

Labs 1,2,3,4**HW1-5****Lectures 1-10 (no SPI, DAC)**

Since the exam is closed book without calculator, you will need to know some basic constants:

$2^{**}0=1=0x01$

$2^{**}6=64=0x40$

$2^{**}12=4096=0x1000$

$2^{**}1=2=0x02$

$2^{**}7=128=0x80$

$2^{**}13=8192=0x2000$

$2^{**}2=4=0x04$

$2^{**}8=256=0x0100$

$2^{**}14=16384=0x4000$

$2^{**}3=8=0x08$

$2^{**}9=512=0x0200$

$2^{**}15=32768=0x8000$

$2^{**}4=16=0x10$

$2^{**}10=1024=0x0400$

$2^{**}16=65536$

$2^{**}5=32=0x20$

$2^{**}11=2048=0x0800$

Range of integers for 8-bit signed (-128 to +127), 8-bit unsigned (0 to 255)

Range of integers for 16-bit signed (-32768 to +32767), 16-bit unsigned (0 to 65535)

Fixed-point numbers (decimal/binary, signed/unsigned, 8/16 bit)

- 1) Convert value to integer. E.g., What integer is stored in the computer, when the value 2.1 is stored in 16-bit unsigned binary fixed-point, with a resolution of 2^{**-10} ? Answer: $I = 2.1 * 1024$, which is about 2150. I will make the math so easy a calculator will not be needed.
- 2) Convert integer to value. E.g., What is the value of an 8-bit signed decimal fixed-point number (resolution is 0.1) if the integer stored in memory is -123? Answer: $-123 * 0.1$ equals -12.3.
- 3) Basic concepts of range, resolution and rounding.
- 4) Given the range and resolution, choose the format

Programming (C on the 9S12C32) Techniques

- 1) How to check for overflow when performing integer calculations in C. Answer: promote to higher precision and check intermediate results.
- 2) Stack picture of parameters, and locals
- 3) Memory allocation: global, local, constants, and reset/interrupt vectors
- 4) Modularity linking call graph to `#include 'file.h'`, private versus public
- 5) `const volatile static`
- 6) Data flow graph
- 7) Debugging: dumps, monitors, scans, breaks, filter, profiling
- 8) Putting prototypes to public functions in the `'file.h'`, implementations in the `'file.c'`

I/O programming (on the 9S12C32/9S12DP512): focus on concepts and don't memorize details

- 1) Direction register. Friendly means setting only the bits that are needed, leaving the rest unchanged.
- 2) Masking input bits to check for individual signals, making individual output bits high, low, or toggle.
- 3) Using TCNT to measure elapsed time, and to create a time delay
- 4) Output compare interrupts: rate, and vector
- 5) SCI input/output (busy-wait synchronization)
- 6) Moore and Mealy FSM, running the machine using output compare interrupts

Interfacing

- 1) Switches and LEDs
- 2) Stepper motors
- 3) Speaker (where software creates a squarewave)
- 4) Solid state relay (just like a LED)
- 5) Solenoid or electromagnetic relay (just like one coil of a stepper motor)

Definitions (match definition with the following terms)

Real-time, friendly, latency, ROM, RAM, computer, CPU, processor, ALU, BIU, CU, registers, read cycle, write cycle, address bus, data bus, control bus, embedded computer, microprocessor, microcomputer, microcontroller, I_{OL} , I_{OH} , I_{IL} , I_{IH} , V_{OL} , V_{OH} , V_{IH} , V_{IL} , nonvolatile, open collector, tristate, memory-mapped I/O, functional debugging, performance debugging, non-intrusiveness, profile, desk check, instrument, stabilize, scan, break, thread, busy waiting, atomic, **critical section**, **reentrant**, interrupt vector, interrupt acknowledge, interrupt arm, interrupt enable.

