**Architecture**

The flow of data through our system mirrors our implementation architecture closely. Knowing that iRay had to handle both the older ERG and newer BDM memory access protocols, we wanted to minimize cross-component coupling. It is straightforward to extend iRay with new widgets via the Lua scripting language, new memory access interfaces via a C++ API, and new debug formats as external tools. A small number of CSV schemas describe our application interchange formats, meaning that these files can also be automatically generated and processed by tools like Excel. This modularity also enabled the developers to divide up work easily without interfering with each other.

**Process**

Given our frequent client and team communication and iRay's well-defined scope, we opted to pursue a lightweight implementation process. We considered bugs and features alike as Trac items, and treated Trac as a combination of to-do list, bug reporter, and documentation facility. In the more volatile stages of our project, we worked solely through meetings, email, and IRC, electing not to take the extra time to create and manage Trac tickets. Our small and motivated team worked well with this process flexibility and we fulfilled our scope as determined at the project’s halfway point.

**Future Expansion**

- **Folding** - The ability to fold a segment of the graph down in order to directly compare segments of the graph that are separated by a large distance in time.
- **Bookmarks** - The ability to bookmark a specific point in the grapher and quickly return to it.
- **Play back overwrite target** - The ability to use iRay in reverse, and stream the data into the target processor having it use the values from iRay instead of the actual memory values.
- **Manipulation of graph, and write-back to TRC file** - The ability to manipulate the graph and the values of the signals and then replay them to a trace file, or to the target hardware.
- **Configurable key bindings.**
- **Able to have two or more Grapher windows.**
- **Custom dashboards** - The ability to create custom dashboard displays using custom widgets.
- **Custom widgets** - Custom ways of displaying signal information.

**Overview**

**Motivation/Background Information**

iRay is a PC-based tool that provides a dynamic, interactive, readily-comprehensible “x-ray” view into the operation of an embedded microcontroller’s program, by monitoring and displaying certain interesting memory variables. iRay will enable the dynamic collection and saving of the internal memory state of an embedded microcontroller over time.