

CMPEN 472 Homework 2, Blinking LED on HC12C128 Simulation

The screenshot shows the Freescale CodeWarrior IDE interface with the project file "cmpen472hw2b_choi.mcp" open. The main window displays the assembly code for "main.asm". The code is a demonstration program for the HC12C128 microcontroller, titled "LED Light Blinking". It includes comments detailing the program's purpose, hardware connections, and simulation parameters. The assembly code defines symbols for I/O ports (PORTA, PORTB, DDR), memory addresses, and registers (Counter1, Counter2). It implements a delay loop using nested loops and toggles four LEDs (LED 1-4) via PORTB bit 4-7. The code is annotated with extensive comments explaining its functionality.

```
*****  
* Title: LED Light Blinking  
* Objective: CMPEN 472 Homework 2 in-class-room demonstration program  
* Revision: V3.2 for CodeWarrior 5.2 Debugger Simulation  
* Date: Aug. 26, 2020  
* Jan. 25, 2021  
* Programmer: Kyusun Choi  
* Company: The Pennsylvania State University  
* Department of Computer Science and Engineering  
* Algorithm: Simple Parallel I/O use and time delay-loop demo  
* Register use: A: LED Light on/off state and Switch 1 on/off state  
* X,Y: Delay loop counters  
* Memory use: RAM Locations from $3000 for data,  
* RAM Locations from $3100 for program  
* Input: Parameters hard-coded in the program - PORTB  
* Switch 1 at PORTB bit 0  
* Switch 2 at PORTB bit 1  
* Switch 3 at PORTB bit 2  
* Switch 4 at PORTB bit 3  
* Output: LED 1 at PORTB bit 4  
* LED 2 at PORTB bit 5  
* LED 3 at PORTB bit 6  
* LED 4 at PORTB bit 7  
* Observation: This is a program that blinks LEDs and blinking period can be changed with the delay loop counter value.  
* Note: All Homework programs MUST have comments similar to this Homework 2 program. So, please use those comment format for all your subsequent CMPEN472 Homework programs.  
* Adding more explanations and comments help you and others to understand your program later.  
* Comments: This program is developed and simulated using CodeWarrior development software and targeted for Axion Manufacturing's CSM-12C128 board running at 24MHz.  
*****  
* Parameter Declaration Section  
*  
* Export Symbols  
    XDEF pstart ; export 'pstart' symbol  
    ABSENTRY pstart ; for assembly entry point  
* Symbols and Macros  
PORTA EQU $0000 ; i/o port A addresses  
DDR A EQU $0002  
PORTB EQU $0001 ; i/o port B addresses  
DDRB EQU $0003  
*****  
* Data Section: address used [ $3000 to $3OFF ] RAM memory  
*  
    ORG $3000 ; Reserved RAM memory starting address  
; for Data for CMPEN 472 class  
Counter1 DC.W $0100 ; X register count number for time delay  
; inner loop for msec  
Counter2 DC.W $00BF ; Y register count number for time delay  
; outer loop for sec  
; Remaining data memory space for stack,  
; up to program memory start  
*  
*****  
* Program Section: address used [ $3100 to $3FFF ] RAM memory  
*  
    pstart ORG $3100 ; Program start address, in RAM  
    LDS #$3100 ; initialize the stack pointer  
;  
    LDAA #\$11110000 ; LED 1,2,3,4 at PORTB bit 4,5,6,7 FOR CSM-12C128 board  
    LDAA #\$11111111 ; LED 1,2,3,4 at PORTB bit 4,5,6,7 FOR Simulation only  
    STAA DDBR ; set PORTB bit 4,5,6,7 as output  
;  
    LDAA #\$00000000  
    STAA PORTB ; Turn off LED 1,2,3,4 (all bits in PORTB, for simulation)  
mainLoop BSET PORTB,%10000000 ; Turn ON LED 4 at PORTB bit 7  
*****
```

Freescale CodeWarrior - [main.asm]

File Edit View Search Project Processor Expert Device Initialization Window Help

cmplen472hw2b_choi.mcp

Full Chip Simulation

Files Link Order Targets

```

        ORG      $3000 ; Reserved RAM memory starting address
        ; for Data for CMLEN 472 class
Counter1 DC.W    $0100 ; X register count number for time delay
Counter2 DC.W    $00BF ; Y register count number for time delay
                    ; inner loop for msec
                    ; outer loop for sec

                    ; Remaining data memory space for stack,
                    ; up to program memory start

*
***** Program Section: address used [ $3100 to $3FFF ] RAM memory
*
pstart   ORG      $3100 ; Program start address, in RAM
        LDS      #$3100 ; initialize the stack pointer

        ; LDAA    #<11110000 ; LED 1,2,3,4 at PORTB bit 4,5,6,7 FOR CSM-12C128 board
        ; LDAA    #<11111111 ; LED 1,2,3,4 at PORTB bit 4,5,6,7 FOR Simulation only
        ; STAA    DDRB    ; set PORTB bit 4,5,6,7 as output

        LDAA    #<00000000
        STAA    PORTB   ; Turn off LED 1,2,3,4 (all bits in PORTB, for simulation)

mainLoop BSET    PORTB,<10000000 ; Turn ON LED 4 at PORTB bit 7
        JSR     delay1sec ; Wait for 1 second

        BCLR    PORTB,<10000000 ; Turn off LED 4 at PORTB bit 7
        JSR     delay1sec ; Wait for 1 second

        LDAA    PORTB   ; read switch 1 at PORTB bit 0
        ANDA    #<00000001
        BNE     sw1pushed ; check to see if it is pushed

        sw1notpsh BCLR    PORTB,<00010000 ; turn OFF LED 1 at PORTB bit 4
        BRA     mainLoop

        sw1pushed BSET    PORTB,<00010000 ; turn ON LED 1 at PORTB bit 4
        BRA     mainLoop

*****
* Subroutine Section: address used [ $3100 to $3FFF ] RAM memory
*
; delay1sec subroutine
; Please be sure to include your comments here!
;

delay1sec PSHY    ; save Y
        LDY     Counter2 ; long delay by

dly1Loop JSR     delayMS ; total time delay = Y * delayMS
        DEV
        BNE     dly1Loop

        PULY    ; restore Y
        RTS    ; return

*****
; delayMS subroutine
; This subroutine cause few msec. delay
;
; Input: a 16bit count number in 'Counter1'
; Output: time delay cpu cycle wasted
; Registers in use: X register as counter
; Memory locations in use: a 16bit input number at 'Counter1'
;
; Comments: one can add more NOP instructions to lengthen
; the delay time.

delayMS  PSHX    ; save X
        LDX     Counter1 ; short delay

dlyMSLoop NOP    ; total time delay = X * NOP
        DEX
        BNE     dlyMSLoop

        PULX    ; restore X
        RTS    ; return

*
* Add any subroutines here
*

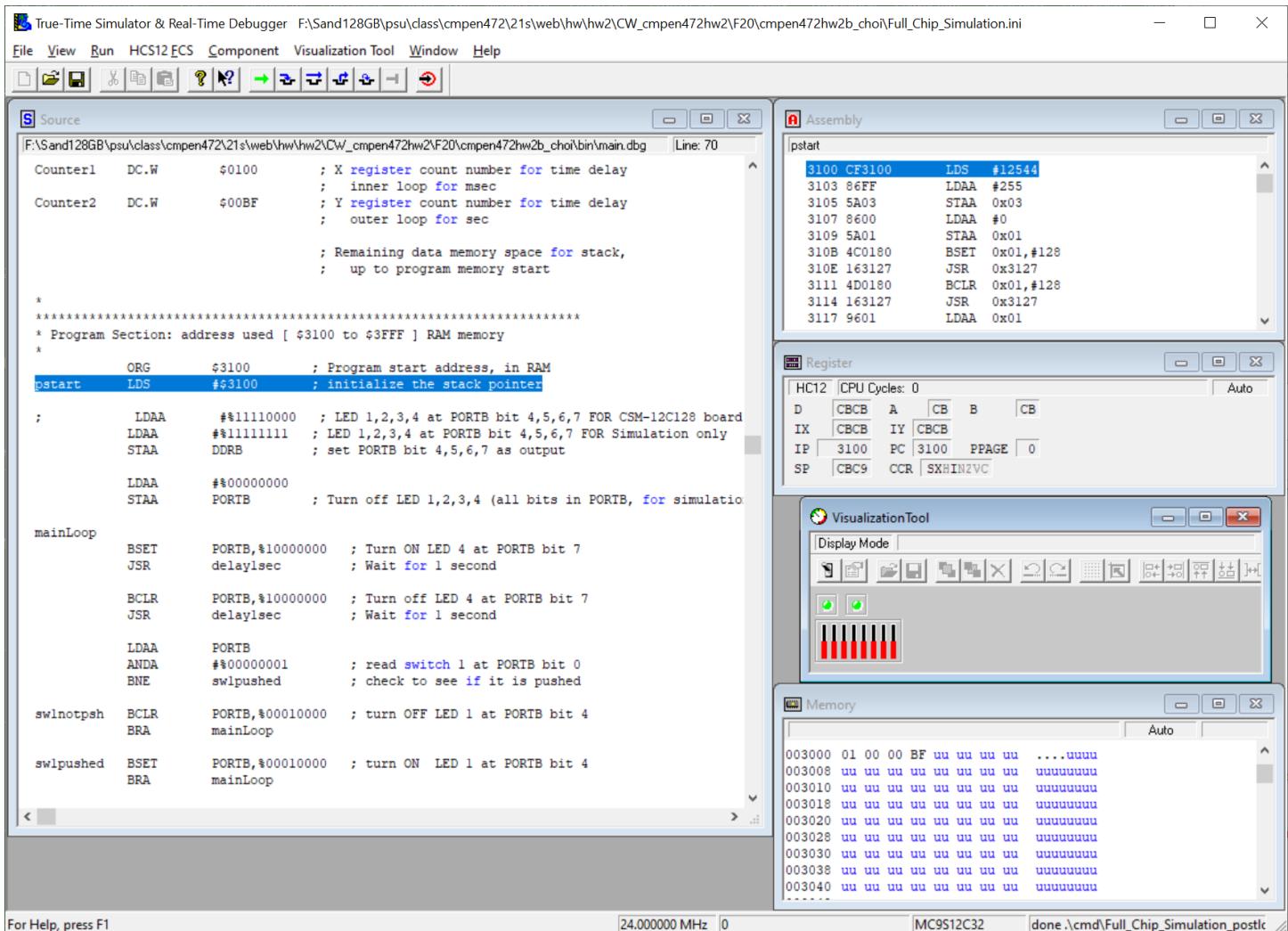
        end      ;last line of a file

