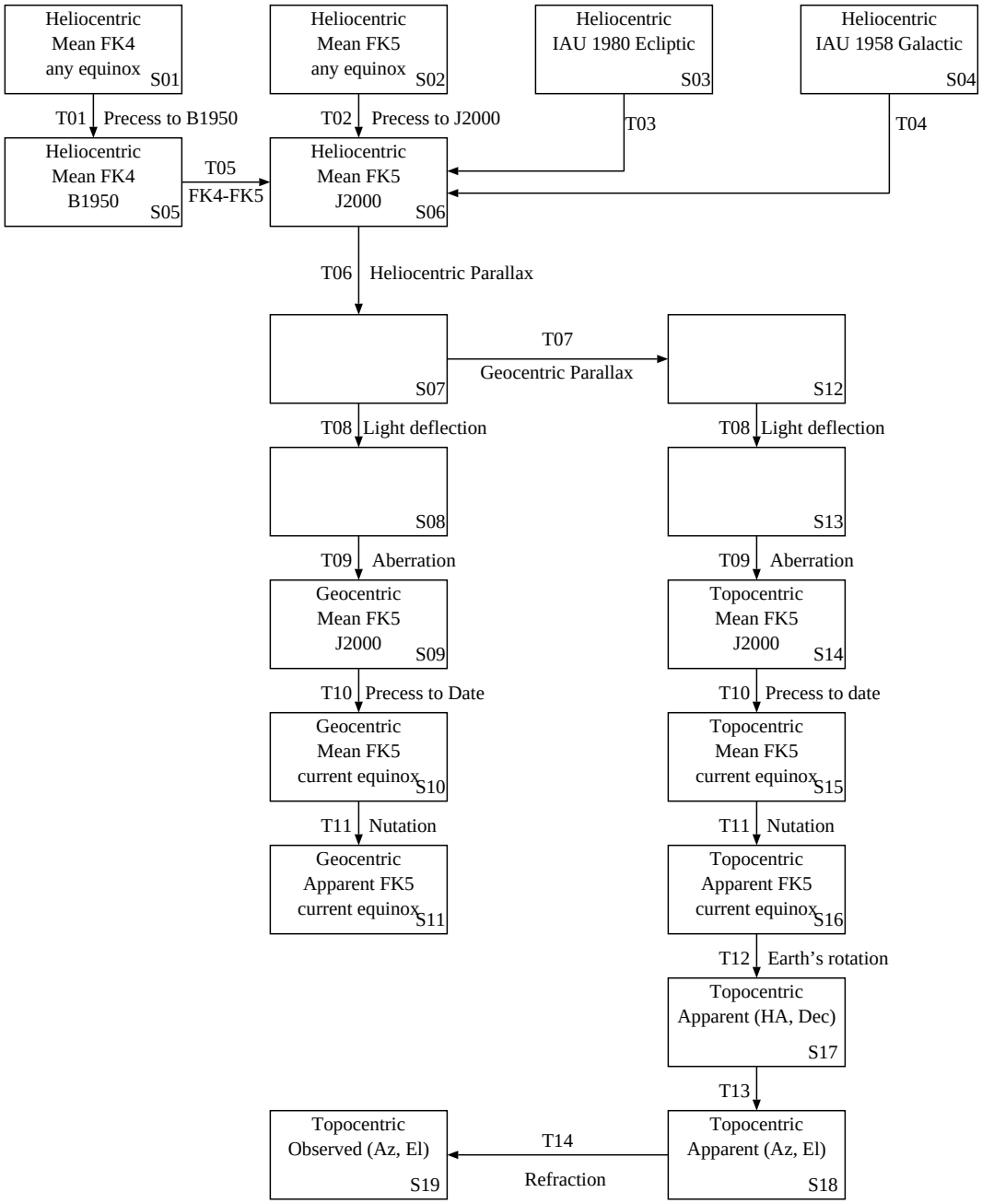


Figure 1. Telescope Pointing Machine (based on the Keck pointing flow by P. T. Wallace)

Table 1. State table (part 1) for the Telescope Pointing Machine.

start state	destination state									
	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10
S01	+T00 S01	+T01 S05	+T01 S05	+T01 S05	+T01 S05	+T01 S05	+T01 S05	+T01 S05	+T01 S05	+T01 S05
S02	+T01 S06	+T00 S02	+T01 S06	+T01 S06	+T01 S06	+T01 S06	+T01 S06	+T01 S06	+T01 S06	+T01 S06
S03	+T03 S06	+T03 S06	+T00 S03	+T03 S06	+T03 S06	+T03 S06	+T03 S06	+T03 S06	+T03 S06	+T03 S06
S04	+T04 S06	+T04 S06	+T04 S06	+T00 S04	+T04 S06	+T04 S06	+T04 S06	+T04 S06	+T04 S06	+T04 S06
S05	-T01 S01	+T02 S06	+T02 S06	+T02 S06	+T00 S05	+T02 S06	+T02 S06	+T02 S06	+T02 S06	+T02 S06
S06	-T02 S05	-T01 S02	-T03 S03	-T04 S04	-T02 S05	+T00 S06	+T05 S07	+T05 S07	+T05 S07	+T05 S07
S07	-T05 S06	-T05 S06	-T05 S06	-T05 S06	-T05 S06	-T05 S06	+T00 S07	+T07 S08	+T07 S08	+T07 S08
S08	-T07 S07	-T07 S07	-T07 S07	-T07 S07	-T07 S07	-T07 S07	-T07 S07	+T00 S08	+T08 S09	+T08 S09
S09	-T08 S08	-T08 S08	-T08 S08	-T08 S08	-T08 S08	-T08 S08	-T08 S08	-T08 S08	+T00 S09	+T01 S10
S10	-T01 S09	-T01 S09	-T01 S09	-T01 S09	-T01 S09	-T01 S09	-T01 S09	-T01 S09	-T01 S09	+T00 S10
S11	-T09 S10	-T09 S10	-T09 S10	-T09 S10	-T09 S10	-T09 S10	-T09 S10	-T09 S10	-T09 S10	-T09 S10
S12	-T06 S07	-T06 S07	-T06 S07	-T06 S07	-T06 S07	-T06 S07	-T06 S07	-T06 S07	-T06 S07	-T06 S07
S13	-T07 S12	-T07 S12	-T07 S12	-T07 S12	-T07 S12	-T07 S12	-T07 S12	-T07 S12	-T07 S12	-T07 S12
S14	-T08 S13	-T08 S13	-T08 S13	-T08 S13	-T08 S13	-T08 S13	-T08 S13	-T08 S13	-T08 S13	-T08 S13
S15	-T01 S14	-T01 S14	-T01 S14	-T01 S14	-T01 S14	-T01 S14	-T01 S14	-T01 S14	-T01 S14	-T01 S14
S16	-T09 S15	-T09 S15	-T09 S15	-T09 S15	-T09 S15	-T09 S15	-T09 S15	-T09 S15	-T09 S15	-T09 S15
S17	-T10 S16	-T10 S16	-T10 S16	-T10 S16	-T10 S16	-T10 S16	-T10 S16	-T10 S16	-T10 S16	-T10 S16
S18	-T11 S17	-T11 S17	-T11 S17	-T11 S17	-T11 S17	-T11 S17	-T11 S17	-T11 S17	-T11 S17	-T11 S17
S19	-T12 S18	-T12 S18	-T12 S18	-T12 S18	-T12 S18	-T12 S18	-T12 S18	-T12 S18	-T12 S18	-T12 S18