This is a closed book exam. You must put your answers in the space provided. You have 3 hours, so allocate your time accordingly. Please read the entire exam before starting.
(6) Q8 a)
const struct stuff{

};
typedef const struct stuff StuffType;

(4) Q8 b)

(4) Q8 c)

(4) Q8 d)
Please read and affirm our honor code:
“The core values of The University of Texas at Austin are learning, discovery, freedom, leadership, individual opportunity, and responsibility. Each member of the university is expected to uphold these values through integrity, honesty, trust, fairness, and respect toward peers and community.”

(10) Question 1. State the term that is described by each definition.

Part a) The process of converting an unsigned 8-bit integer into an unsigned 16-bit integer.
Part b) You are given a 4-bit DAC to test. The DAC input is stepped from 0 to 15. For each input change, the change in DAC output is measured. The results are processed by averaging all the changes in output.
Part c) The part of the processor that performs: addition, multiplication, and, or, shift.
Part d) A system where the response time from when new input is ready until when the new input is processed is less than 25 μsec.
Part e) Error that can occur as a result of a left shift.
Part f) Error that can occur as a result of a right shift.
Part g) A variable that can only be accessed by one function.
Part h) A function parameter that is a pointer to the data.
Part i) A characteristic of a debugger when the presence of the collection of information itself makes a large and important effect on the parameters being measured.
Part j) A debugging process that fixes all the inputs to a system, so the systems can be run over an over yielding the same outputs.

(5) Question 2. The system uses an 8-bit ADC and a serial port running at 100 bits/sec. Assume every ADC sample you take must be transmitted over the serial channel.

Part a) How many bytes of information per second are being transferred?
Part b) At this rate, what would be ADC sampling rate in Hz?

(5) Question 3. What is the output voltage $V_{out}$ when PT1 is high and PT0 is low? Assume $V_{OH}$ is 5V and $V_{OL}$ = 0V.

(10) Question 4. Assume Register X contains the integer portion of an unsigned binary fixed point number with resolution $2^{-4}$, and Register Y contains the integer portion of an unsigned binary fixed point number with resolution $2^{-2}$. For example, if the first number is 1.5 then Register X equals 24. If the second number is 2.25, then Register Y is 9. Write assembly code that adds the two numbers such that the sum is in Register D with a resolution of $2^{-2}$. Since 1.5+2.25 is 2.75, Register D should be returned with 11. No global variables are allowed, but you may use the stack. Handle potential overflow errors by implementing ceiling. Some dropout may occur.

(10) Question 5. There are arrays of 16-bit numbers. The first element of the array is the length and remaining elements are 16-bit signed numbers. For example, here are three such possible arrays.

```
short buf1[5]={4,1000,-1000,0,33};
short buf2[7]={6,-4,100,200,2,0,44};
short buf3[1]={0};
```

Part a) Write a C function that takes a pointer to an array and returns the difference between the maximum and minimum values. For example
Result1 = MaxDiff(buf1);  // should return 2000 = 1000 - (-1000)
Result2 = MaxDiff(buf2);  // should return 204 = 200 - (-4)
Result3 = MaxDiff(buf3);  // should return 0 because array is empty

You are not allowed to add global variables. Don’t worry about overflow calculating the difference.

Part b) Write an assembly subroutine that performs the same operation. The pointer to the array is passed in Register D, and the result is returned in Register D. You are not allowed to add any global variables. You must use binding to implement local variables.

(5) Question 6. The following interface can be used for low current LEDs.
Assume the LED voltage drop is 2 V. The resistor is 1000 Ω. When the software outputs a high, the voltage on PP0 becomes 4.9 V. When the software outputs a low, the voltage on PP0 becomes 0.5 V. What is the LED current when the LED is on?

(5) Question 7. Assume Register B equals $55, Register A equals $F0 and Register X equals $5678. What is the value in Register X after executing these instructions? Give the answer as ??? if the value cannot be determined.
   ```
   stx  2,-sp
   std  2,sp-
   pulx
   puld
   ```

(18) Question 8. This question tests your ability to create and use structures.
Part a) Complete the C code that defines a structure containing an array of three 8-bit unsigned numbers, and one 16-bit unsigned number. Call the array Position, and call the number Time.

Part b) Use the StuffType structure to define a ROM-based constant with a Position of {100,60,50} and a Time of 1000. Call this constant Command.

