Andrew Palardy
FRC #33 – The Killer Bees
Autonomous Scripting System

BeeScript – Autonomous Scripting System Basics

Introduction:

The BeeScript autonomous scripting system is designed to take a script file, written in a simple text editor such as Notepad, and execute the commands contained. It it's most primitive form, it is a LabVIEW command interpreter, running through a text file line by line. In my use, I wrote autonomous scripts to be executed during autonomous mode, allowing me to make minor changes to the autonomous routines without re-compiling all of the robot code. BeeScript is written entirely in LabVIEW.

Usage:

BeeScript is called during autonomous mode in Autonomous_Thread.vi. In the Robot Main.vi, you can replace the existing static reference to Autonomous Independent.vi with a static ref to Autonomous_Thread.vi, by dropping the Autonomous_Thread.vi into the static ref.

Inside of Autonomous_Thread, there isn't much at all. A global variable defines which autonomous slot to execute (and is presumably set during the Disabled period, elsewhere in your robot code), and the system finds the filename in the array. All files are assumed to be in the \ni-rt\startup directory, but you can change the directory in read_auto_file.vi. There are three other calls in Autonomous_Thread.vi

One calls CreateCommands, where you must register all of the autonomous commands you have created (see the DRIVE_STRAIGHT and ELEV_SET_STATE examples for how to create command functions, and CreateCommands for how to register them). All commands are strictly typed, and must conform to the standard set by TEMPLATE.vi. I recommend that you open the template and save it as your new function, to guarantee that the VI conforms to the standard.

Another call is to Auto_Initialization. This is where you, for example, reset your gyro before entering automode.

The third call calls “ExecuteFile”, which is the actual interpreter itself.

Commands:

All commands follow a specific connector pattern defined in TEMPLATE.vi. This is strictly typed throught the software. All commands are fed the number of arguments and an array of string arguments by the interpreter. They can use them as they choose. DriveStraight shows an example of driving for distance at speed, and how to use two strings as numbers. Elev_Set_State shows how to lookup an enumerated state in a table of strings, useful for mapping a string input with an enumerated output. All commands must be registered in CreateCommands.vi.

Script files:

Each command must have it's own line. Comments start with #. Comments must have their own line, they cannot be typed after commands. Only commands registered in CreateCommands can be called, if a command does not exist it will be skipped. Blank lines will also be skipped.