```

4 files 65 0

Line 38 Col 66

Save Make Debug



HCS12_FCS: Clock Frequency 24MHz

Component: VisualizationTool

VisualizationTool: Properties

VisualizationTool: Properties: Refresh Mode = CPU Cycles, Cycle Refresh Count = 1

VisualizationTool: Add New Instrument = LED

VisualizationTool: LED: Properties:

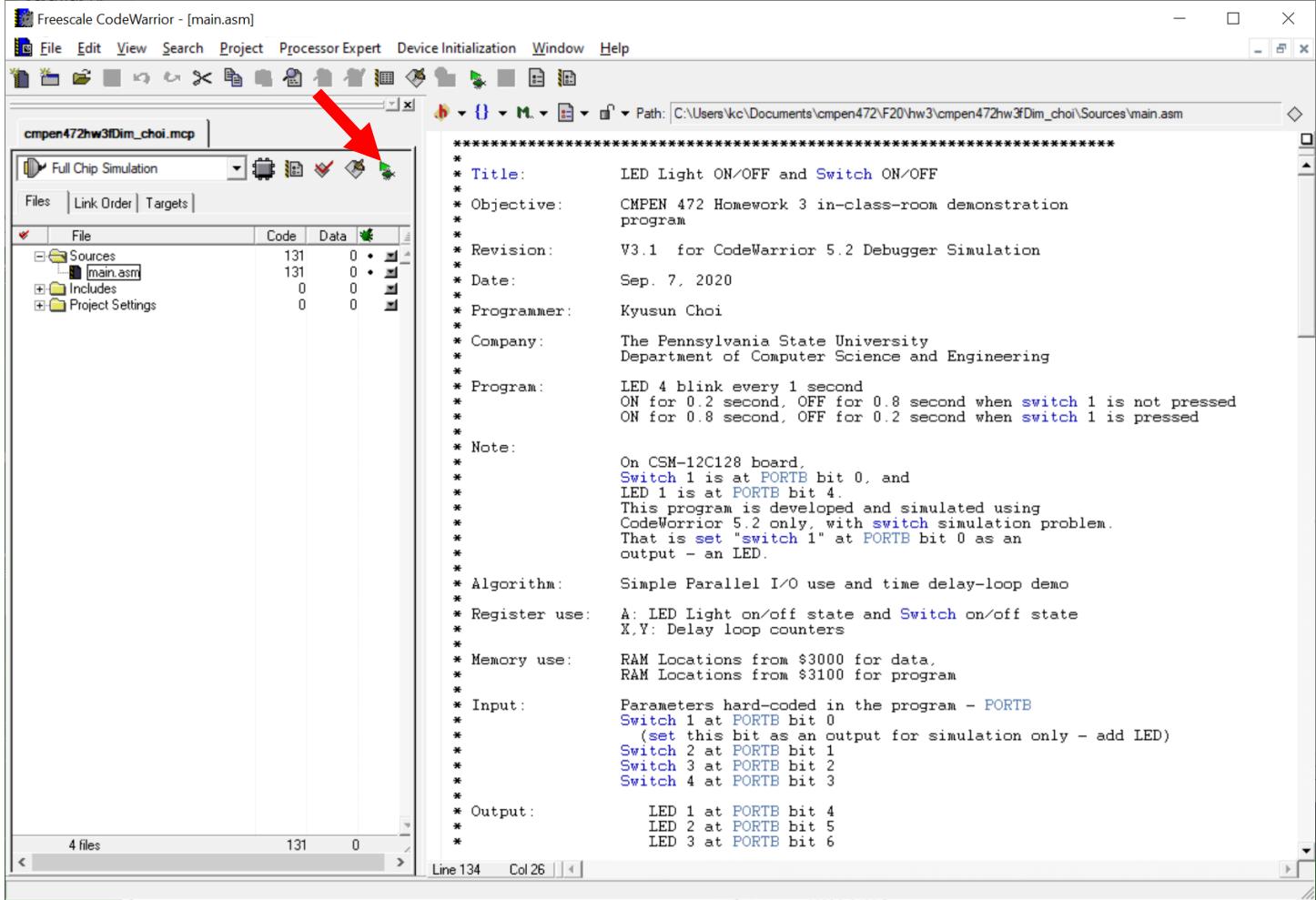
Set Port to Display to 0x0001 PORTB
 Set Size 1 Byte 8bit port
 Set Byte Order to BigEndian
 Set Bit Number to Display to 4 LED 1

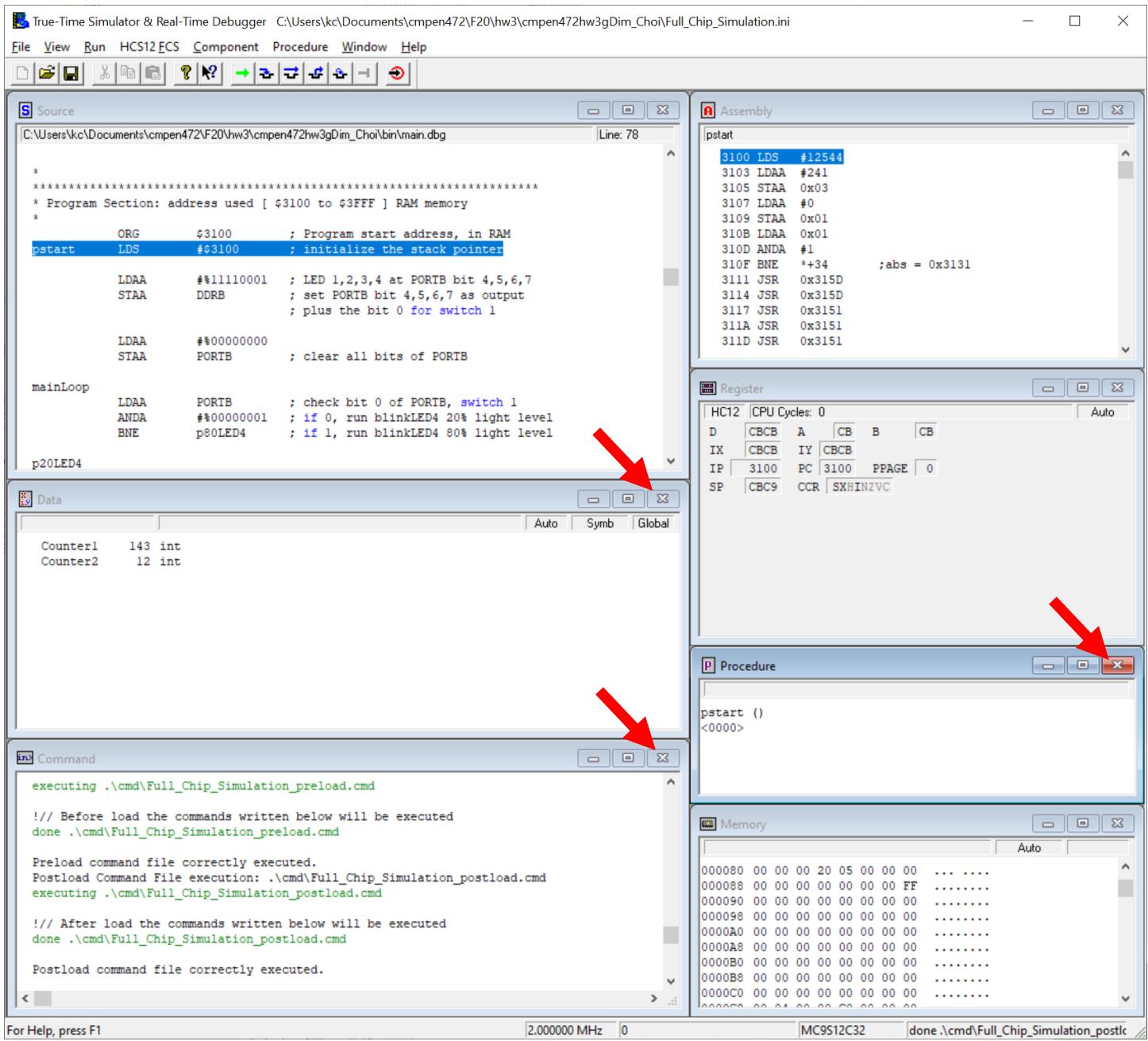
VisualizationTool: Save Layout As cmpen472hw2_choi.vtl