Part c) Write a C code (no function, just code) that accesses the above constant and sets a variable max to the largest position number of the three. In this case, max will become 100.

Part d) Write a C function that takes a pointer to a constant and returns the largest position number of the three. One possible way to call your function is
   ```
   max = MaxPosition(&Command);
   ```
In this case, max will become 100.

(2) Question 9. You are given two 8-bit numbers, where each number is known to exist between 0 and 100. An 8-bit addition is operated on two numbers. Is it possible for the overflow (V) bit to be set?

(5) Question 10. Assume the E clock is operating at 8 MHz, and TSCR2 = 4. The output compare ISR executes these instructions. What value goes in ????? to make the interrupt frequency 100 Hz?
   ```
   OC6ISR movb #$40,TFLG1
   ldd TC6
   ```
addd #?????
std  TC6
rti

(5) Question 11. Assume the PC contains $4007, and the SP equals $3FF4.
$4007  0750    bsr Function
Part a) What number is pushed on the stack during the execution of bsr?

Part b) What is the value in the PC after bsr is executed?

(20) Question 12. In this problem, your software should output the alphabet ‘A’ ‘B’ ‘C’ … ‘Z’ over and over using SCI0 serial port. You must use SCI0 interrupts (not output compare). The baud rate is 10000 bits/sec. You may assume the E clock is 8 MHz.

Part a) Show the C code that specifies any global variables you need.

Part b) Write the initialization function in C that sets up the SCI0 interrupts. The main will call this initialization once at the beginning, and then perform unrelated tasks. This function should arm and enable interrupts. No loops are allowed.

Part c) Write the ISR in C that outputs the alphabet using SCI0. No loops are allowed.
movw 16-bit move memory to memory  
movw #13,SCIBD
mul 8 by 8 to 16-bit unsigned  
RegD=RegA*RegB
neg 8-bit 2's complement negate memory
negb 8-bit 2's complement negate RegB
ora 8-bit logical or to RegA
orab 8-bit logical or to RegB
orcc 8-bit logical or to RegCC
psha push 8-bit RegA onto stack
pshb push 8-bit RegB onto stack
pshc push 8-bit RegCC onto stack
psdh push 16-bit RegD onto stack
psdx push 16-bit RegX onto stack
pshy push 16-bit RegY onto stack
pula pop 8 bits off stack into RegA
pulb pop 8 bits off stack into RegB
pulc pop 8 bits off stack into RegCC
puld pop 8 bits off stack into RegD
puly pop 8 bits off stack into RegX
pulx pop 8 bits off stack into RegY
rev Fuzzy logic rule evaluation
revw weighted Fuzzy logic rule evaluation
rol 8-bit roll shift left Memory
rola 8-bit roll shift left RegA
rolb 8-bit roll shift left RegB
ror 8-bit roll shift right Memory
rorc 8-bit roll shift right RegA
rorb 8-bit roll shift right RegB
rtc return sub in expanded memory
rti return from interrupt
rts return from subroutine
sba 8-bit subtract RegA=RegA-RegB
sbca 8-bit sub with carry from RegA
sbcb 8-bit sub with carry from RegB
sec set carry bit, C=1
sei set I=1, disable interrupts
sev set overflow bit, V=1
sex sign extend 8-bit to 16-bit reg sex B,D
staa 8-bit store memory from RegA
stab 8-bit store memory from RegB
std 16-bit store memory from RegD
sts 16-bit store memory from SP
stx 16-bit store memory from RegX
styx 16-bit store memory from RegY
suba 8-bit sub from RegA
subb 8-bit sub from RegB
subd 16-bit sub from RegD
swi software interrupt, trap
tab transfer A to B
tap transfer A to CC
tha transfer B to A
tbeq test and branch if result=0  
tbeq Y,loop
tbl 8-bit look up and interpolation
tbne test and branch if result≠0

sbne A,loop
tfr transfer register to register tfr X,Y
tpa transfer CC to A	rap illegal instruction interrupt
trap illegal op code, or software trap
tst 8-bit compare memory with zero
tsta 8-bit compare RegA with zero
tstb 8-bit compare RegB with zero
tsx transfer S to X
tsy transfer S to Y
txs transfer X to S
tys transfer Y to S
wai wait for interrupt
wav weighted Fuzzy logic average

