Jonathan W. Valvano May 12, 2004, 9am-12noon

This is a closed book exam. You must put your answers in the boxes on the answer pages. You have 3 hours, so please allocate your time accordingly. ***Please read the entire quiz before starting.***

(4) Question 1. Syntactically, I/O ports are public globals. In order to separate mechanisms from policies (i.e., improve the quality of the software system), how should I/O be actually used?

A) Local in allocation D) Private in scope
B) Global in allocation E) Volatile
C) Public in scope F) Nonvolatile

(4) Question 2. Step 1) the I bit in the CCR is set to one. Step 2) output compare interrupt 5 is armed. Step 3) the `TCNT` matches `TC5` setting `C5F`. Step 4) the I bit in the CCR is cleared to zero. Step 5) the `TCNT` counts all the way around and matches `TC5` again setting `C5F`. When is the first interrupt?

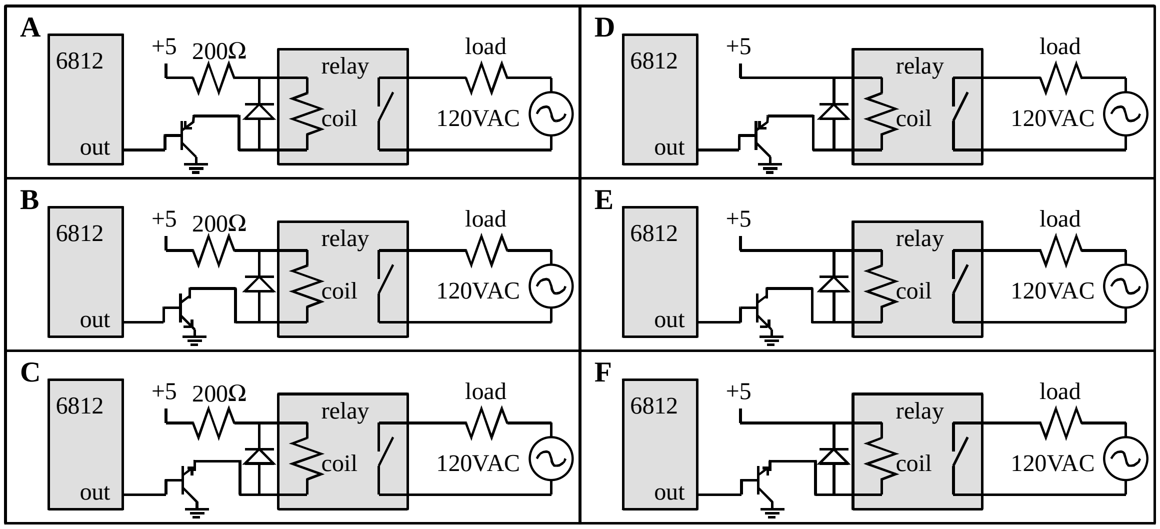
A) No interrupt occurs B) After step 1
C) After step 2 D) After step 3
E) After step 4 F) After step 5

(4) Question 3. What happens if an interrupt service routine does not acknowledge or disarm?

A) Software crashes because no more interrupts will be requested
B) The next interrupt is lost
C) This interrupt is lost
D) Software crashes because interrupts are requested over and over.

(4) Question 4. An electromagnetic **relay** can be used to switch 120 VAC power to a load. For example, the load might be an AC motor. To activate the relay (apply power to the motor), you must deliver between 4V and 5V to the relay coil. The relay coil impedance is 50Ω in series with 1mH. To deactivate the relay, the relay coil current should be zero. Assume V_{CE} of the transistor is 0.5V.

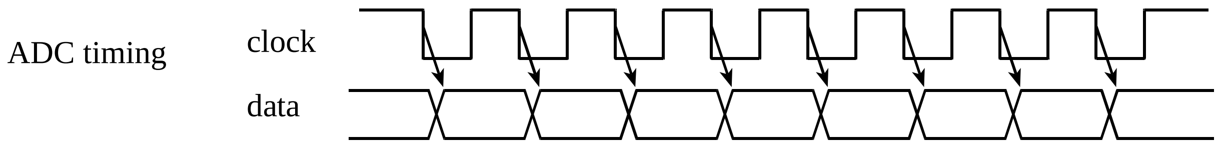
Part a) Choose the proper interface circuit.



