DAS3 SVN Organization

Goals and Discussion

1. In the past we were project based. Every project got it's own code base. To start a new project, you picked up the code from the closest old project, copied it to the new project, and started hacking. This can be inefficient and error prone. It's like we are always starting from scratch more or less.
2. Updates to the code for sensors in one project, never made it back to old projects. John started to fix this with the creation of the /path/sens directory where he's been putting the code for our commonly used sensors. I think we want to try to use this strategy of grouping common code elsewhere, not just for sensors and utility functions.
3. Backwards compatibility is non-existent. If we start hacking on a car for CICAS, there's a good chance that we'll break a bunch of things for NT. When we need to resurrect a demo, it's a lot of work. However, I'm not sure that anything we're going to do is going to fix that, but we might be able to make it better.
4. Right now we have 4 cars, all with the same hardware/software, but pretty soon, those cars are going to start diverging. It would be nice to somehow keep the cars similar enough that they can be easily swapped between projects. However, this sort of means keeping the cars up-to-date on all current and recently finished projects. At a certain point, there's going to have to be shedding of backwards compatibility.
5. Right now, we do have some advantages being project oriented. When a project ends, you have all the stable code stored in the repository, and you don't need to bother with legacy when you move on to the next project. You don't even need to download that old project's code.

Directory Structure	**Usage**
/das3	• All the code common to a specific generation of DAS. I numbered it 3 because I consider this our 3rd generation DAS, where the DAS is some combination of computers & vehicles & software. If somehow the system, through the years, diverges too much, making for backwards compatibility between experiments impossible, you could just add a /das4.
/doc /src	• System docs, cases, motherboards, etc. • Common code for most /das3 systems
/startup	• This is where vehicle startup scripts go.
/ui /sounds /src /twd	• Base Vehicle User Interface Directory • Sounds • Tilcon Source Code • Tilcon .twd Resources
/veh /vehicletype /can /src /test	• Code and start-up scripts that are vehicle-specific, such as the code needed to read the vehicle's CAN data and put it into the data hub. The startup script is going to point to a vehicle-based startup script in the test directory.
/path /db /local /sens /tilcon	• Common code to almost all projects. • Data Hub Code • Utility Functions • Various Frequently Used Sensors • PATH Tilcon Base Code
/project /veh	Each project would be like ntmm, cicas, etc. Depending on the scope of the projects, you may or may not need a /veh directory layer. Or, you may need multiple /veh layers if you had multiple vehicles that ran vastly different code. The point is that this is project-specific code.
/src /test	• Experiment-specific code, wrtfiles, etc. • Compiled apps and project startup scripts
/ui /sounds /src /twd	• Project-Specific User Interface Directory • Sounds • Tilcon Source Code • Tilcon .twd Resources

How to Handle Write Files

Wrtfiles I the most complicated program to handle in the move, and the main points of the discussion include the following:

- There is a lot of commonality between experiments. Wrtfiles still needs to write a-files, d-files based on the vehicle you are on, and it needs to synch with the video computer. We don't want to keep copying all this functionality for each new experiment.
- However, we also don't want the wrtfiles code in the DAS3 directory to explode in size/complexity, or even to have to change with every new experiment.

I think that Tom and I came up with a way to handle wrtfiles in the reorganization that satisfies everyone's concerns. Basically, we are thinking that in the /das3 directory structure, you could break out the common parts of wrtfiles into a library. It would know how to write an a-file and a d-file for each vehicle that is supported in the /das3 code base, and it would know how to synch with the video computer, etc.

Then, in the project directories, there would be the wrtfiles program and executable. This would be the program called by each project, and it would be where the switches were read and handled to specify if you want an a-file, which vehicle d-file you want, maybe even what vehicle ID letter to use, and what project specific files to write.

This will probably be more work to break apart wrtfiles to function this way, but it will handle both of our concerns. It's less code to write/copy over for each new project, and you can still switch a vehicle's experimental without recompiling (by changing the start-up scripts). However, the wrtfiles code in the /das3 directory won't grow, or even change, unless you add new vehicle types. Also, you won't have an issue if say, 3 cars are in development for CICAS, and a fourth cars is in development for another project. You only have to have the code for the projects you want to run on a car checked out and compiled.