Example | Mode       | Effective Address |
---------|------------|------------------|
ldaa #u  | immediate  | No EA            |
ldaa u   | direct     | EA is 8-bit address |
ldaa U   | extended   | EA is a 16-bit address |
ldaa m,r | 5-bit index| EA=r+m (-16 to 15) |
ldaa v,+,r| pre-inc    | r=r+v, EA=r (1 to 8) |
ldaa v,–r| pre-dec    | r=r–v, EA=r (1 to 8) |
ldaa v,r+| post-inc   | EA=r, r=r+v (1 to 8) |
ldaa v,r–| post-dec   | EA=r, r=r–v (1 to 8) |
ldaa A,r  | Reg A offset| EA=r+A, zero padded |
ldaa B,r  | Reg B offset| EA=r+B, zero padded |
ldaa D,r  | Reg D offset| EA=r+D            |
ldaa q,r  | 9-bit index| EA=r+q           |
ldaa W,r  | 16-bit index| EA=r+W           |
ldaa [D,r]| D indirect | EA=[r+D]       |
ldaa [W,r]| indirect   | EA=[r+W]        |

tbeq Y,loop

=xgdx exchange RegD with RegX
=xgdy exchange RegD with RegY

Freescale 6812 addressing modes r is X, Y, SF, or FC

Pseudo op | Meaning                      |
-----------|-----------------------------|
= equ     set           | Define a constant symbol |
dc.b db fcb .byte    | Allocate byte(s) with values |
dcc             | Create an ASCII string      |
dc.w dw fdb .word   | Allocate word(s) with values |
dc.l dl .long      | Allocate 32-bit with values |
dc.w .blkw        | Allocate words without init |
dc.w .blkw        | Allocate words without init |

n is Metrowerks number

Vector | n          | Interrupt | Source Arm |
--------|------------|-----------|------------|
SF0    | Reset      | None      |
SF7    | Trap       | None      |
SF6    | SWI        | None      |
SF0    | Real time interrupt | CRGINT.RTIE |
SFEE   | Timer channel 0 | TIE.C0I   |
SFEC   | Timer channel 1 | TIE.C1I   |
SFEA   | Timer channel 2 | TIE.C2I   |
SFDE   | Timer channel 3 | TIE.C3I   |
SFDE   | Timer channel 4 | TIE.C4I   |
SFEE   | Timer channel 5 | TIE.C5I   |
SFDE   | Timer channel 6 | TIE.C6I   |
SFDE0  | Timer channel 7 | TIE.C7I   |
SFDE   | Timer overflow | TSCR2.TO1 |
SFDE   | SCI0 TDRE, RDRF | SCI0CR2.TIE,RIE |
SFDE4  | SCI1 TDRE, RDRF | SCI1CR2.TIE,RIE |
SFCE   | Key Wakeup J  | PIEJ,[7,6,1,0] |
SFCC   | Key Wakeup H  | PIEH,[7:0] |
SFDE8E | Key Wakeup P  | PIEP,[7:0] |