Source
 Assembly
 Register
 VisualizationTool
 Memory

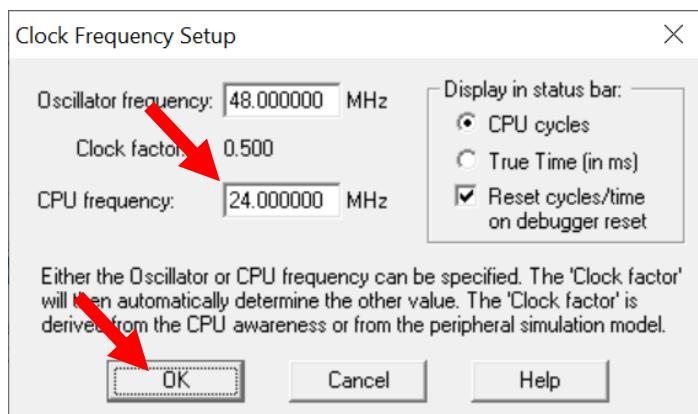
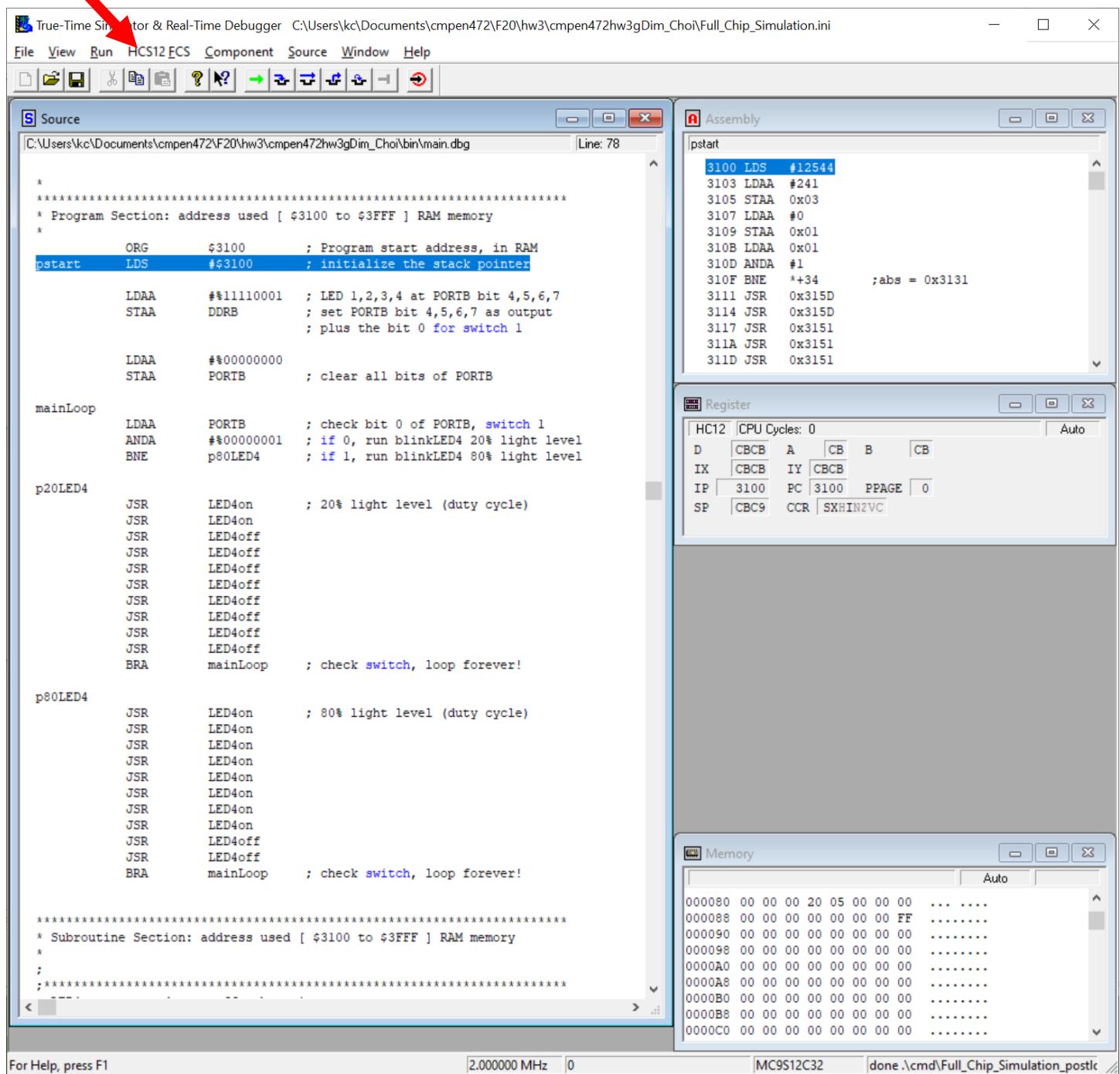
File: Save Configuration

Start CodeWarrior Debugger/Simulator – pictorial instruction





Click HC12FCS, and select "Clock Frequency . . . "



True-Time Simulator & Real-Time Debugger C:\Users\kc\Documents\cmpen472\F20\hw3\cmpen472hw3gDim_Cho\Full_Chip_Simulation.ini

File View Run HCS12FCS Component Source Window Help

S Source C:\Users\kc\Documents\cmpen472\F20\hw3\cmpen472hw3gDim_Cho\bin\main.dbg Line: 78

```

*
***** Program Section: address used [ $3100 to $3FFF ] RAM memory
*
    ORG      $3100      ; Program start address, in RAM
pstart LDS      #$3100      ; initialize the stack pointer

    LDAA     #\$11110001      ; LED 1,2,3,4 at PORTB bit 4,5,6,7
    STAA     DDRB      ; set PORTB bit 4,5,6,7 as output
                  ; plus the bit 0 for switch 1

    LDAA     #\$00000000
    STAA     PORTB      ; clear all bits of PORTB

mainLoop
    LDAA     PORTB      ; check bit 0 of PORTB, switch 1
    ANDA     #\$00000001      ; if 0, run blinkLED4 20% light level
    BNE     p80LED4      ; if 1, run blinkLED4 80% light level

p20LED4
    JSR      LED4on      ; 20% light level (duty cycle)
    JSR      LED4on
    JSR      LED4off
    BRA     mainLoop      ; check switch, loop forever!

p80LED4
    JSR      LED4on      ; 80% light level (duty cycle)
    JSR      LED4on
    JSR      LED4on
    JSR      LED4on
    JSR      LED4on
    JSR      LED4on
    JSR      LED4on
    JSR      LED4off
    JSR      LED4off
    BRA     mainLoop      ; check switch, loop forever!

*****
* Subroutine Section: address used [ $3100 to $3FFF ] RAM memory
*
;
;
```

Assembly

pstart

```

3100 LDS      #12544
3103 LDAA     #241
3105 STAA     0x03
3107 LDAA     #0
3109 STAA     0x01
310B LDAA     0x01
310D ANDA     #1
310F BNE     *+34      ;abs = 0x3131
3111 JSR     0x315D
3114 JSR     0x315D
3117 JSR     0x3151
311A JSR     0x3151
311D JSR     0x3151

