Power Management Software for MOOS
Functional Requirements

Tom O'Reilly, Bob Herlien
5/12/2003
Version 1.3

1.0 Introduction

The requirements for power management for MOOS can be categorized and classified in several dimensions. In the following requirements, we use two primary classifications. Mechanism versus Policy follows the traditional Unix idea of separating "what needs to be done" versus "how that capability is used". System versus Application is intended, in this context, to distinguish things that need to be accomplished at the Linux kernel and/or driver level (system) versus the SIAM code (application). In each section below, we define our terms and then list the requirements.

2.0 Mechanism

As noted above, this refers to the "what"; in particular, to the ability to put the CPU and/or instruments into low-power mode.

2.1 System

The SideArm CPU has three supported power modes: "running", "idle", and "sleeping".

2.1.1 The SideArm power management software **SHALL** support "running" and "sleeping" power modes.

2.1.2 The SideArm hardware and system software **SHALL** support mechanisms to wake the CPU from "sleeping" mode to "running" mode on the following events:

- Wake on elapsed time
- Wake on received character from a UART
- Wake on received packet from MMC Communications Controller
- Wake on Ethernet packet ???

2.1.3 The SideArm system code (Linux drivers) **SHALL** support Linux power management interface (pm_register(), etc) for all appropriate drivers, including UARTs (?), Ethernet (?), ….

2.2 Application

2.2.1 The SIAM code **SHALL** support methods or interfaces to enable the SIAM code to place the system into the "sleeping" power mode, and to successfully return to the "running" power mode.

2.2.2 The SIAM code **SHALL** support methods or interfaces to enable the SIAM code to power instruments on or off as appropriate.

3.0 Policy

"Policy" for the power management code refers to the ability to know when it's appropriate to put the processor into sleep mode, when it's appropriate to go into run mode, and how to manage the power consumption of attached instruments.

In some cases, turning the SideARM off is *not* an acceptable power management solution. The node should not be turned off if it is required to respond to an external event in a time interval which is shorter than the time required to restore full "running" condition. Also, if it is desired to preserve the state of node applications (including SIAM), turning the SideARM off is very problematic.

Before putting the SideARM into a low power state, power management must determine the status of activities on the node. Some activities are predictable from onboard schedules (e.g. instrument sample acquisition), while other activities are not (e.g. offboard client access of the node services, event detection). "Critical" activities must not be disrupted by power management, unless the node is in danger of completely exhausting its power. Power management must never cause the node's file system to be left in an irrecoverable inconsistent state.

When restoring the SideARM state to "running" from a low power state, it is required that the system time be restored to nominal "correct" value.

3.1 System

This section refers to Linux power management policy. It is not acceptable to put the processor to sleep while clients (onboard or offboard) are accessing Linux services; e.g. if telnetd, ftpd, or nfsd are servicing clients. At this writing, it is unknown whether Linux either has or enforces such policies. The proposed approach, in the spirit of rapid prototyping, is to implement the application power management policies, and then see whether this disturbs correct functioning of the underlying Linux system.

3.2 Application – Static Power Management

This refers to the ability of the SIAM software to determine whether, from the perspective of the SIAM application framework software and its attached instrument drivers, it is correct to put the CPU into sleep mode; and to manage the power status of the instruments. The "static" part means that, in this mode, the SIAM software will power-manage the instruments according to their declared power policy. It is assumed that an engineer or technician, when configuring the system, has determined that sufficient power is available for the attached instrument suite. At this level, the SIAM software does not attempt to measure battery state or otherwise determine that the power is actually available.

3.2.1 The SIAM software **SHALL** present a power interface to its attached instrument suite that allows each instrument to declare its power policy as one of POWER_ALWAYS, POWER_NEVER, or POWER_WHEN_SAMPLING.

3.2.2 The SIAM software **SHALL** include interfaces to its attached instrument suite and to the remainder of the SIAM application framework that will allow it to determine whether it is appropriate to enter sleep mode. Furthermore, it will ensure that the system will exit sleep mode at the appropriate point in time or on receiving an event that requires servicing.

3.3 Application – Power Prediction and Adaptive Power Management

Onboard power management will include capability to predict power availability and loads as a function of time, based on readings from power subsystem, time of day, instrument sampling and communication schedules, etc. This capability will enable power management to make autonomous decisions regarding rationality of device usage schedules.

3.3.1 Appropriate interfaces to power subsystem **SHALL** be implemented; voltage/current reading of each onboard power source (solar, battery, wind turbine, etc) will be available to node power management software through this interface.

3.3.2 Mechanisms to prioritize energy usage between system functions and among instruments **SHALL** be designed and implemented.

3.3.3 Interfaces to the subsystem that detects external events of scientific interest (event detection), predicts power requirements for responding to these events, and allocates that power, **SHALL** be designed and implemented.

4.0 Network Power Management

TBD

Do we have a requirement to manage all nodes' power from a centralized point? Do we need to respond to requests to power down, reboot, sleep, stay awake, etc?