Recap

Timer
Debugging
Intrusiveness
Monitors and dumps

Overview

Finite state machine

8.7. Finite state machines with statically-allocated linked structures

8.7.1. Abstraction

Software abstraction
define a problem with a set of basic abstract principles
separate policies mechanisms

Finite State Machine (FSM.)
inputs, outputs, states, and state transitions
state graph defines relationships of inputs and outputs

The three advantages of this abstraction are
1) it can be faster to develop
2) it is easier to debug (prove correct) and
3) it is easier to change

What is a state?
Description of current conditions

What is a state graph?
Graphical interconnection between states

What is a controller?
Software that inputs, outputs, changes state
Accesses the state graph

What is a finite state machine?
Input sensors
Output actuators
Controller
State graph

Moore FSM
output value depends only on the current state, and
inputs affect the state transitions
significance is being in a state
input: when to change state
output: how to be in that state
Mealy FSM depend both on the current state and the inputs.
- output value depends on input and current state
- inputs affect the state transitions.
- significance is the state transition

input: when to change state
output: how to change state

Moore: Output needed to be in that state
Mealy: Output needed to cause a state change

data structure embodies the FSM
- multiple identically-structured nodes
- statically-allocated fixed-size linked structures
- one-to-one mapping FSM state graph and linked structure
- one structure for each state

linked structure
- pointer (or link) to other nodes (define next states)

table structure
- indices to other nodes (define next states)
Stepper motor controller

This stepper motor FSM has two input signals four outputs.

Many hardware circuits in this class will be drawn with a free drawing tool
PCBArtist, http://www.4pcb.com/
1) Design- data flow graph, flowchart, pseudocode
2) Implement in TExaS, debug it
3) Switch to Real mode
4) place 9S12 in LOAD mode
   Cable from PC to docking module
   Power applied to embedded system, reset button
   Execute Assemble to download
   Run from debugger, 24 MHz
5) place in RUN mode
   Power applied to embedded system, reset button
   It’s running at 8 MHz

Write in 9S12C32 assembly
;*********************************************************************************************
PTAD    equ $0270 ; Port AD I/O Register
DDRAD   equ $0272 ; Port AD Data Direction Register
ATDDIEN equ $008D ; ATD Input Enable Mask Register
DDRM    equ $0252 ; Port M Data Direction Register
PTM     equ $0250 ; Port M I/O Register
DDRT    equ $0242 ; Port T Data Direction Register
PTT     equ $0240 ; Port T I/O Register
TCNT    equ $0044 ; Timer Count Register

Jonathan W. Valvano
TSCR1 equ $0046 ; Timer System Control Register1
TSCR2 equ $004D ; Timer System Control Register 2
org $3800 ; Globals go in 2K Ram
delay ds.w 1 ; number of cycles to wait
start ds.w 1 ; TCNT value at the start of wait
Pt rmb 2 ; pointer to current state

org $4000
out equ 0 ; 8-bit output
wait equ 1 ; time to wait, 32us units
next equ 3 ; 4 pointers to next state
S5 fcb $05 ; 4-bit output
fdb 4000
fdb S5,S9,S6,S5 ; next for each in
S6 fcb $06
fdb 4000
fdb S6,S5,S10,S6
S10 fcb $0A
fdb 4000
fdb S10,S6,S9,S10
S9 fcb $09
fdb 4000
fdb S9,S10,S5,S9

* ROM program
Main lds #$4000
bsr Timer_Init ; activate TCNT
bset DlRT,#$03 ; PT1 PT0 output to LEDs
bset ATDDIEN,#$C0 ; PAD6,7 digital
bclr DDRM,#$C0 ; PAD6,7 input
movb #$05,PTM ; initial output
movw #S5,Pt ; initial state
cli ; allow debugger

loop ldx Pt
movb out,X,PTM ; step motor
ldd wait,X
bsr Timer_Wait ; wait specified time
ldaa PTAD ; read inputs (negative logic)
eora #$C0 ; positive logic
anda #$C0 ; just CCW,CW
    ; 0,40,80,C0
    ; 0,20,40,60
    ; 0,10,20,30
    ; 0,08,10,18
    ; 0,04,08,0C
    ; 0,02,04,06
leax next,X ; list of pointers
ldx A,X ; next depends on in
stx Pt
ldaa PTT
eora #$01
staa PTT ; heart beat
bra loop

Run in simulator, scan point on PTM output
Run on 9S12C32
Debugging example (running the 9S12C32 stepper program)

Quit debugger
Close all windows
Open stepper.uc
Execute assemble to download

In Real-Time Debugger
Click the green Go arrow (press just one button at a time)
Press just PAD6 (in=10), rotates CW
Press just PAD7 (in=01), rotates CCW
What happens if you press both switches?
Quit debugger

Add a debugging instrument called a dump or a scan
Add to RAM
Dump rmb 1000 ; place for 500 scans
DumpPt rmb 2 ; where to store next
Add to initialization code
ldx #Dump
stx DumpPt ; buffer is empty
Add debugging code to end of the loop (right before bra loop)
The place we add this dump/scan is called a ScanPoint
ldx DumpPt ; pointer to buffer
cpx #Dump+1000 ; skip if full
beq skip
ldy Pt ; data to record
sty 2,x+ ; record it in buffer
stx DumpPt ; update pointer
skip

Execute assemble to download
In Real-Time Debugger
Set memory address to $3800
Resize memory window to make it big
Set the format to 16-bit hexadecimal
Run and push a button quickly
Halt
Observe Dump, see the state changes

Reset the software
Set mode to Periodical
Run and push a button quickly
Observe Dump, see the state changes

The bottom line
Finite state machines provide for abstraction
Simple controller, complicated state graph
Simulation (testing), prototype (test), final system (test)