```

Register

HC12	CPU Cycles: 0	Auto
D	CBCB	A [CB] B [CB]
IX	CBCB	IY CBCB
IP	3100	PC 3100 PPAGE 0
SP	CBC9	CCR SXHIN2VC

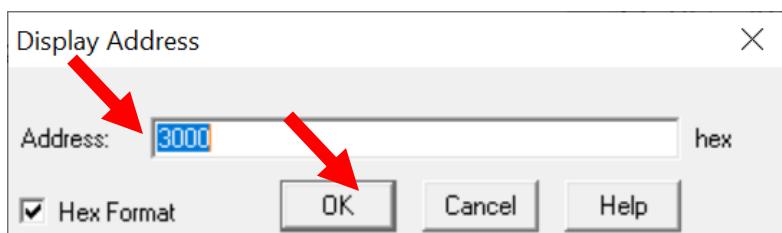
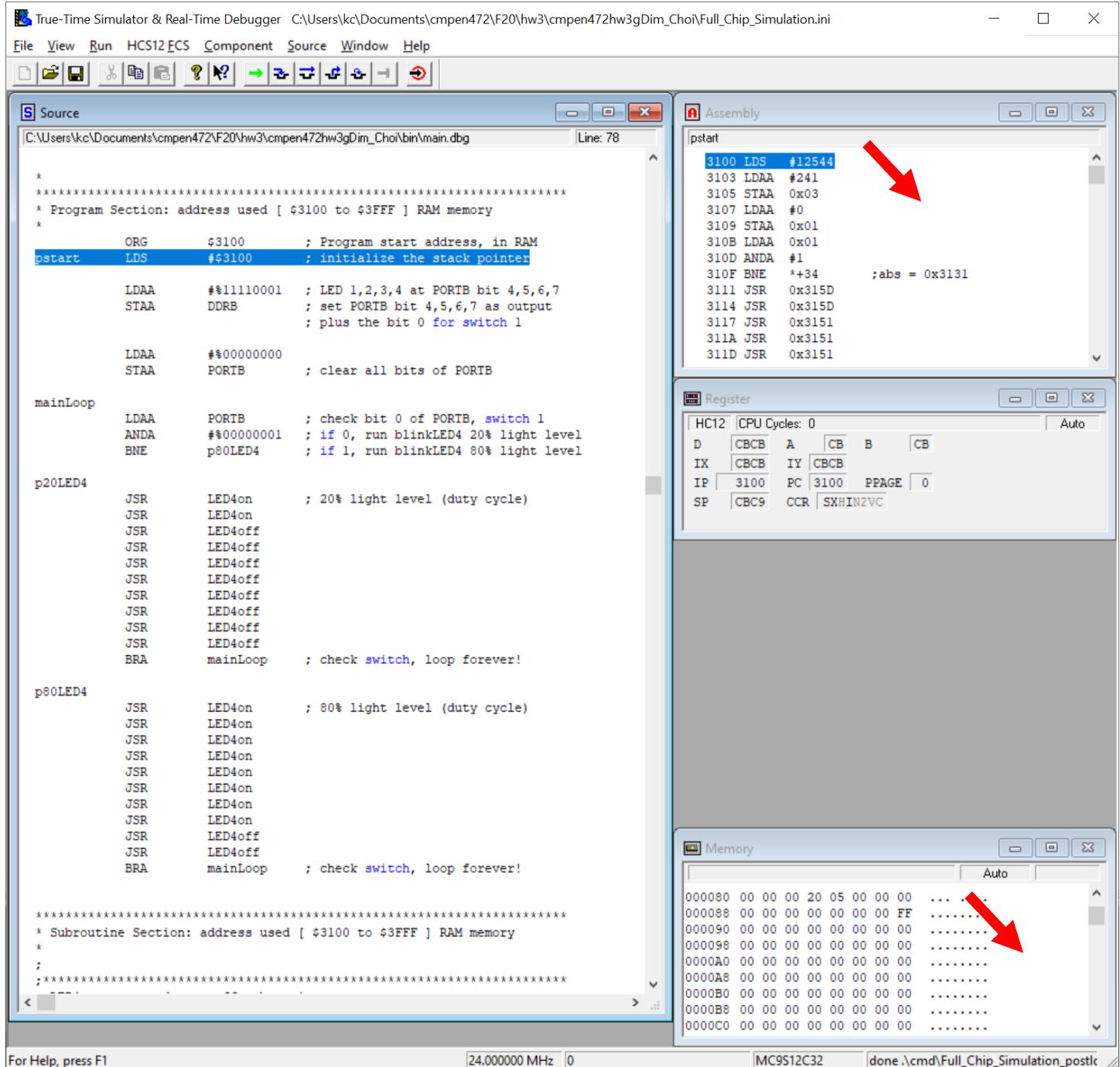
Memory

000080	00 00 00 20 05 00 00 00
000088	00 00 00 00 00 00 00 FF
000090	00 00 00 00 00 00 00 00
000098	00 00 00 00 00 00 00 00
0000A0	00 00 00 00 00 00 00 00
0000A8	00 00 00 00 00 00 00 00
0000B0	00 00 00 00 00 00 00 00
0000B8	00 00 00 00 00 00 00 00
0000C0	00 00 00 00 00 00 00 00

For Help, press F1 24.000000 MHz | 0 MC9S12C32 done \cmd\Full_Chip_Simulation_postlc

Display machine code on Assembly window: right click Assembly window and select “Display”, and then select “Code”.

Also display memory location \$3000 on the Memory window: right click Memory window and select “Address”. Then type “3000” and click “OK”.



True-Time Simulator & Real-Time Debugger C:\Users\kc\Documents\cmpen472\F20\hw3\cmpen472hw3gDim_Cho\Full_Chip_Simulation.ini

File View Run HCS12FCS Component Memory Window Help

S Source C:\Users\kc\Documents\cmpen472\F20\hw3\cmpen472hw3gDim_Cho\bin\main.dbg Line: 78

```

*
***** Program Section: address used [ $3100 to $3FFF ] RAM memory
*
      ORG      $3100      ; Program start address, in RAM
pstart LDS      #$3100      ; initialize the stack pointer

      LDAA     #\$11110001      ; LED 1,2,3,4 at PORTB bit 4,5,6,7
      STAA     DDRB      ; set PORTB bit 4,5,6,7 as output
                      ; plus the bit 0 for switch 1

      LDAA     #\$00000000
      STAA     PORTB      ; clear all bits of PORTB

mainLoop
      LDAA     PORTB      ; check bit 0 of PORTB, switch 1
      ANDA     #\$00000001      ; if 0, run blinkLED4 20% light level
      BNE     p80LED4      ; if 1, run blinkLED4 80% light level

p20LED4
      JSR      LED4on      ; 20% light level (duty cycle)
      JSR      LED4on
      JSR      LED4off
      BRA     mainLoop      ; check switch, loop forever!

p80LED4
      JSR      LED4on      ; 80% light level (duty cycle)
      JSR      LED4on
      JSR      LED4on
      JSR      LED4on
      JSR      LED4on
      JSR      LED4on
      JSR      LED4on
      JSR      LED4off
      JSR      LED4off
      BRA     mainLoop      ; check switch, loop forever!

*****
* Subroutine Section: address used [ $3100 to $3FFF ] RAM memory
*
;
;
```

A Assembly pstart

```

3100 CF3100 LDS #12544
3103 86F1 LDAA #241
3105 5A03 STAA 0x03
3107 8600 LDAA #0
3109 5A01 STAA 0x01
310B 9601 LDAA 0x01
310D 8401 ANDA #1
310F 2620 BNE *+34 ;abs = 0x3131
3111 16315D JSR 0x315D
3114 16315D JSR 0x315D
3117 163151 JSR 0x3151
311A 163151 JSR 0x3151
311D 163151 JSR 0x3151

```

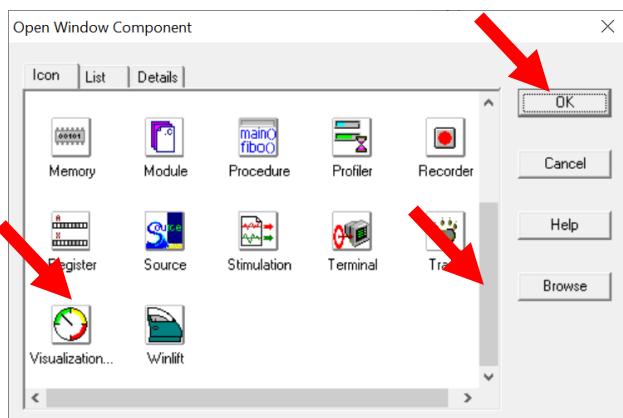
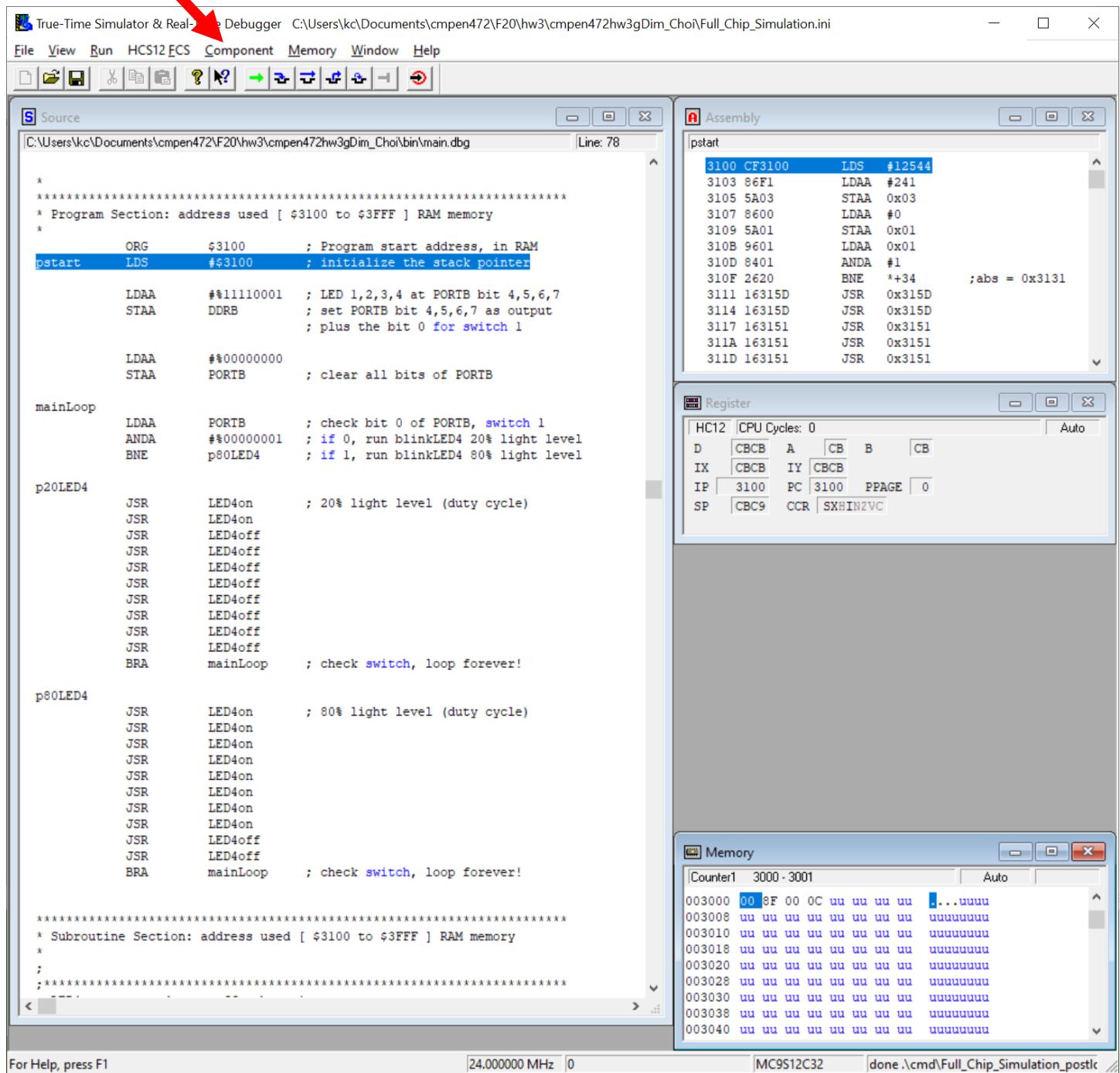
B Register

