ENGI 6806 ECE Project Design Lab
Judging Software Interface

Dennis K. Peters* Siu O'Young

August 17, 2007

The 6806 Rally Judging software is used to help ensure that all rally teams are judged fairly and timed accurately and consistently. The software has an internal model of the arena and the route that the robots are expected to take, and uses the system clock to keep track of the elapsed times as required. This document describes the interface that the judging software presents to the team software. Through this interface the team software can

- Receive the initial go signal, with starting point and first waypoint.
- Query if the robot has achieved a waypoint.
- Report the next waypoint, as interpreted from the images on the waypoint marker, and receive confirmation that it is/isn't correct.
- Report that it has reached the end of the route.

1 Protocol Description

All communications between the team software and the judge software takes place over a TCP/IP socket on port 23400. The judge software will open a socket server on this port and accept one connection from the team software. Information is exchanged in the form of messages as described below.

1. All messages consist of ASCII encoded text using the following characters only:
   - Upper and lower case letters (a-z, A-Z).
   - Digits (0-9).
   - The following special characters: space (0x20), newline (0x10), dash (0x2d) and underscore (0x5f).
2. All messages end in a newline character ('`\n`' in C++ or Java.[1])

*Initial version with input from Michael Bruce-Lockhart

[1]Note the Java `println` method and the C++ `endl` operator are **not** suitable means to append the newline character.

3. All waypoints are identified by their name, which is a single capital letter.

4. The judge software does not initiate any exchange, but rather only responds to requests sent to it, so the protocol proceeds in a strict request-response fashion. The rally team software must send a message and then wait for a response. It is an error to send a new message before the response is received.

The grammar for the protocol is given in Figure 1, using the following conventions.

- Items in `typewriter font` are keywords (language terminals).
- Case (upper/lower) is always significant.
- The symbol $\diamond$ is used to represent the space character (0x20).
- The symbol $\hookleftarrow$ represents the newline character (0x10, '`\n`').
- Items in *italic font* are symbols in the grammar.
- *t2j*(M) indicates that M is sent by the team software to the judge, whereas *j2t*(M) indicates that M is sent by the judge software to the team.
- The notation $\{M\}_n^m$ represents between n and m repetitions of M.
- *Start* is the start symbol for the grammar.

Note that the grammar in Figure 1 does not express the conditions under which each response will be given nor does it enforce the limit on the number of incorrect `to` or `atEnd` messages. These are described in the following section.

2 States

As illustrated in Figure 2, the following are the observable states of the judge software.

Init Judge waits for the team software to connect and announce itself with "`Rally2007` *name*", where *name* is the single word team name not exceeding 25 characters. It responds with "`Rally2007`".

Ready Judge waits for "`ready`" from the team software, indicating that it is ready, then issues the "`go` *wp1 wp2*" command, where *wp1* is the starting location for the robot and *wp2* is the first waypoint in the route. This is the only point in the protocol where there may be a significant delay before the judge software responds (the judge may be waiting for operator input indicating that the arena is ready). The game clock is started when the `go` message is sent.

Moving Moving to a waypoint. When the team software believes that the robot is in the vicinity of the waypoint it sends the "`queryAt` *wp*" message, where *wp* is the name of the waypoint being sought. The response is "`yes`" if the waypoint has been achieved, or "`no`" if not. Note that if *wp* is not the correct waypoint then the response will always be "`no`".

$$
\begin{aligned}
\mathit{Start} &\rightarrow\ t2j(\texttt{Rally2007} \diamond \mathit{Name} \hookleftarrow)\ j2t(\texttt{Rally2007} \hookleftarrow)\ \mathit{Ready} \\
\mathit{Ready} &\rightarrow\ t2j(\texttt{ready}\ \hookleftarrow)\ j2t(\texttt{go} \diamond \mathit{wp} \diamond \mathit{wp} \hookleftarrow)\ \mathit{Moving} \\
\mathit{Moving} &\rightarrow\ t2j(\texttt{queryAt} \diamond \mathit{wp} \hookleftarrow)\mathit{QueryResp} \\
\mathit{QueryResp} &\rightarrow\ j2t(\texttt{yes} \hookleftarrow)\ \mathit{Acquiring} \\
&\quad |\ j2t(\texttt{no} \hookleftarrow)\ \mathit{Moving} \\
\mathit{Acquiring} &\rightarrow\ t2j(\texttt{to} \diamond \mathit{wp} \hookleftarrow)\ \mathit{AcquiringResp} \\
&\quad |\ t2j(\texttt{atEnd} \hookleftarrow)\ \mathit{EndResp} \\
\mathit{AcquiringResp} &\rightarrow\ j2t(\texttt{yes} \hookleftarrow)\ \mathit{Moving} \\
&\quad |\ j2t(\texttt{no} \hookleftarrow)\ \mathit{Acquiring} \\
&\quad |\ j2t(\texttt{no} \hookleftarrow)\ \mathit{Terminate} \\
\mathit{EndResp} &\rightarrow\ j2t(\texttt{yes} \hookleftarrow)\ \mathit{Terminate} \\
&\quad |\ j2t(\texttt{no} \hookleftarrow)\ \mathit{Acquiring} \\
&\quad |\ j2t(\texttt{no} \hookleftarrow)\ \mathit{Terminate} \\
\mathit{Terminate} &\rightarrow \\
\mathit{wp} &\rightarrow [\texttt{A} - \texttt{Z}] \\
\mathit{Name} &\rightarrow \{\mathit{AlphaNum}\}_1^{25} \\
\mathit{AlphaNum} &\rightarrow [\texttt{a} - \texttt{z}] \mid [\texttt{A} - \texttt{Z}] \mid [\texttt{0} - \texttt{9}] \mid \texttt{-} \mid \_
\end{aligned}
$$

