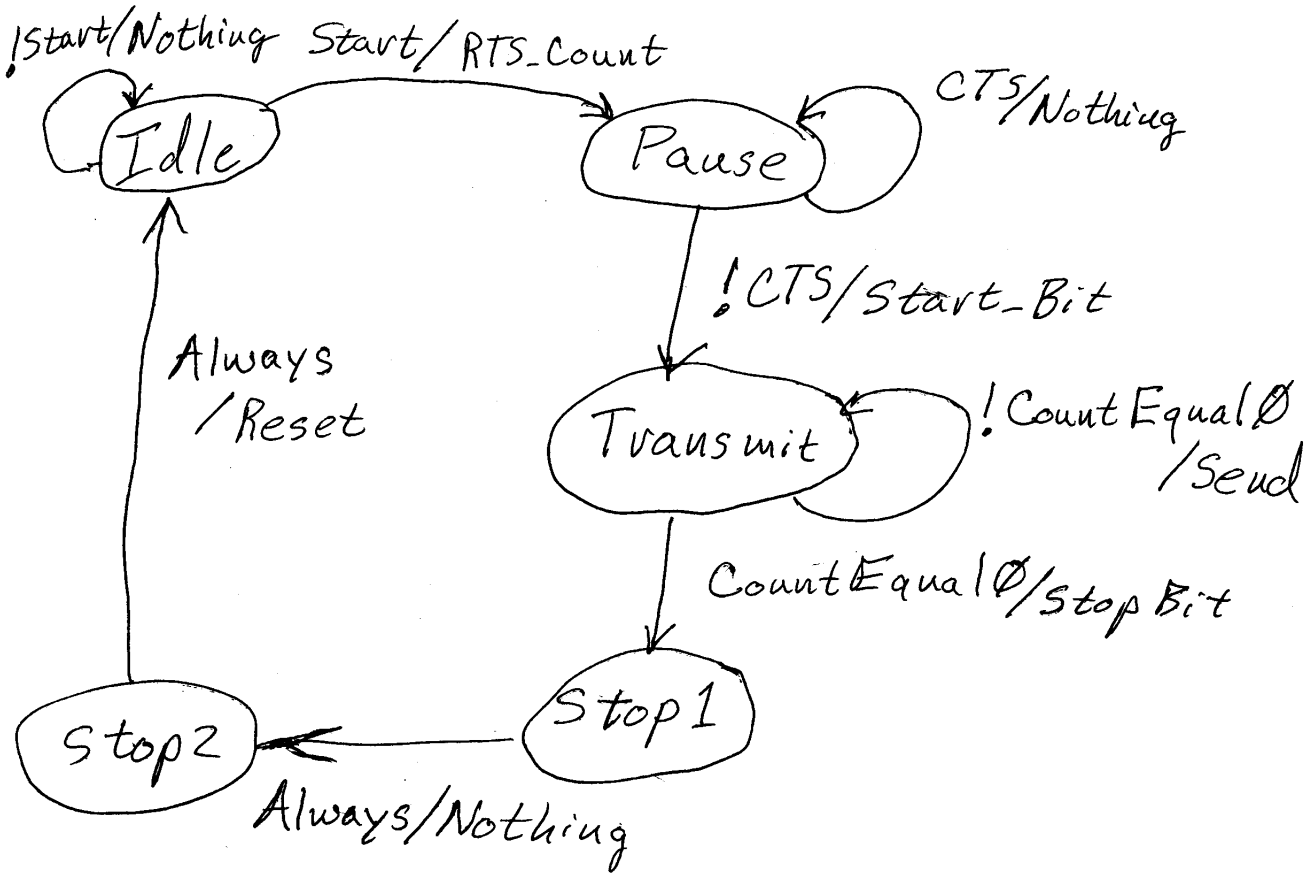
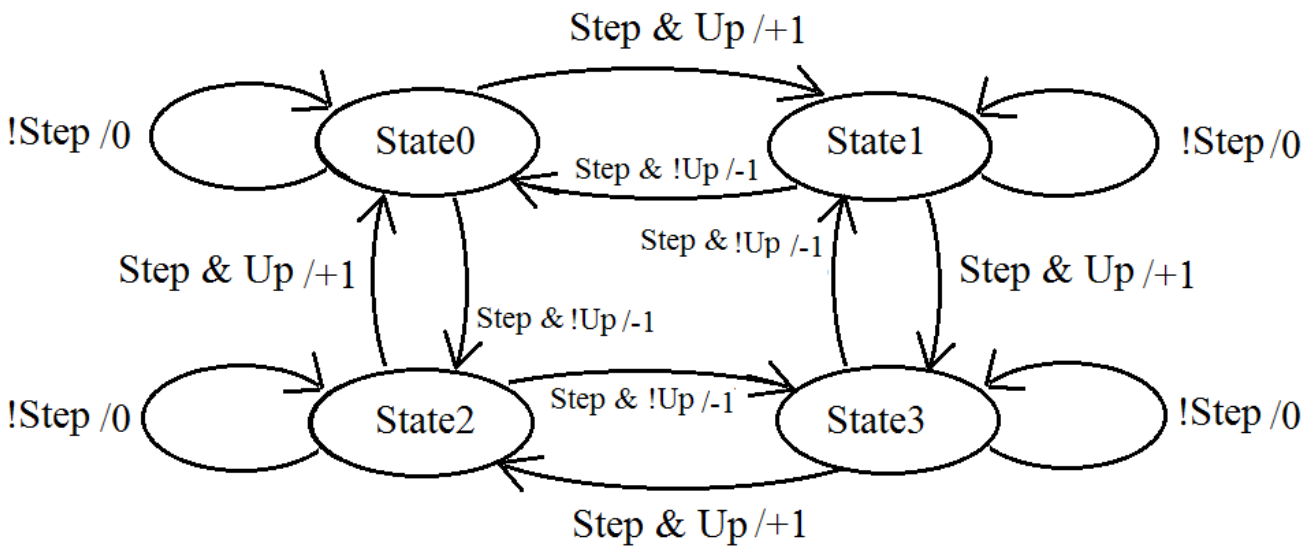
1) Draw the State Transition Diagram for the program in Appendix A.



