

Digi-Key / Freescale Design Contest Entry

Automated Carnivorous Fish Feeding System Contest Entry by Brad Hanken

Design Implementation

Overview

As described in my original contest submission, my entry uses the MC9S12NE64 at the core of a flexible, modular, adaptable feeding solution that would meet the needs of any keeper of large or carnivorous fish. I have successfully designed a main control unit that houses the MC9S12NE64 microcontroller and a simple character LCD module based UI for initial TCP/IP setup when DHCP is unavailable, as well as for manual feeder configuration and status display. I decided to use an encoder switch assembly to simplify the front panel interface. Menu navigation on the control unit is a simple matter of turning and clicking the encoder wheel to select, or clicking the back button on the opposite side of the LCD to back out. By reducing the number of interface elements, I've made the system easier to use.

The control unit is capable of feeding fish in a variety of ways. The unit alone is capable of driving a stepper motor controller I also designed during this build phase of the contest, to turn a large-capacity dry pellet food auger to meet dry food feeding requirements. It can also mate up with up to six slave modules. For the contest, I designed a live food feeding module capable of being configured with up to two live food bays. It incorporates a simple button and LED user interface for manual operation. In addition, since each slave port incorporates a flexible SPI interface along with 3.3, 5, and 12 volt supplies, the design could be expanded to support a wide variety of other peripheral devices, such as water monitoring systems, filtering systems, etc. During operation, the control unit polls any external modules for device capabilities. The modules then respond with their capabilities and status. This modular approach allows a single system architecture to accommodate a variety of system requirements. A user could begin with only the control unit, and enjoy the convenience of an automated dry food feeder. Later, the system could be tailored to the aquarium's expanding needs by adding additional dry food feeding modules, multi-compartment live food modules, large-capacity single-compartment live food modules, or even additional control units, each with an optional combination of additional feeding modules. Additional peripherals could later be designed, enhancing the utility of the system.

Design Implementation

Hardware Design – Control Unit

The original control unit concept consisted of four main parts: the PCB housing the MC9S12NE64 and associated circuitry, the front panel housing our LCD, LEDs, and buttons, the electromechanical subsystem for releasing dry food, and the external connections for power and communications. In my actual design, however, I combined all central subsystems onto the main PCB except for the stepper motor driver.

During the design of the board, I tried to make it as flexible and multi-purpose as possible. The generic expansion port configuration and simple UI are examples of decisions to that end. Although not used or required for this design, I also reserved pads on the PCB for level converters and headers for the SCI ports.

The main purpose of the control unit is to dispense dry food and coordinate the dispensation of dry and live food via other feeding modules. In order to accomplish this, it also needs to keep time, store feeding schedules, and interact with users through both a simple LCD and pushbutton interface, and through the network in a predefined TCP/IP based protocol the desktop UI can access. These tasks, and the manner in which I implemented them, are outlined in the table below.

Task	Implementation
Dispense dry food	Connect general purpose I/O from the MC9S12NE64 to a stepper motor driver through a buffer. This drives a stepper motor connected to an auger, which is capable of dispensing large dry food pellets.
Keep Time	Poll a DS1305 RTC via SPI.
Store frequently updated variables	Utilize the onboard non-volatile ram onboard the DS1305.
Store Feeding Schedules	Utilize a 2432 type I2C EEPROM.
Front panel UI	I created a simple front panel UI by interfacing the NE64 to a parallel 16X2 LCD module through a buffer. Backlight control is via a solid-state relay, and contrast control is accomplished via a WMS7201 non-volatile SPI digital pot. Menu selection is accomplished through an encoder switch assembly connected to GP I/O. A back switch via GP I/O completes the UI.
Communicate with desktop UI	Use the MC9S12NE64's built-in ethernet controller.
Power	I used the LM2595 switching 3.3V regulator, and a surface-mount variant of the 7805 for 5V regulation.

Hardware Design – Stepper Motor Controller

In order to create a complete, working design, I needed to be able to drive my auger stepper motor. To this end, I created the submitted stepper motor controller. It is based on the L297/L298 combination of stepper motor driver chips. My design is an adaptation of the circuit described in the datasheet for the L297 and L298. I took the basic design from the datasheet, and made a physical interface suitable for interfacing with microcontrollers for a variety of needs. Although not utilized in this project, I made several signals available that make the board particularly flexible. For example, the reference voltage that drives the chopper current limiting circuit is accessible, and could be driven by a digital pot or other D/A mechanism for more control of the motor behavior.

