Jonathan W. Valvano May 14, 2005, 9am-12noon

This is a closed book exam. Other than a 1-page hand-written crib sheet pencils, pens, erasers, no other items are allowed. 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) Problem 1. Consider the following RTI 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. The listing includes absolute addresses.

```
short Num,Result;

short LowPassFilter(const short x){
static short y=0;
  y = (x+y)/2;
  return y;
}




void interrupt 7 handler(){
  CRGFLG = 0x80;
  Num++;
  Result = LowPassFilter(1000);
}




void main(void){short out;
  CRGINT = 0x80;
  RTICTL = 0x33;
  Num = 0;
  asm cli
  for(;;) {
    out = LowPassFilter(Num);
  }
}
```

```
LowPassFilter
  4100 3b          PSHD
  4101 fc3804      LDD   y
  4104 e380        ADDD  0,SP
  4106 ce0002      LDX   #2
  4109 1815        IDIVS
  410b 7e3804      STX   y
  410e b754        TFR   X,D
  4110 30          PULX
  4111 3d          RTS
handler
  4112 c680        LDAB  #128
  4114 5b37        STAB  _CRGFLG
  4116 fe3800      LDX   Num
  4119 08          INX
  411a 7e3800      STX   Num
  411d fc3800      LDD   #1000
  4120 07de        BSR   LowPassFilter
  4122 7c3802      STD   Result
  4125 0b          RTI
Main    ; <=start execution here
  4126 3b          PSHD
  4127 c680        LDAB  #128
  4129 5b38        STAB  _CRGINT
  412b 8633        LDAA  #51
  412d 5a3b        STAA  _RTICTL
  412f c7          CLRB
  4130 87          CLRA
  4131 7c3800      STD   Num
  4134 10ef        CLI
  4136 fc3800      LDD   Num
  4139 07b3        BSR   LowPassFilter
  413b 6c80        STD   0,SP
  413d 20f7        BRA   *-7
```

(4) Problem 2. Is there a critical section in the software system shown in Question 1? If so, state the two assembly instructions between which the critical section exists.

(2) Question 3. What does the **`short`** qualifier in the function **`LowPassFilter()`** mean?

A) unsigned integer
B) signed integer
C) floating point
D) could be either an unsigned integer or an unsigned fixed-point
E) could be either a signed integer or a signed fixed-point
F) depends on the compiler settings

(2) Question 4. What does the **`const`** qualifier in the function **`LowPassFilter()`** mean?

A) private in scope
B) stored in ROM
C) stored in global RAM
D) the value is fixed and can not be changed by the function
E) tells the compiler to fetch a new value, and do not optimize
F) promoted to the next high precision

(2) Question 5. What does the **`static`** qualifier in the function **`LowPassFilter()`** mean?

A) private in scope
B) stored in ROM
C) stored in global RAM
D) the value is fixed and can not be changed by the function
E) tells the compiler to fetch a new value, and do not optimize
F) promoted to the next high precision

(2) Question 6. What is the programming bug in the function **`LowPassFilter()`**?

A) potential overflow in addition
B) y is initialized over and over again
C) type mismatch in **`return y`**
D) type mismatch in adding **`x+y`**
E) uninitialized variable **`y`**
F) there are no errors in this program

(4) Question 7. `LowPassFilter()` is called from an ISR as part of a real-time system. The SCI, **`PTT`** and **`PTAD`** are unused by the system, and **`PTT`** and **`PTAD`** are digital outputs. The debugging code will be placed at the end just before the return, unless otherwise stated. **`SCI_OutSDec`** outputs a 16-bit signed integer. **`BufX`** and **`BufY`** are global buffers of length 100, **`n`** is a global variable initialized to 0. Which debugging code would you add to verify the correctness of this function?