2) Write a system that will implement the following STD. The system should consist of an "enum" defining the states, and a function to be called every 100 milliseconds (the prototype of which is shown below). The STD shows the transitions and the inputs that cause them and what value should be returned in each case.

```
int SystemNextState( int Step, int Up );
```



Appendix A: Code for Part 1.

```
// Variable of actions and states
// for Serial Transmitter.
enum Actions {
       Nothing, RTS_Count,
       StartBit, Send, StopBit,
       Reset
};
enum States  { Idle, Pause, Transmit, Stop1, Stop2 };
int State = Idle;

// Transmit variables
int Count = 0;
char OutputByte = 0;

// Function that manages the states.
int NextState(int Start, int CTS, int CountEqual0)
{
       int ReturnValue = Nothing; // Default action is nothing.
       switch (State)
       {
       case Idle:     // Waiting for data
              if (Start) // Data Ready
              {
                     State = Pause;  // Set RTS and pause.
                     ReturnValue = RTS_Count;
              }
              break;
       case Pause: // Waiting on Clear To Send
              if (!CTS) // Clear To Send low
              {
                     State = Transmit; // Go to transmit
                     ReturnValue = StartBit; // Send Start bit.
              }
              break;
       case Transmit:  // Transmitting
              if (CountEqual0) // Done transmitting
              {
                     State = Stop1;  // Go to First Stop Bit.
                     ReturnValue = StopBit;
              }
              else
                     ReturnValue = Send; // Send next bit.
              break;
       case Stop1:  // First Stop Bit.
              State = Stop2;
              break;
       case Stop2:  // Second Stop bit.
              State = Idle;
              ReturnValue = Reset; // Clear RTS.
              break;

       } // end of state switch

       return ReturnValue;

} // End of NextState
```

Appendix B: Code Generated For Problem 1.

```
// State Definition for system.
enum SystemStates { State0, State1, State2, State3 };

SystemStates SysState = State0;

// State Transition function.
int SystemNextState(int Step, int Up)
{
	int ReturnValue = 0; // Default is 0.

	// Check state
	switch (SysState)
	{
	case State0:   // State 0,
		if (Step)  // A step is called for.
		{
			if (Up)   // if called to move up
			{
				ReturnValue = 1;  // send +1 to move UP
				SysState = State1;
			}
			else
			{
				ReturnValue = -1; // send -1 to move DOWN
				SysState = State2;
			}
		}
		break;
	case State1:
		if (Step)  // A step is called for.
		{
			if (Up)   // if called to move up
			{
				ReturnValue = 1;  // send +1 to move UP
				SysState = State3;  // Yes it goes to 3, not 2.
			}
			else
			{
				ReturnValue = -1; // send -1 to move DOWN
				SysState = State0;
			}
		}
		break;
	case State3:
		if (Step)  // A step is called for.
		{
			if (Up)   // if called to move up
			{
				ReturnValue = 1;  // send +1 to move UP
				SysState = State2;
			}
			else
			{
				ReturnValue = -1; // send -1 to move DOWN
				SysState = State1;
			}
		}
		break;
```

```
	case State2:
		if (Step)  // A step is called for.
		{
			if (Up)   // if called to move up
			{
				ReturnValue = 1;  // send +1 to move UP
				SysState = State0;
			}
			else
			{
				ReturnValue = -1; // send -1 to move DOWN
				SysState = State3;
			}
		}
		break;

	}// end of switch

	return ReturnValue;

} // end of SystemNextState

// The following is the code to keep test system.
// Timer to phase out state transitions.
unsigned long Timer;
int TimeCount = 10;

// put your setup code here, to run once:
void setup()
{
	Serial.begin(9600);
	Timer = millis();
} // End of setup.

int Direction, Up = 0, Step = 0;
void loop() {
	// put your main code here, to run repeatedly:

	if (millis() - Timer > 100)
	{
		Direction = SystemNextState(Step, Up);

		Serial.print(Step);
		Serial.print(", ");
		Serial.print(Up);
		Serial.print(" => ");
		Serial.print(Direction);
		Serial.print(", ");
		Serial.println(SysState);

		Step = !Step;
		if (TimeCount)
			TimeCount--;
		else
		{
			Up = !Up;
			TimeCount = 10;
		}
		Timer += 100;
	}
} // end of loop.
```

Serial Output from Test Program, with added comments in green

```
0, 0 => 0, 0  // State0
1, 0 => -1, 2 // State2
0, 0 => 0, 2  // State2
1, 0 => -1, 3 // State3
0, 0 => 0, 3  // State3
1, 0 => -1, 1 // State1
0, 0 => 0, 1  // State1
1, 0 => -1, 0
0, 0 => 0, 0
1, 0 => -1, 2
0, 0 => 0, 2
1, 1 => 1, 0
0, 1 => 0, 0
1, 1 => 1, 1
0, 1 => 0, 1
1, 1 => 1, 3
0, 1 => 0, 3
1, 1 => 1, 2
0, 1 => 0, 2
1, 1 => 1, 0
0, 1 => 0, 0
1, 1 => 1, 1
```