Software Design – Control Unit

Upon initialization, the microcontroller will first check for stored settings in the EEPROM and NVRAM in the DS1305. If the current data appears invalid, the system will first set all settings to the default values of no food schedule, and DHCP for IP setup. Once settings have been loaded, the IP stack supplied by the OpenTCP project is initialized. Since the control unit must also be able to run without a network connection to be useful, the program enters the main loop without waiting for the network to come up. Inside the main loop, however, the TCP stack is checked regularly and is initialized when the network is available. Once network connectivity is established, the feeder can be controlled from the external network as well as the front panel. Other activities monitored in the main loop are as follows:

Task	Logic
Data Received from IP Stack (Interrupt Driven)	Several commands are supported: <ul style="list-style-type: none"> • Set RTC (send time / date) • Query schedule • Update schedule • Query capabilities
Monitor User Input	The GP I/O connected to the encoder and select and back switches are checked inside the main loop. When input is detected, events are passed into a menu system, which processes the input and displays the corresponding output.
Update Display	During normal operation, the display is constantly updated with the current date and time.

Hardware Design – Feeding Module

The feeding modules consist of the MC68HC908QB4 SPI capable microcontroller, standard servos to dispense the food, and an extremely basic button and LED based UI. The MC68HC908QB4 was a great choice for this module. Though the unit is a low pin-count device, it possessed all essential internal peripherals required for this module.

Because of the limited requirements of a feeding module and the presence of regulated power from the control module, the PCB is very simple, consisting only of the microcontroller and a few passive components. The UI is integrated into the PCB, and consists only of a button and LED indicator to toggle the presence of food in the feeder, as well as a manual release button to manually request that the system opens or closes the feeding bay or flake feeder. I used a single A/D pin to read as many buttons as needed, in this case six. (I used the technique described in AN1775.) I used the extra I/O freed up by using this technique to drive the extra servos required to make a dual live feeder.

Since servos sometimes draw extra current at the extreme edges of their range, I utilized a solid-state relay to cut current to the servos when not needed. The mechanical design needed to release the food to the aquarium is such that constant correction should not be required.

Software Design – Feeding Module

Upon initialization, the microcontroller will wait for a command to be received from the control unit. The control unit stores state information for all feeding modules in non-volatile ram in the DS1305, and so is responsible for sending initialization commands to each feeding module.

Once initialized, the microcontroller will await one of the following interrupts:

Event	Logic
Data Received from SPI Interface	Several commands will be supported: <ul style="list-style-type: none"> • Open / Close live food bay • Actuate flake food release servo • Set occupied status • Query occupied status • Query capabilities
Periodic Timer Interrupt 1	Actuate flake food release servo if required by current state.
Periodic Timer Interrupt 2	Service servo PWM signaling
Open/Close Button Actuated	Open bay if closed, close bay if opened. Toggle status LED to reflect state.
Occupied Status Button Actuated	Toggle status LED to reflect state. Store status in memory and report if requested by control unit.
Release Flake Food Button Actuated	Actuate associated flake food servo.

Software Design – Scheduling Software

The scheduling software was written in Java, and compiled using GCJ. Due to time limitations, I've written a simple command line interface for scheduling user events. When no arguments are passed, the program lists all valid commands and their syntax. Arguments are available for checking feeder module availability, checking the schedule, as well as adding or deleting scheduled feeding events.

Though the interface is simple, I have created base APIs which could be used as the basis for a more complex GUI interface.

Conclusion

I have successfully created a working implementation of my MC9S12NE64 based feed control system. The microcontroller delivered as advertised, and truly did open up the world of network connectivity to this consumer device. In the process of developing this project, I was constantly seeing ways that this could be expanded. Though the build phase for this contest may be over, I plan on continuing to work with this controller to see where else it could lead me. I plan on exploring the possibilities of the generic expansion ports I've built in. Water temperature and quality monitoring may be a logical direction. Also, the board I've designed is generic enough that it could control a wide variety of devices. I anticipate that my home will soon be full of network enabled appliances.