Old Quizzes and Exams (wherever you see RTI interrupts, replace with output compare interrupts)**Definitions**

Spring 1999 Quiz, Question 2, Memory allocation
Spring 2000 Quiz1, Question 3, Critical section
Spring 2001 Quiz 1, Question 1, C programming syntax `const volatile static`
Spring 2003 Final, Question 12, desk checking
Spring 2003 Final, Question 14, interface latency
Spring 2003 Final, Question 15, polled interrupt
Spring 2003 Final, Question 16, private
Spring 2003 Final, Question 17, intrusive
Spring 2003 Quiz 1, Question 12, scanpoint
Spring 2003 Quiz 1, Question 13, stabilization
Spring 2003 Quiz 1, Question 14, profile
Spring 2003 Quiz 1, Question 15, busy-waiting
Spring 2003 Quiz 1, Question 1-5, Where in memory are variables allocated?
Spring 2003 Quiz 1, Question 16, real time system
Spring 2003 Quiz 1, Question 17, interrupt acknowledge
Spring 2003 Quiz 1, Question 18, ALU
Spring 2003 Quiz 1, Question 19, volatile
Spring 2003 Quiz 1, Question 20, output low current
Spring 2003 Quiz 1, Question 21, nonintrusive
Spring 2003 Quiz 1, Question 22, friendly, set direction register, toggle output
Spring 2003 Quiz 1, Question 6, Data flow graph
Spring 2004 Quiz 1, Question 1, Which variables are stored on the stack?
Spring 2004 Quiz 1, Questions 6,9,12, volatile, open collector, busy-waiting
Spring 2004 Quiz 1, Questions 8,10,13, nonintrusive, stabilization, real time
Spring 2004 Quiz 2, Question 3, FIFO principles
Spring 2005 Final, Question 10, Fixed point
Spring 2005 Final, Question 16, Desk check
Spring 2005 Final, Question 17, Bandwidth
Spring 2005 Final, Question 18, Latency
Spring 2005 Final, Question 19, Polled interrupt
Spring 2005 Final, Question 2, Critical section
Spring 2005 Final, Question 20, Private
Spring 2005 Final, Question 21, Nonintrusive
Spring 2005 Final, Question 22, Buffered I/O
Spring 2005 Final, Question 24, Associative principle
Spring 2005 Final, Question 3, What does **short** mean?
Spring 2005 Final, Question 4, What does **const** mean?
Spring 2005 Final, Question 5, What does **static** mean?
Spring 2005 Final, Question 7, Debugging instruments
Spring 2005 Quiz 1, Question 2, Intrusive
Spring 2005 Quiz 1, Question 3, Open collector
Spring 2005 Quiz 1, Question 4, Stabilizing
Spring 2005 Quiz 1, Question 5, Tristate logic
Spring 2005 Quiz 1, Question 6, Latency
Spring 2006 Quiz 1, Question 1. I/O bound
Spring 2006 Quiz 1, Question 2. vectored interrupt
Spring 2006 Quiz 1, Question 3. intrusive
Spring 2006 Quiz 1, Question 4. stabilization
Spring 2006 Quiz 1, Question 5. tristate
Spring 2006 Quiz 1, Question 6. latency
Spring 2006 Quiz 1, Question 7. buffered I/O

Spring 2007 Quiz 1, Questions 1-5, Intrusive
Fall 2007 Quiz 1, Question 1, Intrusive
Fall 2007 Quiz 1, Question 2, Debugging instrument, which interrupt occurs first?
Spring 2007 Quiz 1, Questions 8-12, Where are variables allocated?
Fall 2007 Quiz 1, Question 4, Interface latency
Spring 2008 Quiz 1, Question 2, Intrusive
Spring 2008 Quiz 1, Question 6, Where are local variables allocated?
Spring 2008 Quiz 1, Question 8, V_{IH} , V_{IL}

