(5) Question 1. LR contains 0xFFFFFFF9, signifying it is executing an ISR.

(9) Question 2. You are asked to debug a FIFO queue module.
Part a) Stabilization means to fix the inputs, so it can be called over and over having the same conditions.
Part b) Profile means to measure where and when the computer is executing. It is a measure of time.
Part c) Print statements are highly intrusive, affecting the real-time behavior. Also, many embedded systems do not have a standard I/O device.

(10) Question 3. There is a read modify write to the PutPt, so it is not reentrant. You can fix it by disabling interrupts during execution of the function.

(5) Question 4. 8-bit signed integers range from -128 to +127, dividing by 8 yields a range of -16 to +15.875. This is a binary fixed point system with resolution 1/8 = 0.125 cm.

(6) Question 5. `static short data;`

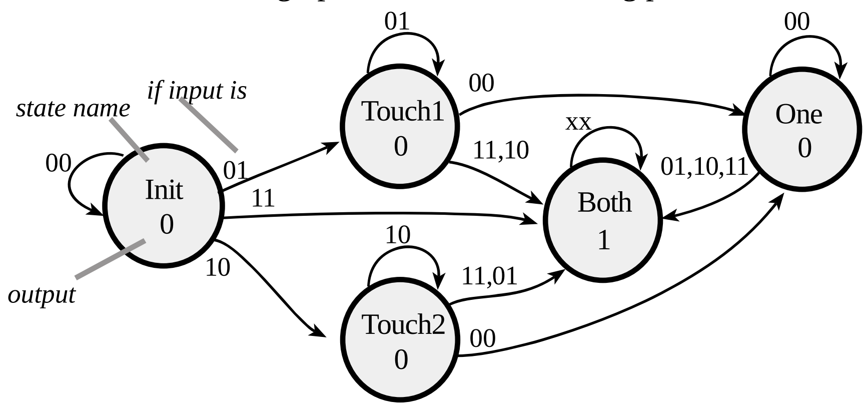
(5) Question 6. I = C dV/dt

(20) Question 7. You are given two tasks: the subroutine `Task1()` should be executed every 1 ms. The subroutine `Task2()` should be executed every 2 ms. Minimize the jitter on executing `Task2()`. This means `Task2()` is never delayed by the running of `Task1`. Assume the PPL is active, making the bus clock 20 ns (50 MHz). You may assume there are no other interrupts.

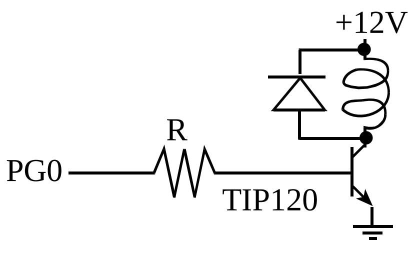
Part a) Show the ritual to initialize this system. You will use SysTick interrupts.
```
volatile unsigned long EveryOther;
void SysTick_Init(void){
  EveryOther = 0;
  NVIC_ST_CTRL_R = 0;         // disable SysTick during setup
  NVIC_ST_RELOAD_R = 49999;   // interrupt every 1ms
  NVIC_SYS_PRI3_R = NVIC_SYS_PRI3_R&0x00FFFFFF; //priority 0
  NVIC_ST_CTRL_R = 0x00000007;// enable with core clock and interrupts
  EnableInterrupts();
}
```

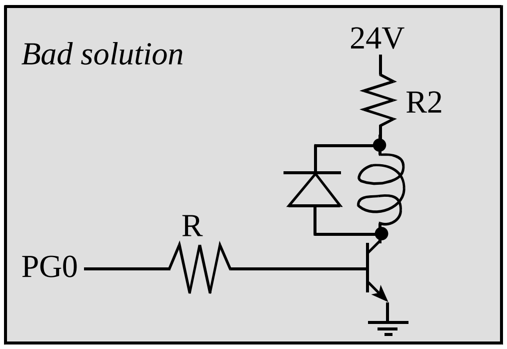
Part b) Show the SysTick interrupt service routine. No backward jumps are allowed.
```
void SysTick_Handler(void){
  EveryOther = EveryOther ^ 1;
  if(EveryOther) Task2();
  Task1();
}
```

(20) Question 8. Draw a Moore FSM graph to solve the following problem.



(20) Question 9. Interface a 12V DC motor to the microcontroller. 1A will require the TIP120 (because it can handle up to 5A of I_{CE}). We could have used any NPN with I_{CE} > 1A. The V_{CE} on voltage of the TIP120 is 0.8V. Because the current gain is 2000 (h_{fe}) the base current needs to be 1A/2000 = 0.5mA. The I_{OH} of the microcontroller can supply this 0.5mA (I_{OH} can be up to 8 mA). Because the V_{OH} of the microcontroller is 2.4V (or greater) and the V_{BE} if the TIP120 is 1.5V (or less), the resistor from the microcontroller to the TIP120 base must be less than (2.4-1.5V)/0.5mA = 0.9V/0.5mA = 1.8 kΩ. I suggest making R much less than 1.8 kΩ (e.g., 1 kΩ) because it will force the NPN into saturation, independent of the V_{OH} of the microcontroller, the V_{BE} of the TIP120, the h_{fe} of the TIP120, and the resistance of the coil. Therefore, when the digital output is high, the voltage across the motor will be 11.2V. You might have been tempted to use a higher voltage supply, like the “Bad solution” and use a series resistor (R2) to drop the voltage down to 12V. There are two fundamental problems with the “Bad solution”. First, the solution wastes power, the power delivered into R2 is lost as heat. Second, the resistance of the coil is a function of the mechanical load on the electromagnet. The coil resistance can not be assumed to be constant.


+12V
R
PG0
TIP120


Bad solution
24V
R2
R
PG0