



Uart Code

General Structure

Uart code is written such that multiple uarts share a common set of functions with a unique structure passed into the general task code that differentiates between uarts. Each uart in the system is managed by its own task.

Uart tasks can use interrupt code to coordinate with the task, or the uarts can be managed by timer-tick calls.

Application interface to the uarts is done with the familiar Posix functions. In this case the file handle identifies the uart. For instance,

```
fprintf( stdout, "var_num = %d\n", var_num );
```

sends output to the uart designated as stdout.

Functions included are:

- fgets . . . get string;
- fputs . . . put string;
- fwrite . . . write N bytes;
- fprintf . . . output using format string;
- fgetc . . . get a character;
- fputc . . . put a character;

Terminal Program

A sample terminal task is supplied with the uart code which can be used to communicate with another device or PC.

Task Monitor, Tmon

The task monitor uses one of the uarts to communicate with the system. It is not necessary, and can be omitted when the uart is needed by the real application.

Tmon has a number of functions to view task stacks, task control blocks and peek/poke memory. Task status can be determined, and if blocked, what the blocking condition is.

Memory poke can be used to satisfy a task waiting on a memory location to change, which allows fine grained task control from a terminal. Via this mechanism, a task can be run once and memory examined for results.

Dormant tasks can be invoked through tmon, and tasks can also be killed.

An application test function can be called through tmon, which is useful for debugging a function under controlled circumstances. Another common use of the test function is to print a structure or general memory.

Tmon can also be used to send mail to a given task as a way to test the recipients handling of the mail.