Interrupts

Spring 2001 Final, Question 5, `cli`, `sei`
Spring 2001 Final, Question 8e, RTI acknowledge
Spring 2002 Quiz 1, Question 1, RTI sequence and critical section
Spring 2002 Quiz 1, Question 3, Interrupt stack
Spring 2003 Final, Question 19e, acknowledge RTI
Spring 2003 Final, Question 3, What happens if a ISR does not acknowledge?
Spring 2005 Final, Question 1, Items on the stack during the execution of an interrupt service routine
Spring 2005 Final, Question 9, What three conditions cause a RTI interrupt?
Spring 2007 Quiz 1, Question 7, Critical section
Spring 2007 Quiz 1, Question 13, What causes an output compare interrupt?
Fall 2007 Quiz 1, Question 3, Debugging instrument, critical section
Fall 2007 Quiz 1, Question 9, Output compare software
Spring 2008 Quiz 1, Question 5, How do we make a system real time?

Fixed-Point

Spring 2003 Final, Question 8, Fixed-point math, implemented in C
Spring 2005 Final, Question 30, Fixed point multiply
Spring 2004 Quiz 1, Questions 2,3,5, Fixed-point numbers
Spring 2005 Quiz 1, Questions 1,7, Fixed-point numbers
Spring 2003 Quiz 1, Questions 7-10, Fixed-point numbers
Spring 2006 Quiz 1, Question 13, How to store the integer part of a fixed-point number
Spring 2007 Quiz 1, Question 6, How to store the integer part of a fixed-point number
Fall 2007 Quiz 1, Question 5, Choosing fixed-point format
Spring 2008 Quiz 1, Question 3, Choosing fixed-point format
Spring 2008 Quiz 1, Question 4, Writing fixed-point math software

FSM

Spring 2000 Quiz 1, Question 2, RTI interrupt, FSM
Spring 2001 Quiz 2, Question 2, RTI/TOF/OC interrupt, FSM
Spring 2002 Final, Question 3, Drawing a FSM state graph
Spring 2002 Quiz 2, Question 3, FSM controller
Spring 2003 Final, Question 20, FSM controller
Spring 2003 Quiz 2, Question 2, FSM controller
Spring 2004 Final, Question 11, FSM controller
Spring 2004 Quiz 2, Question 6, FSM controller
Spring 2005 Final, Question 25, FSM analysis
Spring 2005 Quiz 2, Question 7, Output compare-driven FSM
Spring 2006 Quiz 2, Question 7, Output compare-driven FSM
Spring 2007 Quiz 1, Question 16, Output compare-driven FSM
Fall 2007 Quiz 1, Question 8, Drawing a FSM state graph
Spring 2008 Quiz 1, Question 9, Output compare-driven FSM

Hardware interfacing

Spring 1998 Quiz, Question 1, solenoid interface
Spring 1999 Final, Question 3, stepper motor
Spring 2001 Final, Question 2, Solenoid interface

Spring 2001 Final, Question 7, Solid state relay
Spring 2001 Quiz 1, Question 2, LED interface
Spring 2002 Final, Question 2, Stepper motor
Spring 2002 Quiz 2, Question 2, LED interface
Spring 2003 Final, Question 5, Solid state relay
Spring 2003 Quiz 2, Question 3, LED interface
Spring 2003 Quiz 2, Question 4, Motor interface
Spring 2004 Final, Question 4, Solid state relay
Spring 2004 Quiz 2, Question 2 Solenoid interface
Spring 2004 Quiz 2, Question 4, LED interface
Spring 2005 Final, Question 29, relay interface
Spring 2005 Quiz 2, Question 2, Solenoid interface
Spring 2005 Quiz 2, Question 4, LED interface
Spring 2005 Quiz 2, Question 6, Stepper motor fundamentals
Spring 2006 Quiz 1, Question 11, LED interface
Spring 2006 Quiz 1, Question 14, Switch debouncing using interrupts
Spring 2007 Quiz 1, Question 14, LED interface
Spring 2007 Quiz 1, Question 15, Stepper interface
Fall 2007 Quiz 1, Question 6, LED interface
Fall 2007 Quiz 1, Question 7, Solenoid interface
Spring 2008 Quiz 1, Question 7, LED interface