A)
```
asm{ sei};
PTT=x; PTAD=y;
asm{ cli};
```
B)
```
asm{ sei};
SCI_OutSDec(x); SCI_OutSDec(y);  // busy-wait
asm{ cli};
```
C)
```
asm{ sei};
SCI_OutSDec(x); SCI_OutSDec(y);  // interrupt driven
asm{ cli};
```
D)
```
asm{ sei};
if(n<100){BufX[n]=x; BufY[n]=y; n++;}
asm{ cli};
```
E)
```
PTT |= 0x01;    // at beginning
PTT &= ~0x01;   // at end
```
F)
```
PTT ^= 0x01;    // at beginning
PTT ^= 0x02;    // at end
```

(4) Question 8. Determine read data required (RDR) interval for a 6811 running at 1 MHz. Give your answer in numerical form, defining 0ns at the start of the cycle. I.e., **`(starttime, endtime)`**.

(2) Question 9. List the three conditions that must be true for a RTI interrupt to occur

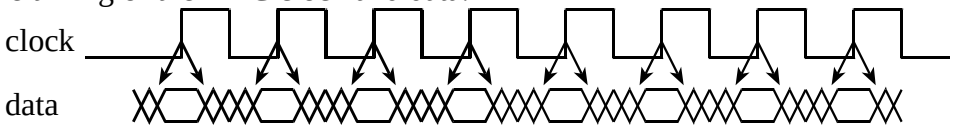
A) The I bit in the CCR is 0
B) The I bit in the CCR is 1
C) **RTIF** bit set by timer hardware
D) Software executes **CRGFLG = 0x80;**
E) Software executes **CRGINT = 0x80;**
F) Software executes the **RTI** instruction

(2) Question 10. An unsigned fixed-point system has a range of values from 0 to 499.99 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**
B) **unsigned short**
C) **unsigned long**
D) **char**
E) **short**
F) **long**
G) **float**
H) **double**

(4) Question 11. A 10-bit ADC has an input range of –10V to +10V. What is the ADC resolution? Please include units in your answer.

(4) Question 12. The DAC serial data input, shown in the figure below, is connected to the SPI MOSI, serial data output. The 6812 is the master and the DAC is the slave. The DAC clock input, connected to the SPI clock output, is normally low (when idle the clock is 0). After receiving a new 8-bit data from the 6812, the DAC sets its analog output. What values of **CPOL**, **CPHA** should be used? The following figure shows the timing of the DAC clock and data.



(4) Question 23. What does the software do to start an ADC conversion?

A) nothing, it starts automatically
B) By clearing the SCF flag
C) By reading **ATDDR0**
D) By reading **ATDCTL5**
E) By setting bit 7 of **ATDCTL2**
F) By writing to **ATDCTL5**

(4) Question 24. Does the associative principle hold for unsigned integer multiply and divide? In particular do these two C calculations always achieve identical outputs? If no, give an example.

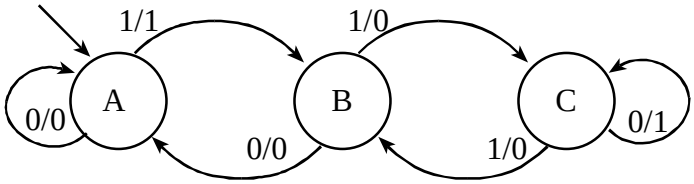
```
Out3 = (A*B)/C;
Out4 = A*(B/C);
```

(2) Question 27. The first point of the IEEE Code of Ethics is *to accept responsibility in making engineering decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;* Give one specific example of how this might apply to embedded systems.

For questions 13-22, choose the term that best fits the definition.