Interrupt Vectors and interrupt number.
<table>
<thead>
<tr>
<th>Address</th>
<th>Bit 7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Bit 0</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0040</td>
<td>IOS7</td>
<td>IOS6</td>
<td>IOS5</td>
<td>IOS4</td>
<td>IOS3</td>
<td>IOS2</td>
<td>IOS1</td>
<td>IOS0</td>
<td>TIOS</td>
</tr>
<tr>
<td>0044-5</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TCNT</td>
</tr>
<tr>
<td>0046</td>
<td>TEN</td>
<td>TSWAI</td>
<td>TSFRZ</td>
<td>TFFCA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>TSCR1</td>
</tr>
<tr>
<td>004C</td>
<td>C7I</td>
<td>C6I</td>
<td>C5I</td>
<td>C4I</td>
<td>C3I</td>
<td>C2I</td>
<td>C1I</td>
<td>C0I</td>
<td>TIE</td>
</tr>
<tr>
<td>004D</td>
<td>TOI</td>
<td>0</td>
<td>PUPT</td>
<td>RDPT</td>
<td>TCRE</td>
<td>PR2</td>
<td>PR1</td>
<td>PR0</td>
<td>TSCR2</td>
</tr>
<tr>
<td>004E</td>
<td>C7F</td>
<td>C6F</td>
<td>C5F</td>
<td>C4F</td>
<td>C3F</td>
<td>C2F</td>
<td>C1F</td>
<td>C0F</td>
<td>TFLG1</td>
</tr>
<tr>
<td>004F</td>
<td>TOF</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>TFLG2</td>
</tr>
<tr>
<td>0050-1</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC0</td>
</tr>
<tr>
<td>0052-3</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC1</td>
</tr>
<tr>
<td>0054-5</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC2</td>
</tr>
<tr>
<td>0056-7</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC3</td>
</tr>
<tr>
<td>0058-9</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC4</td>
</tr>
<tr>
<td>005A-B</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC5</td>
</tr>
<tr>
<td>005C-D</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC6</td>
</tr>
<tr>
<td>005E-F</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td>TC7</td>
</tr>
<tr>
<td>0082</td>
<td>ADPU</td>
<td>AFFC</td>
<td>ASWAI</td>
<td>ETRIGLE</td>
<td>ETRIGP</td>
<td>ETRIG</td>
<td>ASCIE</td>
<td>ASCIF</td>
<td>ATD0CTL2</td>
</tr>
<tr>
<td>0083</td>
<td>0</td>
<td>S8C</td>
<td>S4C</td>
<td>S2C</td>
<td>S1C</td>
<td>FIF0</td>
<td>FRZ1</td>
<td>FRZ0</td>
<td>ATD0CTL3</td>
</tr>
<tr>
<td>0084</td>
<td>SRES8</td>
<td>SMP1</td>
<td>SMP0</td>
<td>PRS4</td>
<td>PRS3</td>
<td>PRS2</td>
<td>PRS1</td>
<td>PRS0</td>
<td>ATD0CTL4</td>
</tr>
<tr>
<td>0085</td>
<td>DJM</td>
<td>DSGN</td>
<td>SCAN</td>
<td>MULT</td>
<td>0</td>
<td>CC</td>
<td>CB</td>
<td>CA</td>
<td>ATD0CTL5</td>
</tr>
<tr>
<td>0086</td>
<td>SCF</td>
<td>0</td>
<td>EORF</td>
<td>FFOR</td>
<td>0</td>
<td>CC2</td>
<td>CC1</td>
<td>CC0</td>
<td>ATD0STAT0</td>
</tr>
<tr>
<td>008B</td>
<td>CCF7</td>
<td>CCF6</td>
<td>CCF5</td>
<td>CCF4</td>
<td>CCF3</td>
<td>CCF2</td>
<td>CCF1</td>
<td>CCF0</td>
<td>ATD0STAT1</td>
</tr>
<tr>
<td>008D</td>
<td>Bit 7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>008F</td>
<td>PAD07</td>
<td>PAD06</td>
<td>PAD05</td>
<td>PAD04</td>
<td>PAD03</td>
<td>PAD02</td>
<td>PAD01</td>
<td>PAD00</td>
<td>PORTAD0</td>
</tr>
<tr>
<td>0090-1</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0092-3</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0094-5</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0096-7</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0098-9</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>009A-B</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>009C-D</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>009E-F</td>
<td>Bit 15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>009C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SBR12</td>