Part b) What is the minimum $\mathbf{I_{CE}}$ needed for the transistor?

(4) Question 5. An 11-bit ADC has an input range of –5V to +5V. What is the ADC resolution?

(4) Question 6. The ADC serial data output, shown in the figure below, is connected to the SPI MISO, serial data input. The 6812 is the master and the ADC is the slave. The ADC clock input, connected to the SPI clock output, is normally high (when idle the clock is 1). After a conversion, the ADC shifts its new data out on the falling edge of the clock. What values of **`CPOL`**, **`CPHA`** should be used?



(4) Question 7. Write the C code to implement the following equation using fixed-point math. **`X`** **`Y`** and **`Z`** are the integer parts (the variables stored in the computer) for 8-bit unsigned binary fixed-point numbers with a resolution of 1/256. . Think about if overflow can occur. If overflow were to occur, set the **`Z`** equal to its maximum positive value.

```
Z = X*Y
```

(4) Question 8. There are 10 points to the IEEE Code of Ethics. What is the basic premise of the first point of this code? Give one specific example of how this might apply to embedded systems.

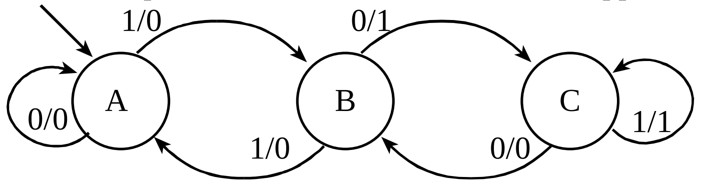
(4) Question 9. An unsigned fixed-point system has a range of values from 0 to 999.99 with a resolution of 10^{-2}. Note: 10^{-2} equals 0.01. With which of the following data types should the software variables be allocated? When more than one answer is possible choose the most space efficient type.

A) `unsigned char`	D) `char`	G) `float`
B) `unsigned short`	E) `short`	H) `double`
C) `unsigned long`	F) `long`	

(4) Question 10. The software writes a 2 to the **`ATDCTL5`**. After the **`SCF`** flag is set, what is in the MC68HC812A4 **`ADR1H`** or 9S12C32 **`ATDDR1`** register?

A) nothing	D) ADC conversion result for channel 0
B) 0x80	E) ADC conversion result for channel 1
C) 0x00	F) ADC conversion result for channel 2

(4) Question 11. Consider the following Mealy FSM, where the initial state is A. The labels on the arrows mean input/output. If the input were to be a constant 1, what happens?



A) Eventually the system ends up in state C with the output high.
B) The system oscillates between state A and state B with the output low.
C) Eventually the system ends up in state A with the output low.
D) The system oscillates between state A and state B with the output toggling high and low.
E) The system oscillates between state B and state C with the output toggling high and low.
F) None of the above.

(4) Question 12. The following code was used to acknowledge a timer channel 7 interrupt. Which explanation best describes this code?

```
    TFLG1 |= 0x80;
```

A) This software only makes the **C7F** bit high. It is friendly.
B) This software only makes the **C7F** bit low. It is friendly.
C) This software will make all flag bits low in the **TFLG1** register. It is not friendly.
D) This software will make all flag bits high in the **TFLG1** register. It is not friendly.
E) This will cause a compile error because the software can not set flag bits in the **TFLG1** register.
F) This will cause a run-time error because the software can not set flag bits in the **TFLG1** register.

(4) Question 13. Consider the following C program, which is implemented on an embedded system. Where are each of the four variables stored? For each variable specify A, B or C:

A) Global RAM means permanently allocated at a fixed location in volatile memory.
B) Stack RAM means temporarily allocated, used, then deallocated in volatile memory.
C) EEPROM means permanently allocated at a fixed location in nonvolatile memory.

Please note that the variable names in this example do not follow the standard naming conventions.

```
const char v1=100;
static char v2=10;
char add3(const char v3){
static char v4;
  v4 = v1+v3;
  return(v4);
}
void main(void){
  v2 = add3(v2);
}
```

Part a) Where is **v1** allocated?
Part b) Where is **v2** allocated?
Part c) Where is **v3** allocated?
Part d) Where is **v4** allocated?