(2) Question 13. A communication protocol where just the data (and not the clock) are passed from transmitter to receiver. **(2) Question 14.** A communication system that can transfer data in two directions at the same time. **(2) Question 15.** The difference between the true value and the measured value. **(2) Question 16.** A debugging technique that uses paper and pencil to determine in advance specific input/output behaviors of our software, then runs the system comparing actual results with expected values. **(2) Question 17.** The transfer rate of information, which could be in bits per second, or in bytes per second. **(2) Question 18.** The amount of time from when the output is idle until the time the computer writes new data to the output device.	**(2) Question 19.** The interrupt mechanism, like RDRF and TDRE, where multiple potential interrupt requests share the same interrupt vector, but have separate interrupt flags, separate interrupt arm bits, and separate acknowledge sequences. **(2) Question 20.** A variable or function that can only be accessed by functions within the same module (e.g., functions within the same file). **(2) Question 21.** A debugging term that means the act of debugging itself has no affect whatsoever on the system being tested. **(2) Question 22.** A multithreaded system where the direct operations of input and output occur in background interrupt service routines, the foreground thread (main program) processes inputs and generates new outputs, and FIFO queues are used to pass data between the foreground and background.	A) accuracy B) asynchronous serial C) atomic D) bandwidth E) baud rate F) bit time G) breakpoint H) buffered I/O I) critical section J) desk check K) even parity L) frame M) friendly N) full duplex O) half-duplex P) latency Q) minimally intrusive R) nonintrusive S) nonvolatile T) periodic polling U) polled interrupt V) precision W) private X) profile Y) promotion Z) public AA) range BB) real-time CC) reentrant DD) resolution EE) scanpoint FF) simplex GG) stabilize HH) synchronous serial II) vectored interrupt JJ) vulnerable window

(4) Question 25. Consider this 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, after an initial startup period what eventually happens?



A) The system ends up in state C with the output high.
B) The system oscillates between state B and state C with the output low.
C) 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.

(2) Question 26. The following code was used to acknowledge a timer channel 0 interrupt. Which explanation best describes this code?

```
TFLG1 |= 0x01;
```

A) This software only makes the **C0F** bit high. It is friendly.
B) This software only makes the **C0F** 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 28. Design a minimal-cost positive-logic address decoder for **YourDevice** in the following system. You do not have to design all three, just the decoder for **YourDevice**. Include chip numbers but not pin numbers

RAM	$6000-$67FF	Show 1) design steps,
YourDevice	$6800-$6FFF	2) equation
ROM	$E000-$FFFF	3) circuit.

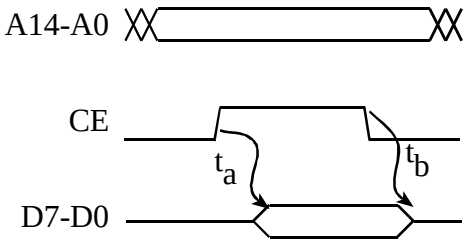
(6) Question 29. A solid-state relay can be used to switch 120 VAC power to a load. For example, the software can turn on/off an AC motor. To activate the relay (apply power to the motor), you must deliver about 2.6 V at 2 mA to the control diode. To deactivate the relay, the diode current should be zero. Design an interface using a 74LS05 to connect this relay to the 6812. Specify resistor values.