The following is the exact information that will be provided on Quiz 1

PTT is 8-bit bi-directional I/O port

DDRT is the associated direction register for Port T (0 means input, 1 means output)

PTM is 6-bit bi-directional I/O port

DDRM is the associated direction register for Port M (0 means input, 1 means output)

TSCR1 is the first 8-bit timer control register

bit 7 **TEN**, 1 allows the timer to function normally, 0 means disable timer including **TCNT**

TSCR2 is the second 8-bit timer control register

bits 2,1,0 are **PR2**, **PR1**, **PR0**, which select the rate, let **n** be the 3-bit number formed by **PR2**, **PR1**, **PR0** without PLL **TCNT** is $4\text{MHz}/2^n$, with PLL **TCNT** is $24\text{MHz}/2^n$, **n** ranges from 0 to 7

TCNT is 16-bit up counter

TIOS is the 8-bit output compare select register, one bit for each channel (1 = output compare, 0 = input capture)

TIE is the 8-bit output compare arm register, one bit for each channel (1 = armed, 0 = disarmed)

TC0 TC1 TC2... TC7 are the eight 16-bit output compare registers, one register for each channel

TFLG1 is the 8-bit flag register, one bit for each channel,

(with output compare, flags are set when **TCNT** equals **TC0 TC1 TC2... TC7**)

flags become zero when software writes a 1 to it (e.g., **TFLG1=0x08**; clears channel 3 flag)

SCIDRL 8-bit data serial data register

SCIBD is 16-bit SCI baud rate register, let **n** be the 16-bit number Baud rate is $12\text{MHz}/n$

SCICR1 is 8-bit SCI control register

bit 4 M, Mode, 0 = One start, eight data, one stop bit, 1 = One start, eight data, ninth data, one stop bit

SCICR2 is 8-bit SCI control register

bit 7 TIE, Transmit Interrupt Enable, 0 = TDRE interrupts disabled, 1 = interrupt whenever TDRE set

bit 5 RIE, Receiver Interrupt Enable, 0 = RDRF interrupts disabled, 1 = interrupt whenever RDRF set

bit 3 TE, Transmitter Enable, 0 = Transmitter disabled, 1 = SCI transmit logic is enabled

bit 2 RE, Receiver Enable, 0 = Receiver disabled, 1 = Enables the SCI receive circuitry.

SCISR1 is 8-bit SCI status register

bit 7 TDRE, Transmit Data Register Empty Flag

Set if transmit data can be written to SCDR

Cleared by **SCISR1** read with TDRE set followed by **SCIDRL** write.

bit 5 RDRF, Receive Data Register Full

set if a received character is ready to be read from **SCIDRL**

Clear the RDRF flag by reading **SCISR1** with RDRF set and then reading **SCIDRL**.

ATDDIEN ADC digital enable register, 1 to make corresponding pin digital, 0 to make corresponding pin analog

PTAD is 8-bit bi-directional I/O port

DDRAD is the associated direction register for digital pins of Port AD (0 means input, 1 means output)

0xFFD6	interrupt 20	SCI
0xFFDE	interrupt 16	timer overflow
0xFFE0	interrupt 15	timer channel 7
0xFFE2	interrupt 14	timer channel 6
0xFFE4	interrupt 13	timer channel 5
0xFFE6	interrupt 12	timer channel 4
0xFFE8	interrupt 11	timer channel 3
0xFFEA	interrupt 10	timer channel 2
0xFFEC	interrupt 9	timer channel 1
0xFFEE	interrupt 8	timer channel 0
0xFFFF	interrupt 7	real time interrupt

9S12C32 parameters

$I_{OL} = 10\text{mA}$,	$I_{OH} = 10\text{mA}$,	$I_{IL} = 1\mu\text{A}$,	$I_{IH} = 1\mu\text{A}$,
$V_{OL} = 0.8\text{V}$,	$V_{OH} = 4.2\text{V}$,	$V_{IL} = 1.75\text{V}$,	$V_{IH} = 3.25\text{V}$