(4) Question 14. Does the associative principle hold for signed integer addition and subtraction? In particular do these two C calculations always achieve identical outputs? If no, give an example.

```
    Out1 = (A+B)-C;
    Out2 = A+(B-C);
```

(4) Question 15. Does the associative principle hold for signed integer multiply and divide? In particular do these two C calculations always achieve identical outputs? If no, give an example.

```
    Out3 = (A*B)/C;
    Out4 = A*(B/C);
```

(4) Question 16. Consider the following interface between two 6812s. One 6812 is master and the other is a slave. Assume the SPI clock frequency is 1 MHz. To communicate, the following sequence of steps occur in this order

1) The slave puts 8-bit data in its SPI data register
2) The master puts 8-bit data in its SPI data register
3) The two SPI hardware systems active transmitting the data
4) The slave reads its SPI data register, getting the value sent by the master
5) The master reads its SPI data register, getting the value sent by the slave

Part a) Is this communication protocol synchronous or asynchronous?
Part b) Is this communication protocol simplex, half-duplex or full-duplex?
Part c) Assuming the software runs much faster than the SPI hardware, what is the maximum bandwidth communicated in this system, in bytes/sec.

(4) Problem 17. Consider the following TOF interrupting system with its corresponding assembly code generated by the Metrowerks compiler. Assume at the time of the first instruction of `main`, there are exactly two (2) bytes pushed on the stack. In other words, after `main` executes `PSHD`, there will be 4 bytes on the stack. Calculate the maximum number of bytes that will be pushed on the stack at any given point as this system executes. This is all the software.

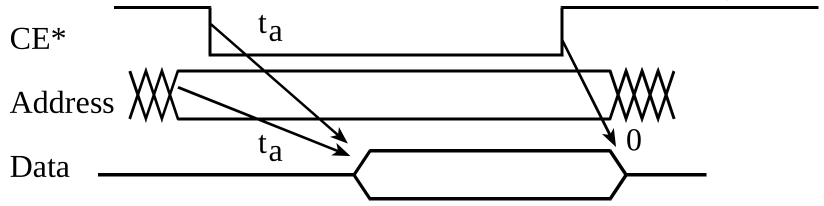
```
unsigned short time;
interrupt 16 void TOFhndlr(void){
  time++;
  TFLG2 = 0x80;
}

void TOFinit(void){
  time = 0;
  COPCTL = 0;
  TSCR = 0x80;
  TMSK2 = 0x85;
  TFLG2 = 0x80;
asm cli
}

short calc(short a, short b){
  return a*b;
}

void main(void){  short c;
  TOFinit();
  c = 1;
  for(;;) {
    c = calc(c,time);
  }
}
```

```
TOFhndlr:
      LDX   time
      INX
      STX   time
      LDAB  #128
      STAB  TFLG2
      RTI
TOFinit:
      CLRB
      CLRA
      STD   time
      STAB  COPCTL
      LDAB  #128
      STAB  TSCR
      LDAB  #133
      STAB  TMSK2
      LDAB  #128
      STAB  TFLG2
      CLI
      RTS
calc: PSHD
      LDD   4,SP
      LDY   0,SP
      EMUL
      PULX
      RTS
main:    <= start execution here
      PSHD
      BSR   TOFinit
      LDAB  #1
      CLRA
      STD   0,SP
      PSHD
      LDAA  time
      LDAB  time+1
      BSR   calc
      LEAS  2,SP
      BRA   *-10
```

(2) Problem 18. Are there any critical sections in the software system shown in the previous problem? If so, state where the critical section is.

(4) Question 19. Assume an expanded mode MC68HC12A4 is initialized and running with three (3) cycle stretches. A RAM is interfaced with 6812 **CSD** connected to **CE*** and 6812 **R/W** connected to RAM **WE***. To write data into this RAM, both **CE*** and **WE*** must be zero. The data will be clocked into the RAM during a write cycle on the rise of **CE*** or the rise of **WE*** whichever occurs earlier. The setup time for this write event is $\mathbf{t_{su}}$, and the hold time is $\mathbf{t_h}$.
Part a) What is the maximum possible value for the set up time, $\mathbf{t_{su}}$? (assuming three stretches)
Part b) What is the maximum possible value for the hold time, $\mathbf{t_h}$? (assuming three stretches)

(4) Question 20. A ROM is interfaced to MC68HC812A4 running at 8MHz. CSP0 is connected to the ROM CE*. $\mathbf{t_a}$ has a minimum of 100ns and a maximum of 200ns. How many stretches are needed?