HC12	CPU Cycles: 0	Auto		
D	CBCB	A CB	B CB	
IX	CBCB	IY CBCB		
IP	3100	PC 3100	PPAGE 0	
SP	CBC9	CCR SXHIN2VC		

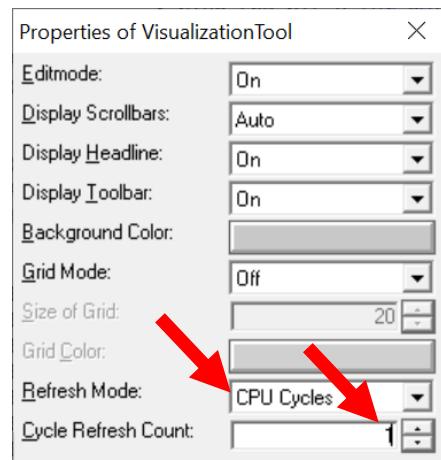
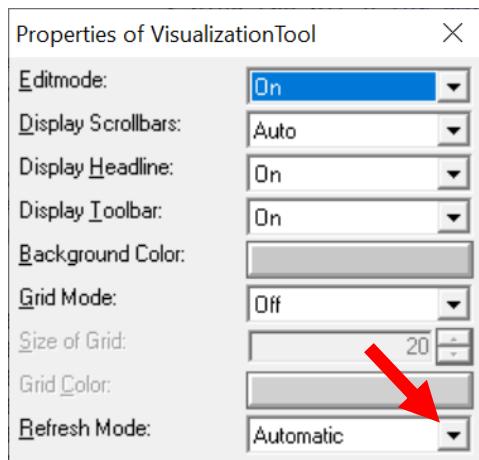
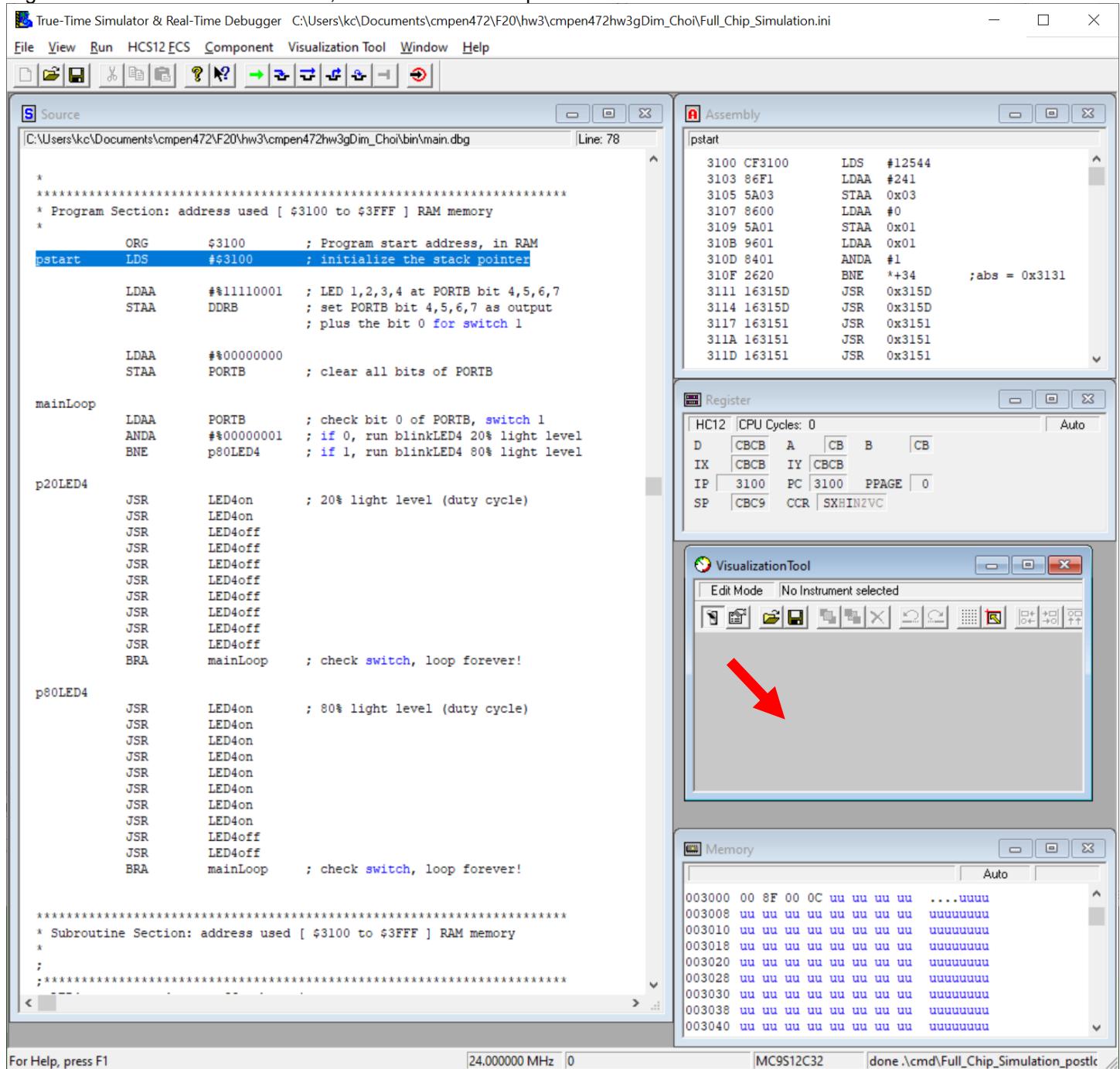
C Memory Counter1 3000-3001

003000	00 BF 00 0C	uu uu uu uu	...uuuu
003008	uu uu uu uu uu uu	uuuuuuuu	
003010	uu uu uu uu uu uu	uuuuuuuu	
003018	uu uu uu uu uu uu	uuuuuuuu	
003020	uu uu uu uu uu uu	uuuuuuuu	
003028	uu uu uu uu uu uu	uuuuuuuu	
003030	uu uu uu uu uu uu	uuuuuuuu	
003038	uu uu uu uu uu uu	uuuuuuuu	
003040	uu uu uu uu uu uu	uuuuuuuu	

Add Component, Open . . . Then select “Visualization . . .”

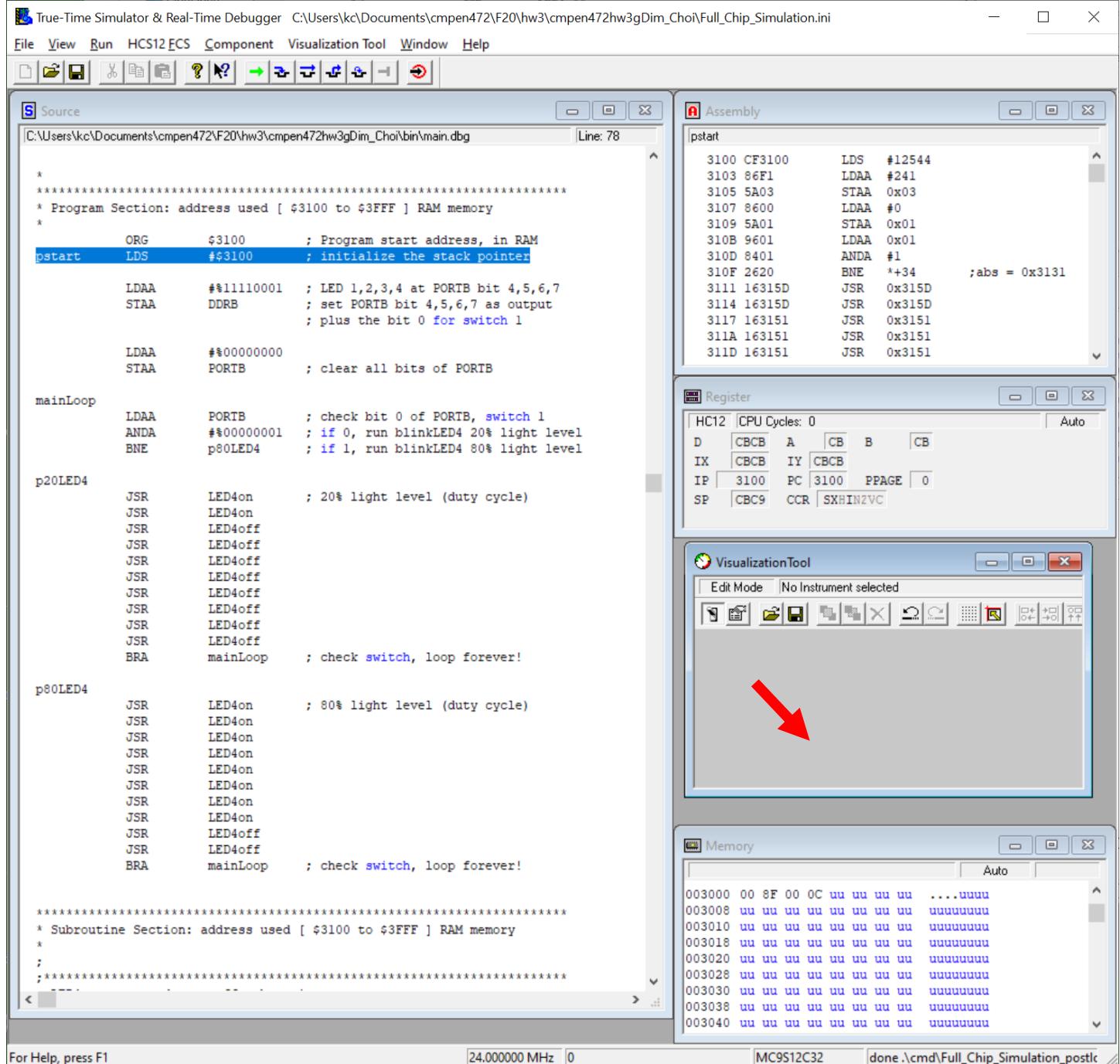


Right click on the Visualization Tool, and then select "Properties".

