

**Recap**

**Timer**  
**Finite state machine**

**Overview**

**Debugging**  
**Intrusiveness**  
**Monitors and dumps**

**Intrusiveness**

degree of perturbation caused by the debugging itself  
 how much the debugging slows down execution

**Nonintrusive**

characteristic or quality of a debugger  
 allows system to operate as if debugger did not exist  
 e.g., logic analyzer, ICE, BDM

**Minimally intrusive**

negligible effect on the system being debugged  
 e.g., dumps (ScanPoint) and monitors

**highly intrusive**

e.g., print statements, breakpoints and single-stepping

**Dump or ScanPoint**

**Question:** Is my program outputting to the motor?

**Method:** Add instruments, assemble, download, run

**Initialization**

`dbuf rmb 100 first 100 outputs to stepper`

`dpt rmb 2 pointer to next place`

`ldx #dbuf`

`stx dpt`

`ldaa #100`

`dloop clr 1,x+`

`dbne A,dloop`

or

`ldy #dbuf`

`sty dpt`

`dloop clr 1,y+`

`cpx #dbuf+100`

`bne dloop`

`bra *`

**Debugging Instrument**

`org *`

`dscan pshx`

`ldx dpt`

`cpx #dbuf+100`

`bhs ddone`

`movb PTT,1,x+`

`stx dpt`

`pulx`

`ddone rts`

`nop`

`dsize equ *-dscan`

**Is it Intrusive??**

1) count cycles

<code>\$F05E</code>		<code>org *</code>
<code>\$F05E 34</code>	<code>[ 2]( 0)</code>	<code>dscan pshx</code>
<code>\$F05F FE0800</code>	<code>[ 3]( 2)</code>	<code>ldx dpt</code>
<code>\$F062 8E0866</code>	<code>[ 2]( 5)</code>	<code>cpx #dbuf+100</code>
<code>\$F065 2409</code>	<code>[ 3]( 7)</code>	<code>bhs ddone</code>

```

$F067 1809300024    [ 5]( 10)    movb PTT,1,x+
$F06C 7E0800       [ 3]( 15)    stx  dpt
$F06F 30           [ 3]( 18)    pulx
$F070 3D           [ 5]( 21)ddone rts
$F071 A7           [ 1]( 26)    nop
$0014                                     dsize equ *-dscan

```

### 2) TCNT measurement

```

test1  movb #$80,TSCR1
       movw TCNT,first
       jsr  dscan
       ldd  TCNT
       subd first
       std  delay

```

### 3) Attach scope to unused output, PT7

```

test2  bset DDRT,#$80
tloop  ldx  #dbuf
       stx  dpt
       bset PTT,#$80
       jsr  dscan
       bclr PTT,#$80
       bra tloop

```

### Second 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

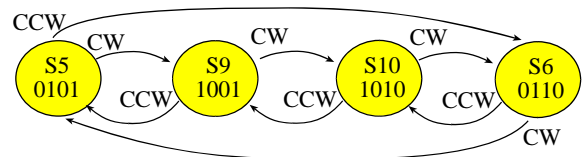
```

**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**

**Monitor is a real-time visualization**  
**Dump records data for later analysis**  
**Simulation (testing), prototype (test), final system (test)**