(16) Question 21. You are asked to write two public functions: **`Tx_Init`** and **`Tx_Out`**, and a SCI interrupt handler, which will implement SCI transmission using interrupt synchronization.

```
void Tx_Init(void);      // initialize transmitter
short Tx_Out(char *pt);  // output a null-terminated ASCII string
```

The serial protocol should be 2400 bits/sec baud rate, 1 start bit, 8 data bits, even parity, and 1 stop bit. The SCI receiver will NOT be used and it should be disabled to save power. You can add one or two private global variables, but otherwise no additional data structures are allowed. In particular, you will not be using a Fifo queue. Instead, you will use the single buffer that is passed by reference into **`Tx_Out`**. If the user program calls **`Tx_Out`** before the previous string has been completely transmitted **`Tx_Out`** will return with a 1. If the user program calls **`Tx_Out`** while the transmitter is idle, transmission will be started and **`Tx_Out`** will return with a 0. *There are no backward jumps (***`while do for`***) in any of the code you are writing.* The following main program illustrates the usage of your device driver. You may assume the ASCII string is available for the ISR for the duration of transmission.

```
const char NewLine[3]={13,10,0}; // CR,LF,null
void main(void){
  Tx_Init();  // enable SCI transmitter
  while(1){
    while(Tx_Out("Hello world"));
    while(Tx_Out(NewLine));
    while(Tx_Out("That was fun, let's do it again!"));
    while(Tx_Out(NewLine));
  }
}
```

(2) Problem 22. How would you characterize the system implemented in the previous question?

A) CPU bound	B) Nonreentrant
C) I/O bound	D) Round Robin

The syntax **PORTT/PTT** means **PORTT** is the MC68HC812A4 name while **PTT** is the 9S12C32 name.
TCNT is 16-bit up counter
TCn are 16-bit input capture/output compare latch registers, n=0 to 7
PORTT/PTT is 8-bit bi-directional I/O port
DDRT is the associated direction register for Port T (0 means input, 1 means output)
TIOS is 8-bit input/output select (0 means input capture, 1 means output compare)
TSCR/TSCR1 is a timer control register
    bit 7 **TEN**, 1 means allow timer to function normally, 0 means disable timer including **TCNT**
TFLG1 is 8-bit timer flag register
    set by input capture or output compare event
    cleared by write to this register with bit set
TFLG2 is 8-bit timer flag register
    bit 7 **TOF** timer overflow interrupt flag, set on **TCNT** overflow, cleared by write to this register with bit set
TMSK1/TIE is 8-bit timer arm register
    1 means corresponding bit in **TFLG1** will request an interrupt
    1 means corresponding bit in **TFLG1** will not request an interrupt
TMSK2/TSCR2 is 8-bit timer control register
    bit 7 **TOI** timer overflow interrupt enable, 1 = interrupt on TOF, 0 = TOF interrupts will not occur
    bits 2,1,0 **PR2**, **PR1**, **PR0**, select rate, let **n** be the 3-bit number
    MC68HC812A4 TCNT frequency is 8MHz/$2^{\mathbf{n}}$, **n** ranges from 0 to 5
    9SC12 TCNT frequency is 24MHz/$2^{\mathbf{n}}$, **n** ranges from 0 to 7
TCTL3 is 8-bit timer control register, input capture mode (00=off, 01=rise, 10=fall, 11=both rise and fall)
    bits 7-6 **EDG7B,EDG7A** input capture 7 edge
    bits 5-4 **EDG6B,EDG6A** input capture 6 edge
    bits 3-2 **EDG5B,EDG5A** input capture 5 edge
    bits 1-0 **EDG4B,EDG4A** input capture 4 edge
TCTL4 is 8-bit timer control register, input capture mode (00=off, 01=rise, 10=fall, 11=both rise and fall)
    bits 7-6 **EDG3B,EDG3A** input capture 3 edge
    bits 5-4 **EDG2B,EDG2A** input capture 2 edge
    bits 3-2 **EDG1B,EDG1A** input capture 1 edge
    bits 1-0 **EDG0B,EDG0A** input capture 0 edge
MC68HC812A4 **RTICTL** real time interrupt control register, M clock is 8 MHz
    bit 7 **RTIE** real time interrupt enable, 1 means interrupt on RTIF, 0 means RTI interrupts will not occur
    bits 2,1,0 **RTR2**, **RTR1**, **RTR0**, select rate, let **n** be the 3-bit number, **n** ranges from 1 to 7
        interrupt period is 512µs*$2^{\mathbf{n}}$
RTIFLG/CRGFLG real time interrupt flag register
    bit 7 **RTIF** real time interrupt flag, set on RTI timeout, cleared by write to this register with bit set
9S12C32 **CRGINT** real time interrupt control register
    bit 7 **RTIE** real time interrupt enable, 1 means interrupt on RTIF, 0 means RTI interrupts will not occur
9S12C32 **RTICTL** real time interrupt control register, M clock is 4 MHz
    bits 6-4 **RTR6**, **RTR5**, **RTR4**, select rate, let **n** be the 3-bit number, **n** ranges from 1 to 7
    bits 3-0 **RTR3**, **RTR2**, **RTR1**, **RTR0**, select rate, let **m** be the 4-bit number, **m** ranges from 0 to 7
        interrupt period is 128µs*(**m**+1)*$2^{\mathbf{n}}$
SP0CR1/SPICR1 SPI control register
    bit 6 **SPE** — SPI System Enable
        0 = SPI internal hardware is initialized and SPI system is in a low-power disabled state.
        1 = SPI function enabled
    bit 4 **MSTR** — SPI Master/Slave Mode Select
        0 = Slave mode
        1 = Master mode
    bits 3-2 **CPOL**, **CPHA** — SPI Clock Polarity, Clock Phase
    bit 1 **SSOE** — Slave Select Output Enable