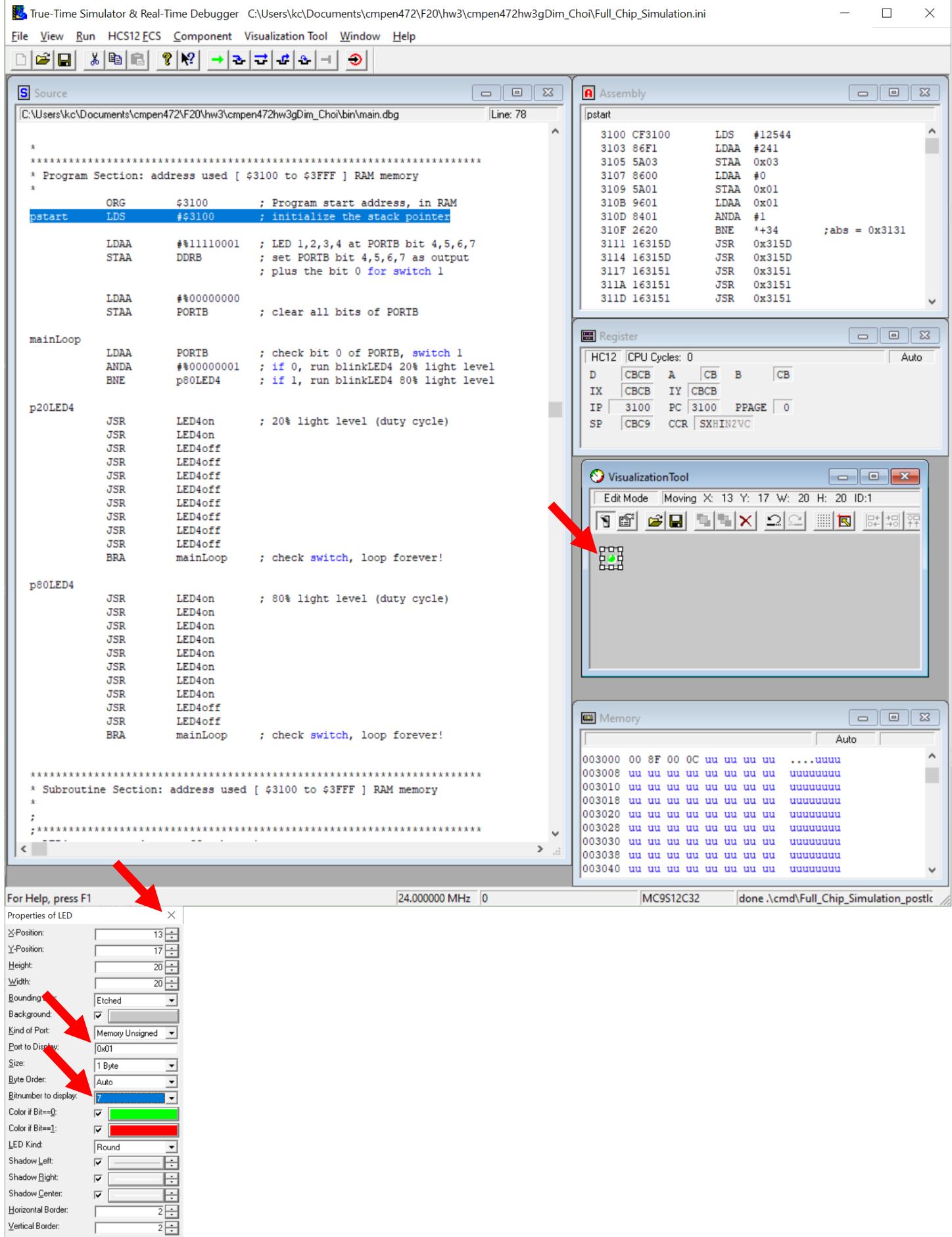


Enter "1" for Cycle Refresh Count.

Right click on the Visualization Tool, and then select “Add New Instrument”. Then select “LED”.



Right click on LED, select "Properties", then enter "1" for "Port to Display:" and select "7" for "Bitnumber to Display:".



In a similar way, add LED 1, 2, and 3. That is, connect LED 1, 2, 3, and 4 to PORTB bit 4, 5, 6, and 7, respectively.

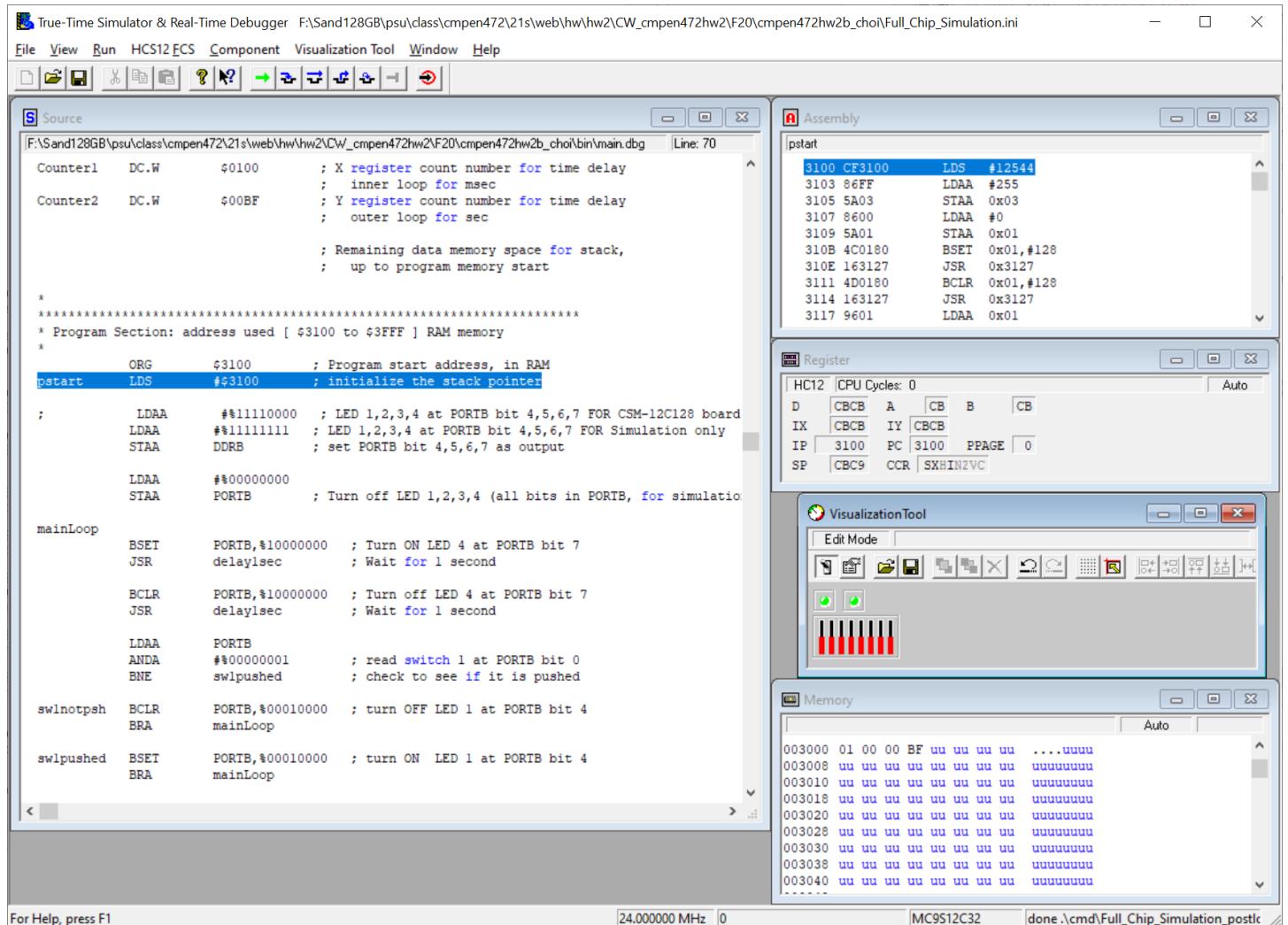
The screenshot shows the HCS12 ECS component of the True-Time Simulator & Real-Time Debugger. The interface is divided into several windows:

- Source** window: Displays the assembly code for the program. It includes sections for `pstart`, `mainLoop`, `p20LED4`, and `p80LED4`. The code initializes the stack pointer, sets up PORTB pins for LEDs, and implements a loop to toggle the LEDs based on a switch input.
- Assembly** window: Shows the assembly code for the `pstart` section. It consists of a series of `STAA` instructions with immediate values from `0x5A` to `0x5F`.
- Register** window: Displays CPU register values. The registers shown are HC12, CPU Cycles (0), D, IX, IP, SP, and various flags like CB, IY, PC, and PPAGE.
- VisualizationTool** window: A graphical interface for monitoring variables. It currently shows four green circular icons.
- Memory** window: Displays memory starting at address `003000`. The memory dump shows repeating patterns of `uu` and `uuuuuu`.

