(10) **Question 1.** Parameters and locals are saved on the stack. The `const` modifier does not change where it is stored, just prevents the function from modifying the parameter value. Both kinds of `static` variables are stored in permanent RAM.

(5) **Question 2.** min = smallest integer*resolution = -128/16
max = largest integer*resolution = 127/16

(5) **Question 3.** value=integer*resolution=12345*100

(4) Part a) The ritual must set the rate, arm, and enable.
(4) Part b) The hardware timer periodically sets the RTIF flag.
(4) Part c) First push, then disable and lastly do the vector fetch
(4) Part d) Every ISR must acknowledge, clearing the flag that triggered the interrupt. The flag is cleared by writing a one to it.
(4) Part e) The `rti` switches the context back to the foreground and reenables interrupts.

(6) **Question 5.** The integer = value/resolution = 100*1024 = 102400, bigger than 65535

(3) **Question 6.** Volatile means information is lost when power is removed.

(3) **Question 7.** $I_{OL}$ is the output current when the signal is low.

(3) **Question 8.** A non-intrusive debugger allows the system to operate normally as if the debugger did not exist.

(3) **Question 9.** Open collector logic has outputs can be low or off.

(3) **Question 10.** Stabilizing means to fix all the inputs to specific values so the experimental procedure can be repeated over and over.

(3) **Question 11.** $V_{IH}$ is the input voltage when the signal is high.

(3) **Question 12.** Busy waiting is an alternative name for gadfly.

(3) **Question 13.** A real time system has bounded latency.

(30) **Question 14.** An LED and two switches are attached Port T.
(5) Part a) R1 and R2 make the voltage high when the switch is off.
(5) Part b) $R = (5-V_C-V_{OL})/I_d = (5-3-1)/0.01$
(5) Part c) C code that makes PT5 an output.
(5) Part d) C code that makes PT2 and PT3 inputs.
(10) Part e) If a switch is not pressed the input will be high. If a switch is pressed the input will be low. The `&0x0C` is performed to mask off unwanted bits.

```c
while(1){
    if((PTT&0x0C)==0x0C){
        PTT &= ~0x20; // LED=off
    } else{
        PTT |= 0x20;  // LED=on
    }
}
```

```c
while(1){
    PTT = (PTT&~0x20) | (((~PTT)&0x08)<<2) | (((~PTT)&0x04)<<3);
} 
```