<td>SBR11</td>
<td>SBR10</td>
<td>SBR0</td>
<td>SC10BD</td>
</tr>
<tr>
<td>009A</td>
<td>LOOPS</td>
<td>SCI1SWAI</td>
<td>RSRC</td>
<td>M</td>
<td>WAKE</td>
<td>ILT</td>
<td>PE</td>
<td>PT</td>
<td>SC10CR1</td>
</tr>
<tr>
<td>009B</td>
<td>TIE</td>
<td>TCIE</td>
<td>RIE</td>
<td>ILIE</td>
<td>TE</td>
<td>RE</td>
<td>RWU</td>
<td>SBK</td>
<td>SC10CR2</td>
</tr>
<tr>
<td>009C</td>
<td>TDRE</td>
<td>TC</td>
<td>RDRF</td>
<td>IDLE</td>
<td>OR</td>
<td>NF</td>
<td>FE</td>
<td>PF</td>
<td>SC10SR1</td>
</tr>
<tr>
<td>009D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BRK13</td>
<td>TXDIR</td>
<td>RAF</td>
<td>SC10SR2</td>
</tr>
<tr>
<td>009F</td>
<td>R7/T7</td>
<td>R6/T6</td>
<td>R5/T5</td>
<td>R4/T4</td>
<td>R3/T3</td>
<td>R2/T2</td>
<td>R1/T1</td>
<td>R0/T0</td>
<td>SC10DR1</td>
</tr>
<tr>
<td>0100-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SBR12</td>
<td>SBR11</td>
<td>SBR10</td>
<td>SBR0</td>
<td></td>
<td>SC11BD</td>
</tr>
<tr>
<td>0102</td>
<td>LOOPS</td>
<td>SCI1SWAI</td>
<td>RSRC</td>
<td>M</td>
<td>WAKE</td>
<td>ILT</td>
<td>PE</td>
<td>PT</td>
<td>SC11CR1</td>
</tr>
<tr>
<td>0103</td>
<td>TIE</td>
<td>TCIE</td>
<td>RIE</td>
<td>ILIE</td>
<td>TE</td>
<td>RE</td>
<td>RWU</td>
<td>SBK</td>
<td>SC11CR2</td>
</tr>
<tr>
<td>0104</td>
<td>TDRE</td>
<td>TC</td>
<td>RDRF</td>
<td>IDLE</td>
<td>OR</td>
<td>NF</td>
<td>FE</td>
<td>PF</td>
<td>SC11SR1</td>
</tr>
<tr>
<td>0105</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BRK13</td>
<td>TXDIR</td>
<td>RAF</td>
<td>SC11SR2</td>
</tr>
<tr>
<td>0107</td>
<td>R7/T7</td>
<td>R6/T6</td>
<td>R5/T5</td>
<td>R4/T4</td>
<td>R3/T3</td>
<td>R2/T2</td>
<td>R1/T1</td>
<td>R0/T0</td>
<td>SC11DR1</td>
</tr>
<tr>
<td>0102</td>
<td>PJ7</td>
<td>PT6</td>
<td>PT5</td>
<td>PT4</td>
<td>PT3</td>
<td>PT2</td>
<td>PT1</td>
<td>PT0</td>
<td>PTT</td>
</tr>
<tr>
<td>0124</td>
<td>DDR7</td>
<td>DDR6</td>
<td>DDR5</td>
<td>DDR4</td>
<td>DDR3</td>
<td>DDR2</td>
<td>DDR1</td>
<td>DDR0</td>
<td>DDRT</td>
</tr>
<tr>
<td>0128</td>
<td>PS7</td>
<td>PS6</td>
<td>PS5</td>
<td>PS4</td>
<td>PS3</td>
<td>PS2</td>
<td>PS1</td>
<td>PS0</td>
<td>PTS</td>
</tr>
<tr>
<td>0124</td>
<td>DDRS7</td>
<td>DDRS6</td>
<td>DDRS5</td>
<td>DDRS4</td>
<td>DDRS3</td>
<td>DDRS2</td>
<td>DDRS1</td>
<td>DDRS0</td>
<td>DDRT</td>
</tr>
<tr>
<td>0125</td>
<td>PM7</td>
<td>PM6</td>
<td>PM5</td>
<td>PM4</td>
<td>PM3</td>
<td>PM2</td>
<td>PM1</td>
<td>PM0</td>
<td>PTM</td>
</tr>
<tr>
<td>0125</td>
<td>DDRM7</td>
<td>DDRM6</td>
<td>DDRM5</td>
<td>DDRM4</td>
<td>DDRM3</td>
<td>DDRM2</td>
<td>DDRM1</td>
<td>DDRM0</td>
<td>DDRM</td>
</tr>
<tr>
<td>0128</td>
<td>PP7</td>
<td>PP6</td>
<td>PP5</td>
<td>PP4</td>
<td>PP3</td>
<td>PP2</td>
<td>PP1</td>
<td>PP0</td>
<td>PTP</td>
</tr>
<tr>
<td>0125</td>
<td>DDRP7</td>
<td>DDRP6</td>
<td>DDRP5</td>
<td>DDRP4</td>
<td>DDRP3</td>
<td>DDRP2</td>
<td>DDRP1</td>
<td>DDRP0</td>
<td>DDRT</td>
</tr>
<tr>
<td>0126</td>
<td>PH7</td>
<td>PH6</td>
<td>PH5</td>
<td>PH4</td>
<td>PH3</td>
<td>PH2</td>
<td>PH1</td>
<td>PH0</td>
<td>PTH</td>
</tr>
<tr>
<td>0126</td>
<td>DDRH7</td>
<td>DDRH6</td>
<td>DDRH5</td>
<td>DDRH4</td>
<td>DDRH3</td>
<td>DDRH2</td>
<td>DDRH1</td>
<td>DDRH0</td>
<td>DDRH</td>
</tr>
<tr>
<td>0128</td>
<td>PJ7</td>
<td>PJ6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PJ1</td>
<td>PJ0</td>
<td>PTJ</td>
</tr>
<tr>
<td>0126</td>
<td>DDRJ7</td>
<td>DDRJ6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>DDRJ1</td>
<td>DDRJ0</td>
<td>DDRJ</td>
</tr>
</tbody>
</table>