MC68HC812A4 **SP0BR** SPI baud rate

bits 2,1,0 **SPR2**, **SPR1**, **SPR0**, select rate, let **n** be the 3-bit number, **n** ranges from 0 to 7

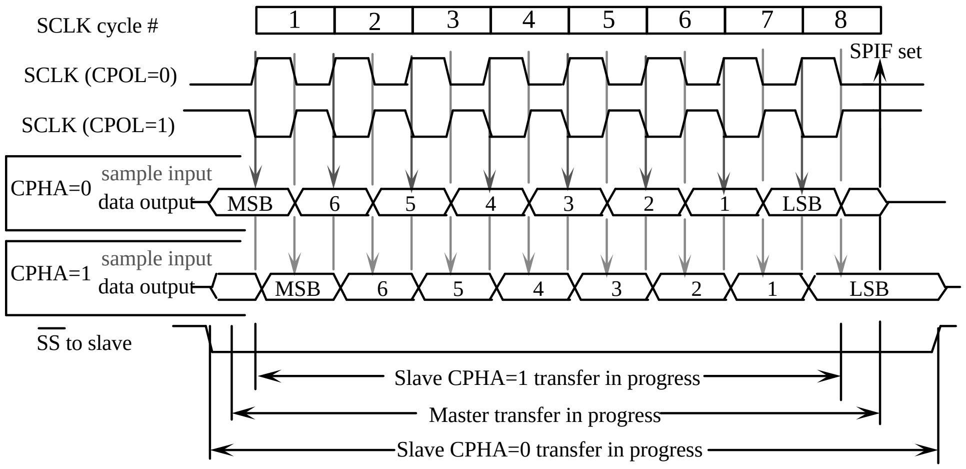
SPI clock period is 4MHz/2^n

9S12C32 **SPIBR** SPI baud rate

bits 6-4 **SPPR2**, **SPPR1**, **SPPR0**, select rate, let **m** be the 3-bit number, **m** ranges from 0 to 7

bits 2,1,0 **SPR2**, **SPR1**, **SPR0**, select rate, let **n** be the 3-bit number, **n** ranges from 0 to 7

SPI clock frequency is 12MHz/(**m**+1)/2^n



SP0SR/SPISR SPI control register

bit 7 **SPIF** set after the eighth SCK cycle in a data transfer

cleared by reading status register (with SPIF set) followed by read or write to SPI data register.