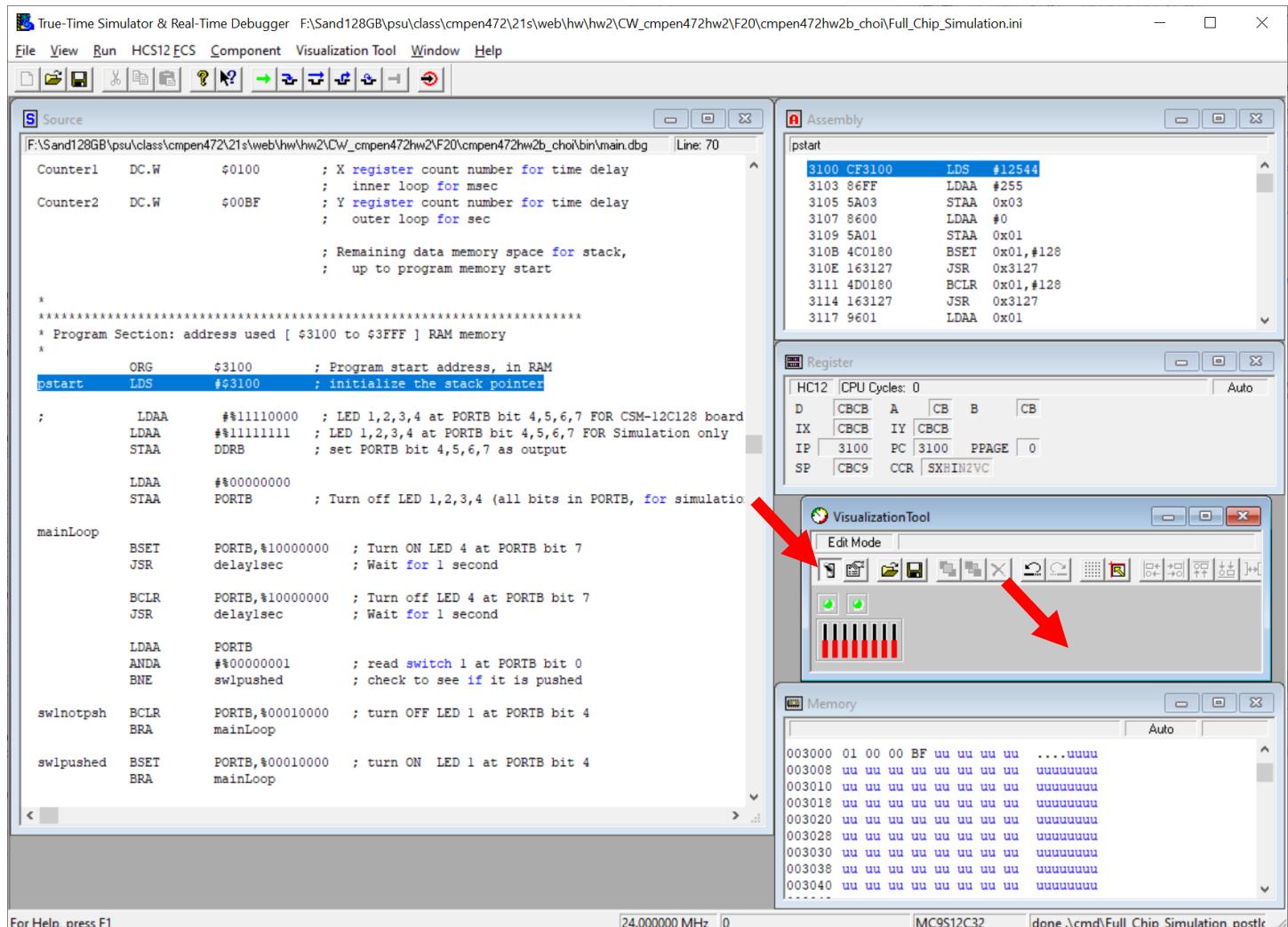
At the bottom of the interface, there are status bars for Help, Frequency (24.000000 MHz), and Processor (MC9S12C32). The command line at the bottom right shows the command used to run the simulation: `done \cmd\Full_Chip_Simulation_postl`.

```
*  
***** [ $3100 to $3FFF ] RAM memory  
*  
* Program Section: address used [ $3100 to $3FFF ] RAM memory  
*  
pstart    ORG      $3100      ; Program start address, in RAM  
          LDS      #$_3100      ; initialize the stack pointer  
  
          LDAA    #$_11110001  ; LED 1,2,3,4 at PORTB bit 4,5,6,7  
          STAA    DDRB      ; set PORIB bit 4,5,6,7 as output  
                      ; plus the bit 0 for switch 1  
  
          LDAA    #$_00000000  ; clear all bits of PORTB  
  
mainLoop   LDAA    PORTB      ; check bit 0 of PORTB, switch 1  
          ANDA    #$_00000001  ; if 0, run blinkLED4 20% light level  
          BNE     p80LED4    ; if 1, run blinkLED4 80% light level  
  
p20LED4   JSR     LED4on     ; 20% light level (duty cycle)  
          LED4on  
          LED4off  
          LED4off  
          LED4off  
          LED4off  
          LED4off  
          LED4off  
          LED4off  
          LED4off  
          BRA    mainLoop    ; check switch, loop forever!  
  
p80LED4   JSR     LED4on     ; 80% light level (duty cycle)  
          LED4on  
          LED4on  
          LED4on  
          LED4on  
          LED4on  
          LED4on  
          LED4on  
          LED4on  
          LED4off  
          LED4off  
          BRA    mainLoop    ; check switch, loop forever!  
  
***** [ $3100 to $3FFF ] RAM memory  
*  
* Subroutine Section: address used [ $3100 to $3FFF ] RAM memory  
*  
;  
;  
;
```

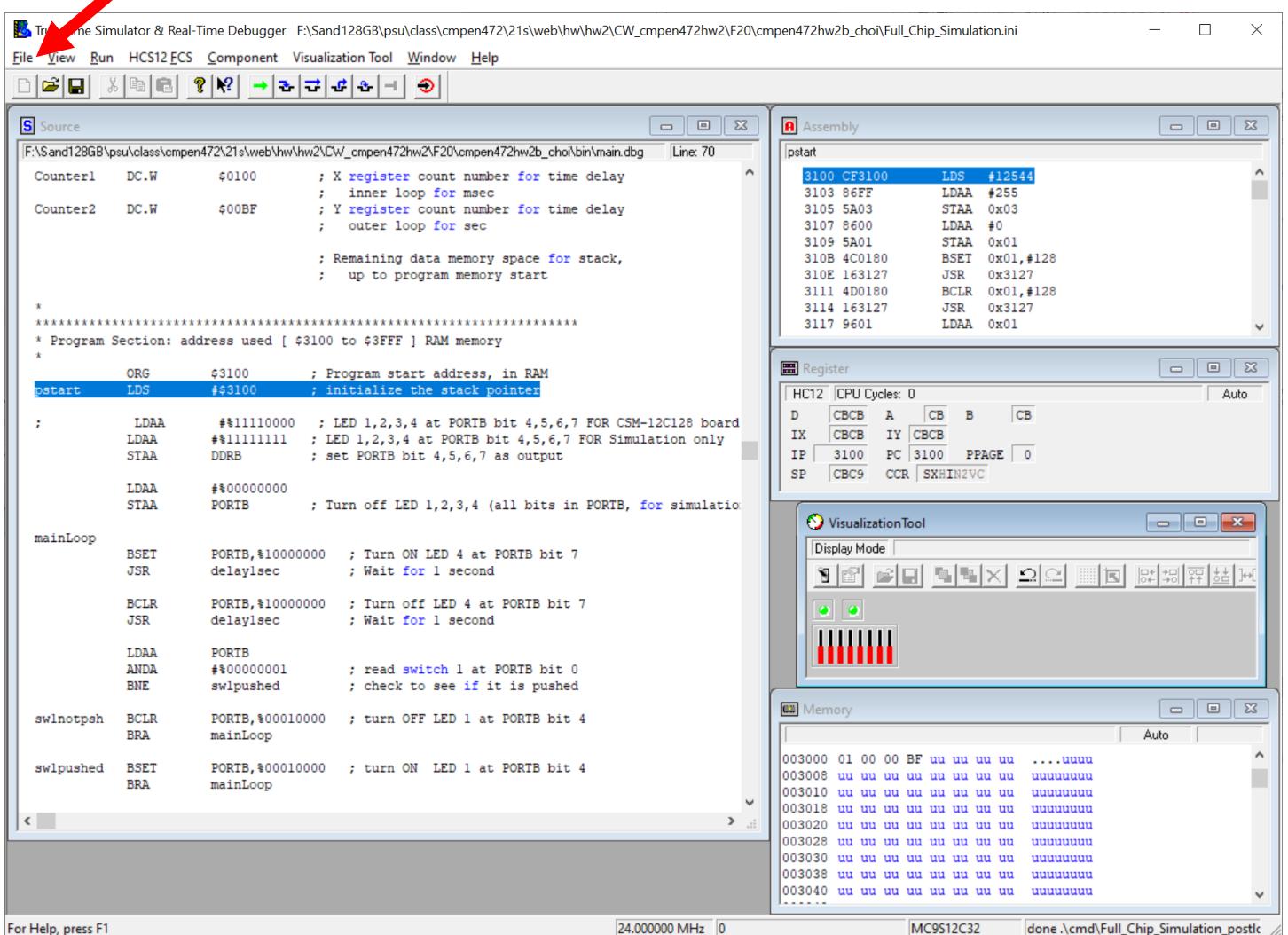
Right click on the Visualization Tool, and then select “Add New Instrument”. Then select “DIL Switch”. Be sure to connect DIL Switch to PORTB (Port to Display set to 0x01 address) in its “properties”.