V_{IL}	V_{IH}	V_{OL}	I_{IL}	I_{IH}	I_{OL}
0.8 V	2.0 V	0.4 V	0.4 mA	20 µA	4 mA

Specifications for 74LS05

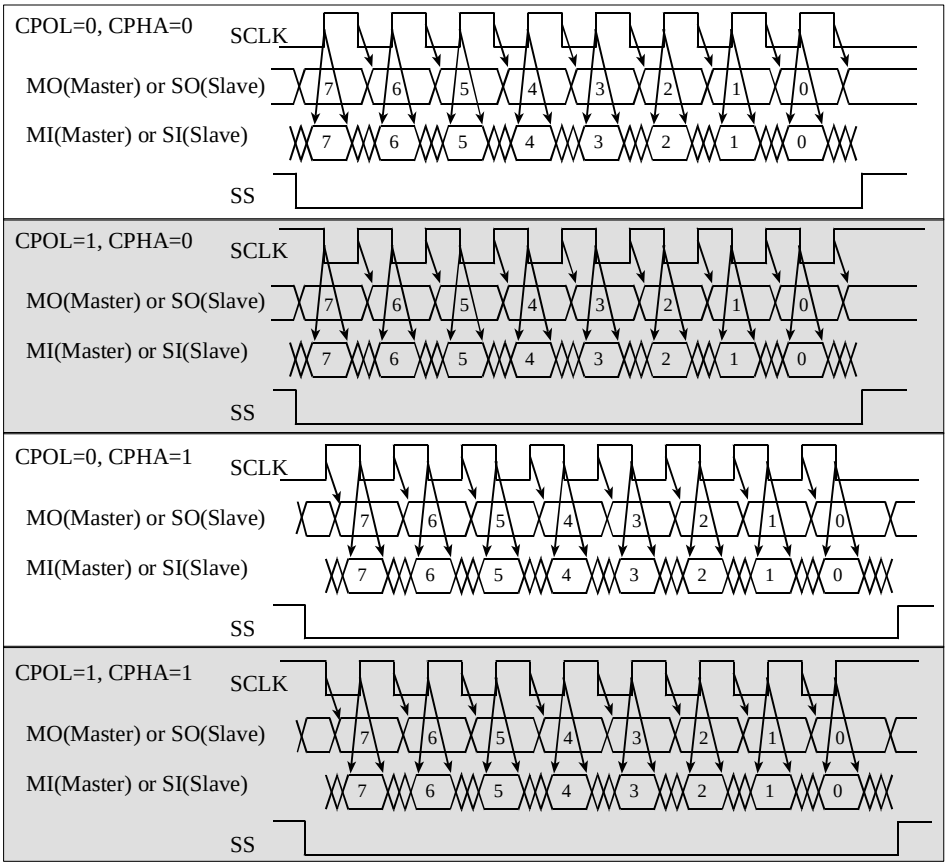
(4) Question 30. Write the C code to implement **Z = X*Y** using fixed-point math. **X Y Z** are the theoretical fixed-point values (not stored in the computer) and **IX IY** and **IZ** are the corresponding integer parts (the global variables stored in the computer) for 16-bit unsigned binary fixed-point numbers with a resolution of 1/16. Think about if overflow can occur. If overflow were to occur, set the **IZ** equal to its maximum positive value.

(8) Question 31. Design the interface between a 32k by 8-bit PROM and a 6811 running at 2 MHz. Implement full address decoding for addresses $8000-$FFFF. Assume t_a = 100 ns, and t_b = 20ns. Assume a 10ns gate delay through any digital logic and the 74HC573 address latch. The PROM has one control signal, CE, which when high reads data at the 15-bit address. Show just the circuit including chip numbers, but not pin numbers. The timing analysis is not required for this question.



(6) Question 32. Write C code that arms and enables an input capture 7 interrupt, so that an interrupt is requested on each rise of the PT7 input. Be friendly, and include comments.

TCNT is 16-bit up counter (see **TSCR2**)
TCn are 16-bit input capture/output compare latch registers, n=0 to 7
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)
TIOS is 8-bit input/output select (0 means input capture, 1 means output compare)
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
TIE is 8-bit timer arm register
1 means corresponding bit in **TFLG1** will request an interrupt
0 means corresponding bit in **TFLG1** will not request an interrupt
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
without PLL TCNT is $4MHz/2^{n}$, with PLL TCNT is $24MHz/2^{n}$, **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
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
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
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 $128\mu s*(\mathbf{m}+1)*2^{\mathbf{n}}$
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
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 $12MHz/(\mathbf{m}+1)/2^{\mathbf{n}}$
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.
SPIDR is 8-bit SPI data register



ATDCTL2 ADC control register, bit 7 **ADPU**, set to enable ADC

ATDCTL4 ADC control register, bit 7 **SRES8**, 1 for 8-bit ADC, 0 for 10-bit ADC

ATDCTL5 ADC control register

bit 7, DJM Justification, 1=right justified, 0=left justified

bit 6, DSGN Signed Representation, 1=signed, 0=unsigned

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

ATDDR0 first 10-bit ADC result

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 12MHz/**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.