Figure 1: Protocol Grammar

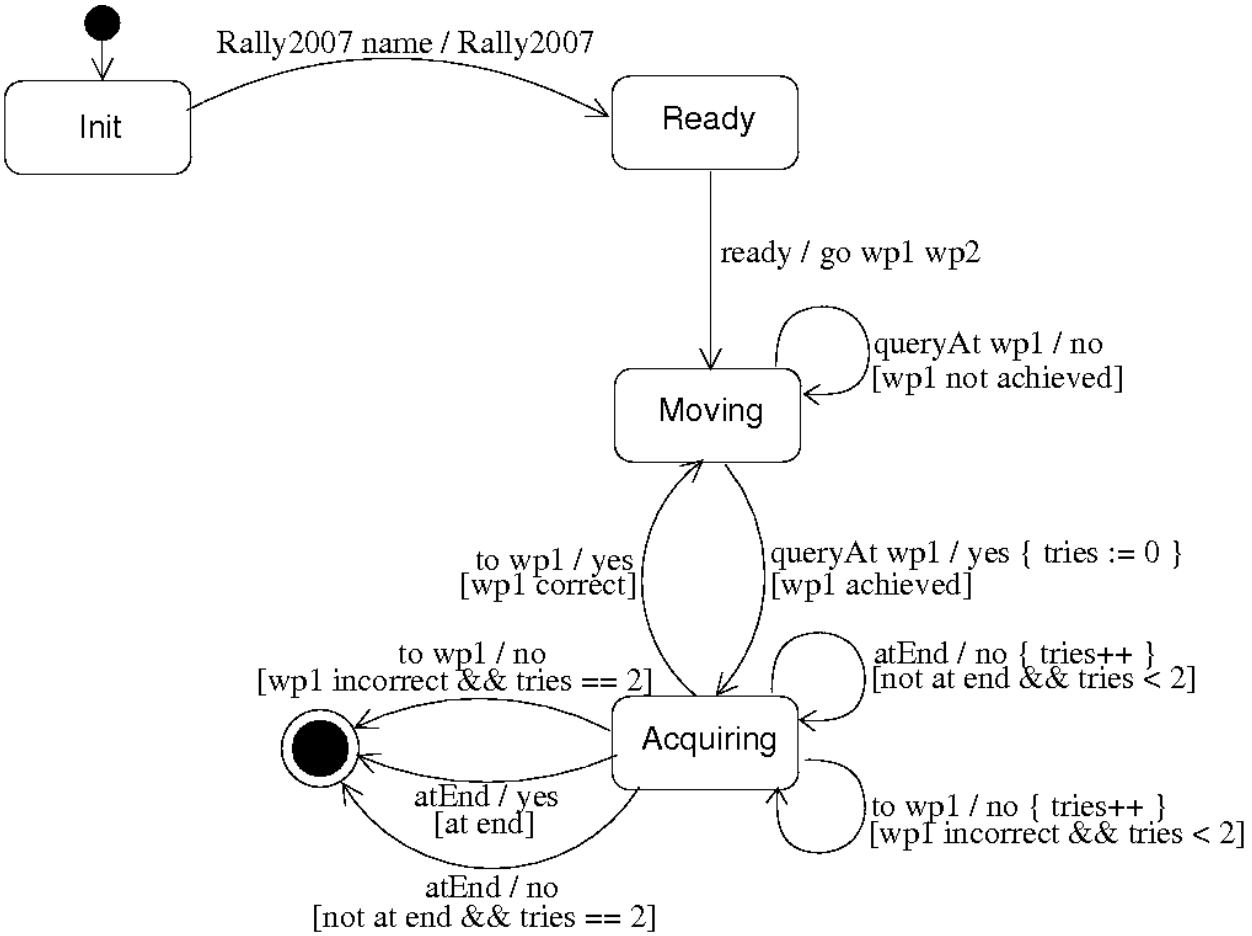

Figure 2: Judge State Machine

Acquiring Robot is acquiring and processing the image to determine the next waypoint. The team software sends "`to` *wp*", indicating that it believes *wp* is the next waypoint, or "`atEnd`", indicating that it believes that the final waypoint has been reached. The response is "`yes`" if correct, or "`no`" otherwise. A "`yes`" response to "`atEnd`" or three consecutive "`no`" responses will cause the judge software to terminate the run and enter the End state.

End Upon entering the final state the game clock is stopped and the team score is calculated based on elapsed time and the number of waypoints achieved.