Derived Software Requirements for the MiniROV Light Controller

Derived Software Requirements

1. Support MODBUS RTU/ASCII at standard baud rates.
2. Control Light power (as coils) and dimming (through holding registers).
3. Control two other switches (as coils).
4. Report GF Current, Temperature, and 0-5VDC Channels as input registers.
5. Must respond to a hardware set modbus address.
6. Report a seawater leak as a discrete input (2 separate sensors).

General Control Philosophy

Initially this firmware will target a reliable device that reflects the control emplaced upon it through Modbus interactions. Behaving like an industrial controller, no control loops will execute and only protection-specific conditionals will limit the controls. Since the Modbus protocol implements a master slave relationship, this philosophy is well suited to the primary means of communications chosen.

Modbus Specification

Coils	Address	Function
	1	LIGHT_CNTL_0
	2	LIGHT_CNTL_1
	3	LIGHT_CNTL_2
	4	LIGHT_CNTL_3
	5	LIGHT_CNTL_4
	6	LIGHT_CNTL_5
	7	INSTR_CNTL_0
	8	INSTR_CNTL_1
	9	DIGITAL_OUT_0
	10	DIGITAL_OUT_1
	11	GF_CNTL_HIGH
	12	GF_CNTL_LOW

Discrete Input	Address	Function
	10001	DIGITAL_IN_0
	10002	DIGITAL_IN_1
	10003	H2O_SENSE_0
	10004	H2O_SENSE_1
	10005	CN_IN_0
	10006	CN_IN_1

Input Register	Address	Function
	30001	ANALOG_IN_0
	30002	ANALOG_IN_1
	30003	ANALOG_IN_2
	30004	ANALOG_IN_3
	30005	AMBIENT_TEMP
	30006	GF_CURR

Holding Register	Address	Function
	40001	LED_DAC_0
	40002	LED_DAC_1
	40003	LED_DAC_2
	40004	LED_DAC_3
	40005	LED_DAC_4
	40006	LED_DAC_5
	40007	DAC_OUT_6
	40008	DAC_OUT_7

Functions

Light Power and Dimming Control

Light power will be controlled through a coil write from the Modbus master. The output voltage signal attached to the dimming input of the light will be maintained by the DAC electronics on the coil's corresponding LED_DAC[X] register (12-bit scale value for 0-5VDC). Output of the dimming voltage signal will not be conditional upon coil state. Default settings of the power and voltage outputs are set for off and zero volts.

There will also be a limit on how often power can be switched to these lights to help prevent ground-bounce. If a command to turn on/off light arrives before this timeout, the command will return a Modbus Error code.

Instrument Power Control

There are two high side switches that will be controlled by the two coils, INSTR_CNTL_0 and INSTR_CNTL_1.

Ground Fault Detection

The ground fault detection circuit can test either the high or low side of the input power. The user must enable the test through a coil write to GF_CNTL_HIGH or GF_CNTL_LOW. These tests are mutually exclusive so a write to one of the two coils will disable the other. These will be left on until a coil write is invoked to disable them. The value of these tests can be read on the GF_CURR input register. The read will trigger a discrete ADC conversion, so one of the two coils must be enable to read an accurate value. [Potential Feature: *create a user defined level of fault and automatically disable any switch output after detection.*]

Water Sensing

The board has connections for two continuity sensors. By issuing a discrete input request to either H20_SENSE_0 or H20_SENSE_1, the controller will report the status of continuity as a 1 (closed) or 0 (open) for either sensor. [Potential Feature: *Read of these pins will occur as well on a repeated basis in in the main loop of the firmware, and any detection of water will latch the value reported as closed, until a coil write is issued to clear the alarm.]*

Temperature Sensor and Analog Input Read

By issuing a read request for the input registers 30001 through 30006, the return will include the 10-bit scale value of the of the voltage input channel. For the temperature sensor, full range is 0 to 100°C (see datasheet for exact curve) and other channels scale from 0 to 5VDC.

Auxiliary Analog Output

There are two spare channels from the DAC. Issue a 12-bit value to either the DAC_OUT_6 or DAC_OUT_7 holding register and the voltage will output on a 0-5VDC scale.

Configuration

The pic will reserve a section of non-volatile memory for use to store configuration parameters. Factory defaults have yet to be chosen. The settings will be modified in a combination of hardware methods and communications. All setting changes will take place after power reset.

Modbus Hardware Configuration

The Modbus connection to the microcontroller has been attached through a multi-protocol transceiver capable of either RS232 or RS485/422. Decision of **protocol** will be made via jumper, and the PIC will monitor this setting in order to drive the transmitter as needed.

The Modbus **address** of the controller will also be set in a hardware manner (yet to be identified). Likely this will take place as a 3-bit input to the micro, providing a range of 8 addresses. The address range requires determination.

Modbus Software Configuration

The remaining Modbus parameters that require configuration will be modified through console access on one of the two auxiliary serial ports. Those items will be configurable as a special escape sequence that will generate a menu prompt. Relevant Modbus configuration parameters are:

- MODBUS ASCII or RTU
- MODBUS serial port settings. {BAUD,BITS,PARITY,STOP}

Auxiliary Sensor Configuration

Alongside the high-side switches for two auxiliary sensors are two spare serial ports. Those ports must possess flexible configuration as well, but we also must pass on a way to parse their data and report it to via Modbus. At a minimum we must provide a flexible method for setting the serial port configuration parameters:

- AUX1 serial port settings. {BAUD,BITS,PARITY,STOP}
- AUX2 serial port settings. {BAUD,BITS,PARITY,STOP}

Without knowledge of what the sensors that may be attached are, we must limit these sensors at this time to ones whom report their data via ASCII string. That string must be somehow character-delimited, and have a definitive end-character. All of this will be configured by passing a standard precision format string that a string parsing function such as SSCANF could ingest over the console serial interface.

Since the slave cannot pre-empt the master in the Modbus configuration, the controller will have to only report the most recent value to the master when requested. Though not planned, there could be a coil associated with new data on these communication channels.

Other Considerations

Bootloader

It is possible to put a serial port bootloader on the PIC24F, and will be considered but not made a high priority, as performing a bootload operation over multi-drop RS485 will likely proved difficult given the timeline of this project.

Watchdog Timer

This could be easy to employ, but there is little reason to reset the device during operation since there will be upstream power control.