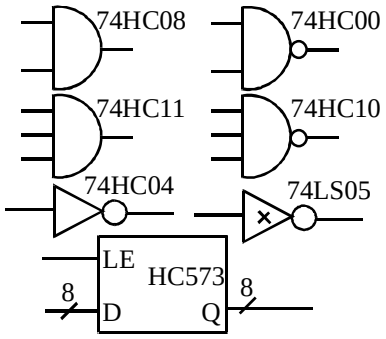
bit 5 RDRF, Receive Data Register Full

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 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
```



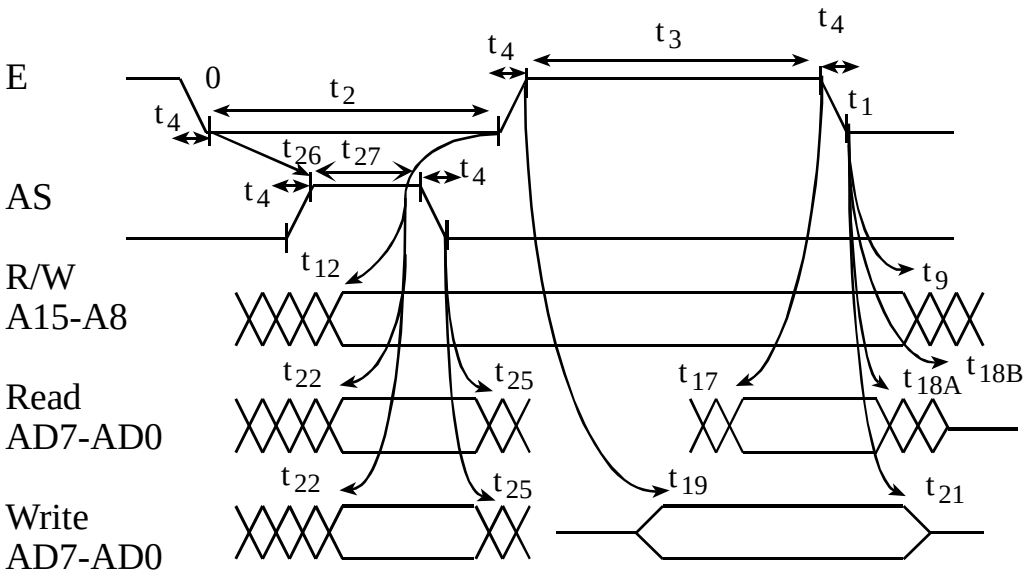


Figure 9.34. Simplified bus timing for the 6811 in expanded mode.

Num	Characteristic	1.0 MHz	**2.0 MHz**	2.1 MHz	Units
	Frequency	1.0	**2.0**	2.1	MHz
t_1	Cycle Time	1000	**500**	476	ns
t_2	Pulse Width E low	480	**230**	218	ns
t_3	Pulse Width E high	480	**230**	218	ns
t_4	rise/fall time	20	**20**	20	ns
t_9	address hold time	95.5 min	**33 min**	30 min	ns
t_{12}	A15-A8,R/W valid time	281.5 min	**94 min**	85 min	ns
t_{17}	Read data setup time	30 min	**30 min**	30 min	ns
t_{18A}	Read data hold time	10 min	**10 min**	10 min	ns
t_{18B}	Read data goes hiZ	145.5 max	**83 max**	80 max	ns
t_{19}	Write data delay time	190.5 max	**128 max**	125 max	ns
t_{21}	Write data hold time	95.5 min	**33 min**	30 min	ns
t_{22}	A7-A0 valid time	271.5 min	**84 min**	75 min	ns
t_{25}	A7-A0 hold time	95.5 min	**33 min**	30 min	ns
t_{26}	E to AS rise time	115.5	**53**	50	ns
t_{27}	AS pulse width	221	**96**	90	ns