(35) **Question 1.** Consider the following simple C program. Assume this code is located in file.c.

```c
short A1; int A2;
short B1; short static B2;
short C1; short volatile C2;
short D1=5; short const D2=5;
short F1(short n){ return n+1;}
short static f2(short n){ return n+1;}
short FreezingPoint1=32; short FreezingPoint2=0x20; // degrees F
void program(void){
  short e1; short static e2;
}
```

(5) **Part a**) What is the difference between \( A1 \) and \( A2 \)?

(5) **Part b**) What is the difference between \( B1 \) and \( B2 \)?

(5) **Part c**) What is the difference between \( C1 \) and \( C2 \)?

(5) **Part d**) What is the difference between \( D1 \) and \( D2 \)?

(5) **Part e**) What is the difference between \( e1 \) and \( e2 \)?

(5) **Part f**) What is the difference between \( F1 \) and \( f2 \)?

(5) **Part g**) What is the difference between \( \text{FreezingPoint1} \) and \( \text{FreezingPoint2} \)?
(15) **Question 2.** Interface an LED to the 6812 Port H bit 5. The LED has a desired operating point of 1.7 V and 5 mA. Show chip numbers and resistor values.

(50) **Question 3.** An ADC converts analog signals into digital numbers. This 8-bit ADC has 32 channels and converts analog 0 to +5V into numbers 0 to 255. For example, if an analog input is 2.5 V, it will be converted to the number 128. A DAC converts numbers to analog outputs. This 8-bit DAC also has 32 channels and converts the numbers 0 to 255 into analog 0 to +5V. For example, if your software write the number 128 to it, the DAC will generate a 2.5 V analog output. The hardware connections are as follows:

- **PH7** --> **ADCstart** your software sets this high to start an ADC conversion
- **PH6** <-- **ADCDone** this signal goes high when the ADC conversion is complete
- **PH5** --> **DACtrigger** your software pulses this signal to cause a DAC conversion
- **PH4-0** --> **channel** 5-bit number specifying which of the 32 channels to convert
- **PJ7-PJ0** <-- **data** bidirectional, output for DAC, input for ADC

To output a number to the DAC, your software should use blind cycle synchronization

1) make Port J outputs
2) write Port J with the 8-bit **data** to convert from digital into analog
3) set the 5-bit **channel** number on PH4-PH0
4) make a 1ms pulse on **DACtrigger** (set to 1, wait 1ms, clear to 0)
5) reset Port J back to inputs

To input a number from the ADC, your software should implement a full handshake gadget

1) set the 5-bit **channel** number on PH4-PH0
2) set **ADCstart** to 1 (starts ADC conversion)
3) wait for **ADCDone** to be 1
4) read the **data** from Port J, analog to digital conversion
5) set **ADCstart** to 0
6) wait for **ADCDone** to be 0

Your device driver has the following prototypes

```c
// Initialize analog interface
void Analog_Init(void);

// Output data to DAC channel
void Analog_Output(unsigned char channel, unsigned char data);

// Input data from ADC channel
unsigned char Analog_Input(unsigned char channel);
```

Show the implementations for these three functions. Use TCNT to make the pulse exactly 1ms.