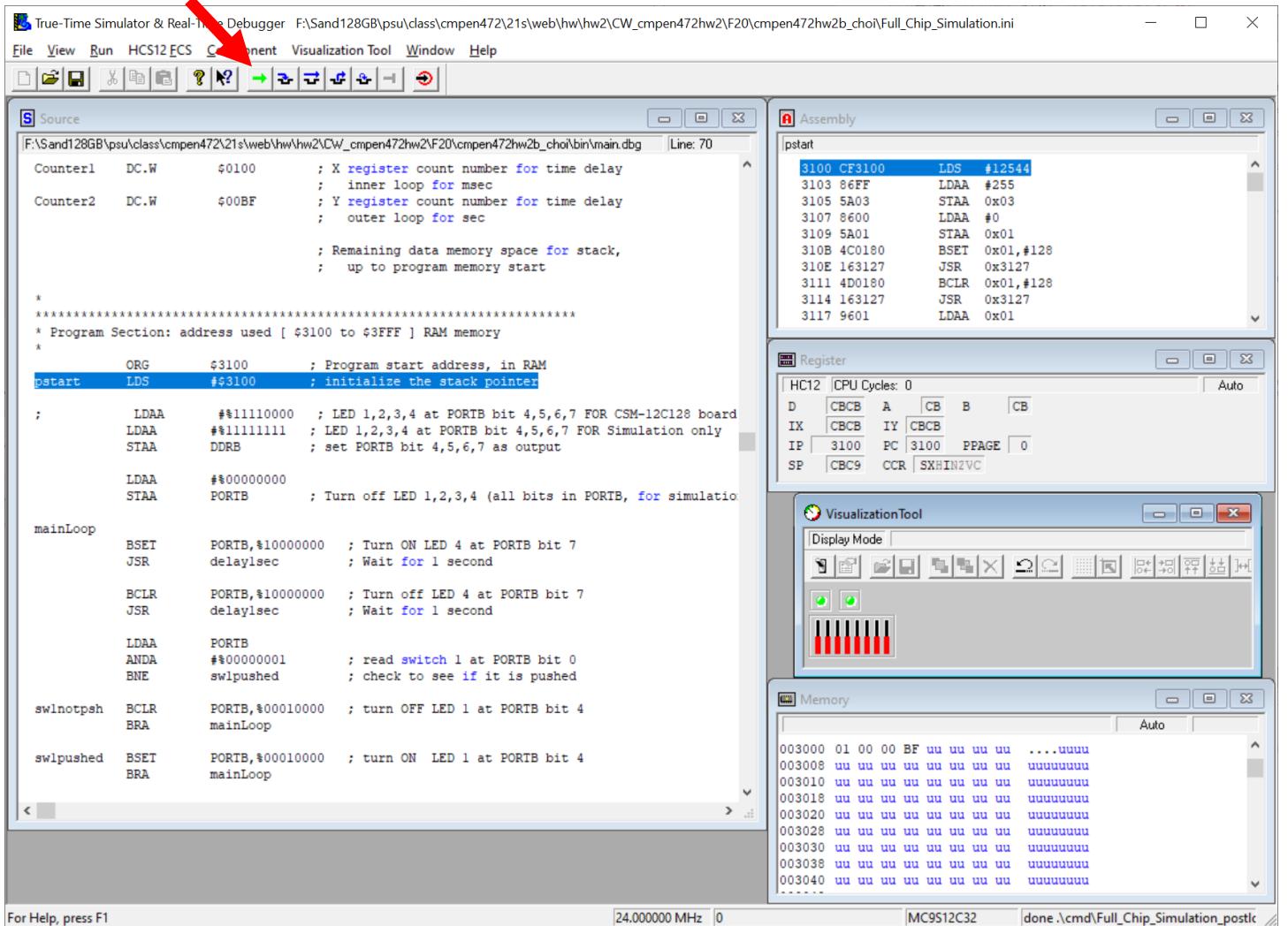
Now change your Visualization Tool from “Edit Mode” to “Display Mode”, by right clicking the Visualization Tool and select the checked “Edit Mode”. The switch at PORTB will not work on “Edit Mode”, they work only on “Display Mode”.



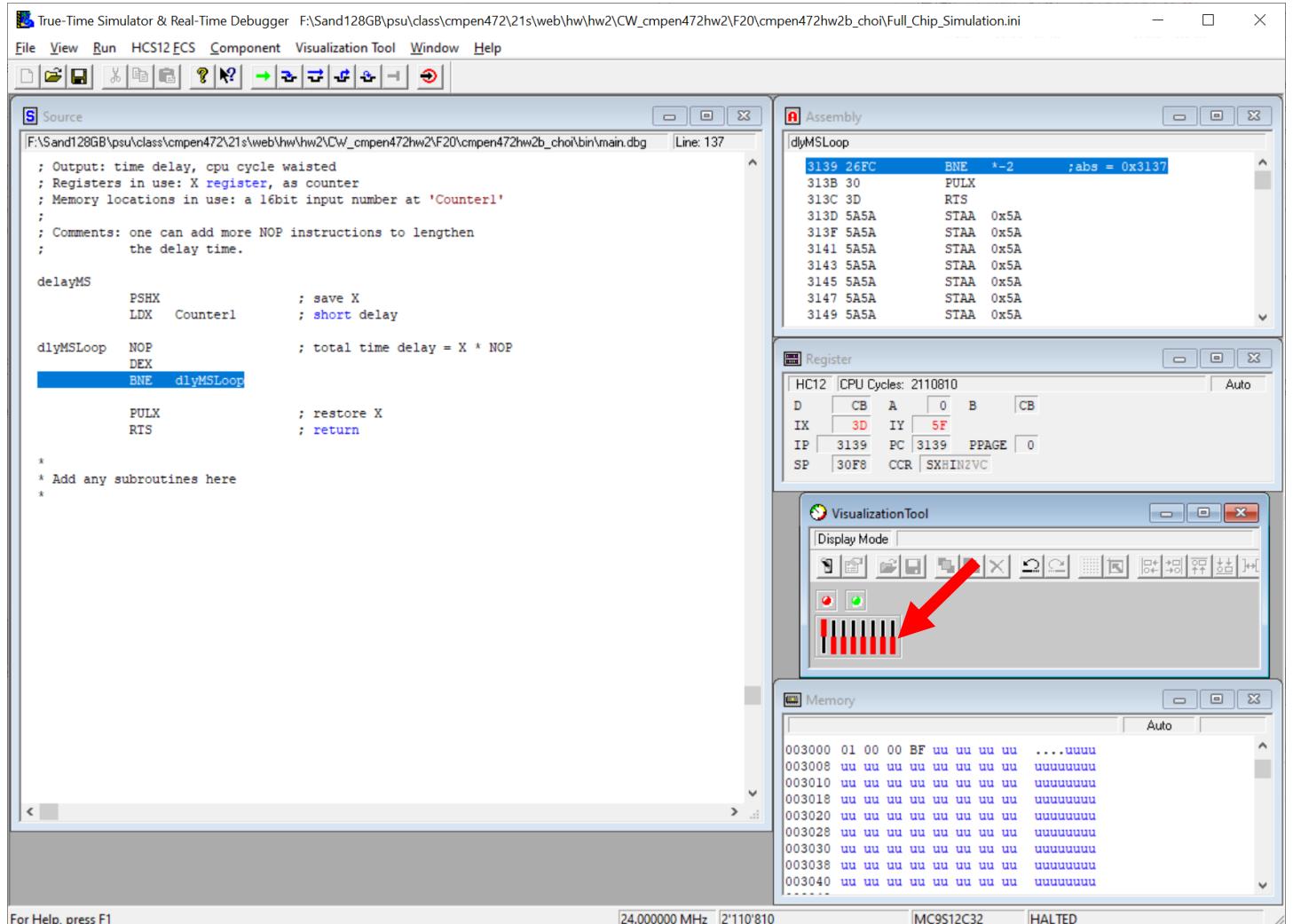
Now the Debugger/Simulator is ready to simulate Homework 2 Sample program and you want to SAVE the setting. Click on the “File” menu and select the “Save Configuration”.



Now run your program and observe LED 4 blinking.



While the program is running, click the Switch 1 in the Visualization Tool. And observe the LED 1 state changing based on the Switch 1 ON or OFF state.



Now you can modify the Sample Homework 2 program to finish the full Homework 2, modify to meet program's full specification.