**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**

**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)
**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, \( \text{TCNT} = 8\text{MHz}/2^n \), with PLL, \( \text{TCNT} = 24\text{MHz}/2^n \), \( n \) ranges from 0 to 7

<table>
<thead>
<tr>
<th>PR2</th>
<th>PR1</th>
<th>PR0</th>
<th>Divide by</th>
<th>E = 8 MHz</th>
<th>E = 24 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>TCNT</td>
<td>TCNT</td>
<td>TCNT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>period</td>
<td>frequency</td>
<td>period</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>frequency</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>125 ns</td>
<td>41.7 ns</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>250 ns</td>
<td>83.3 ns</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>500 ns</td>
<td>167 ns</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>1 ( \mu )s</td>
<td>333 ns</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>2 ( \mu )s</td>
<td>667 ns</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>4 ( \mu )s</td>
<td>1.33 ( \mu )s</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>64</td>
<td>8 ( \mu )s</td>
<td>2.67 ( \mu )s</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>128</td>
<td>16 ( \mu )s</td>
<td>5.33 ( \mu )s</td>
</tr>
</tbody>
</table>

**SCI0DRL** 8-bit SCI0 data register

**SCI0BD** is 16-bit SCI0 baud rate register, let \( n \) be the 13-bit number  Baud rate is \( \text{EClk}/n/16 \)

**SCI0CR1** is 8-bit SCI0 control register

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

**SCI0CR2** is 8-bit SCI0 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.

**SCI0SR1** is 8-bit SCI0 status register

bit 7 TDRE, Transmit Data Register Empty Flag

Set if transmit data can be written to **SCI0DRL**

Cleared by **SCI0SR1** read with TDRE set followed by **SCI0DRL** write

bit 5 RDRF, Receive Data Register Full

Set if a received character is ready to be read from **SCI0DRL**

Clear the RDRF flag by reading **SCI0SR1** with RDRF set and then reading **SCI0DRL**

**ATD0CTL5** is used to start an ADC conversion

bit 7 DJM is set to 1 for right justified and to 0 for left justified

bits 2-0 specify the ADC channel to sample

**ATD0STAT0** is used to tell when the ADC conversion is done

bit 7 SCF cleared on a write to **ATD0CTL5** and is set when the conversion sequence is done

---

**BSR**   
**Branch to Subroutine**

<table>
<thead>
<tr>
<th>Operation:</th>
<th>((SP) \rightarrow \text{SP}0002 \Rightarrow SP)</th>
<th>(\text{RTN}<em>H : \text{RTN}<em>L \Rightarrow M</em>{\text{SP}} : M</em>{\text{SP}+1})</th>
<th>((\text{PC}) + \text{Rel} \Rightarrow \text{PC})</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Address Mode</th>
<th>Object Code</th>
<th>HCS12</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BSR rel0</strong></td>
<td>REL</td>
<td>07 rr</td>
<td>$FFF</td>
</tr>
</tbody>
</table>

**EMAXM**  
**Place Larger of Two Unsigned 16-Bit Values in Memory**

<table>
<thead>
<tr>
<th>Operation:</th>
<th>(\text{MAX} ((D), (M : M + 1)) \Rightarrow M : M + 1)</th>
</tr>
</thead>
</table>

**EMINM**  
**Place Smaller of Two Unsigned 16-Bit Values in Memory**

<table>
<thead>
<tr>
<th>Operation:</th>
<th>(\text{MIN} ((D), (M : M + 1)) \Rightarrow M : M + 1)</th>
</tr>
</thead>
</table>