SP0DR/SPIDR is 8-bit SPI data register

ATDCTL2 ADC control register

bit 7 **ADEN**, set to enable ADC

ATDCTL5 ADC control register

bit 6, S8CM, 0 = four conversion sequence. 1 = eight conversion sequence

bit 5, SCAN, 0 = single sequence of conversions then stop, 1 = continuous conversion

bit 4, MULT, 0 = sequence of conversions on a single channel, 1 = sequence of conversions on multiple channels

bits 2-0, write channel number to start ADC, channel number 0 to 7

ATDSTAT 16-bit ADC status register

bit 15 **SCF**, cleared by write to **ATDCTL5**, set when ADC finished

MC68HC812A4 **ADR0H** first 8-bit ADC result

9S12C32 **ATDDR0** first 10-bit ADC result

SC0DRL/SCIDRL 8 bit data serial data register

SC0BD/SCIBD is 16-bit SCI baud rate register, let **n** be the 16-bit number

MC68HC812A4 Baud rate is 500kHz/**n** 9SC12 Baud rate is 12MHz/**n**

SC0CR1/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

bit 2 PE, Parity Enable, 0 = Parity is disabled, 1 = Parity is enabled.

bit 0 PT, Parity Type, 0 = Even parity is selected, 1 = Odd parity is selected

SC0CR2/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.

SC0SR1/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`** .
bit 3 OR, Receiver Overrun Error Flag
bit 2 NF, Receiver Noise Error Flag, 1 = Noise on a valid start bit, any of the data bits, or on the stop bit
bit 1 FE, Receiver Framing Error Flag, Set when a zero is detected where a stop bit was expected.
Clear the FE flag by reading **`SCISR1`** with FE set and then reading **`SCIDRL`**.
bit 0 PF, Receiver Parity Error Flag, Indicates if received data's parity matches parity bit.

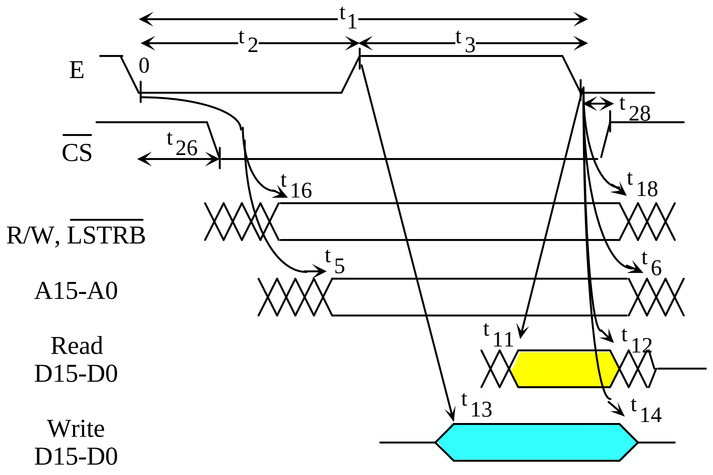


Figure 9.34. Simplified bus timing for the MC68HC812A4 in expanded mode.

Num	Characteristic	0 stretch	1 stretch	2 stretch	3 stretch	Units
t_1	Cycle Time	125	250	375	500	ns
t_2	Pulse Width E low	60 min	60 min	60 min	60 min	ns
t_3	Pulse Width E high	60 min	185 min	310min	435 min	ns
t_5	A15-A0, R/W delay	60 max	60 max	60 max	60 max	ns
t_6	address hold time	20 min	20 min	20 min	20 min	ns
t_{11}	Read data setup time	30 min	30 min	30 min	30 min	ns
t_{12}	Read data hold time	0 min	0 min	0 min	0 min	ns
t_{13}	Write data delay time	46 max	46 max	46 max	46 max	ns
t_{14}	Write data hold time	20 min	20 min	20 min	20 min	ns
t_{16}	R/W delay time	49 max	49 max	49 max	49 max	ns
t_{18}	R/W hold time	20 min	20 min	20 min	20 min	ns
t_{26}	CS delay time	60 max	60 max	60 max	60 max	ns
t_{28}	CS hold time	10 max	10 max	10 max	10 max	ns

Table 9.13. Timing parameters for the MC68HC812A4 with an E clock of 8 MHz.
MC68HC812A4 **`DPAGE`** 8-bit page register MC68HC812A4 **`PPAGE`** 8-bit page register
interrupts vectors

```
0xFFD6    interrupt 20 SCI0/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
0xFFF0    interrupt 7  real time interrupt
```