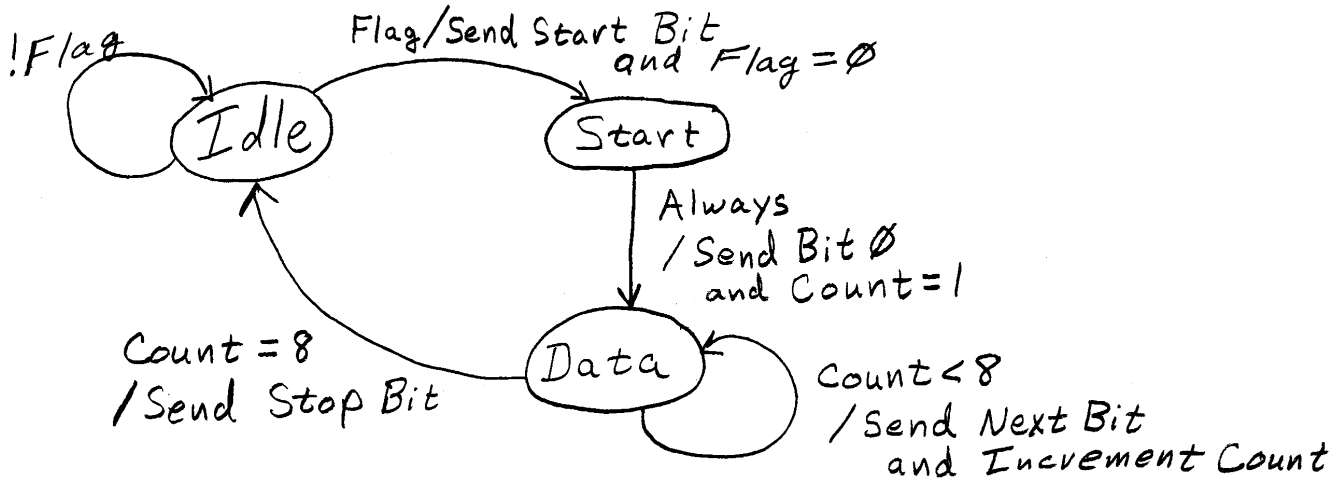
Software Serial Transmit Code

```
// Header file defining software serial system
#ifndef SW_Serial_H
#define SW_Serial_H 1

#include <TimerOne.h>
// Define buffer for outgoing data.
char SW_Serial_Buffer;
int SW_Serial_Flag = 0;
int SW_Serial_Pin; // Output Pin for Serial port.

// Set up transmitter state machine
enum SW_Serial_States { SW_Serial_Idle, // State waiting for character
                        SW_Serial_Start,    // Start Bit Tranmit
                        SW_Serial_Data};    // Send data

SW_Serial_States SW_Serial_State = SW_Serial_Idle;
int SW_Serial_BitCount = 0;
char SW_Serial_Hold = 0; // Holds data being transmitted.

// Interrupt Service Routine that sends characters serially
// over a pin, using a state machine.
void SW_Serial_ISR(void)
{
       // Based on state,
       switch (SW_Serial_State)
       {
       default:
       case SW_Serial_Idle:  // idling waiting for data in buffer
              if (SW_Serial_Flag)
              {
                     // Data in buffer, so Pull out data, and save for later.
                     SW_Serial_Hold = SW_Serial_Buffer;
                     SW_Serial_Hold &= 0x7f; // Mask off top bit
                     // Could add parity bit?
                     // Advance OUT pointer
                     SW_Serial_Flag = 0; // Reset Flag

                     digitalWrite(SW_Serial_Pin, LOW); // Clear output (start bit).
                     SW_Serial_State = SW_Serial_Start;  // Move to start state.
              } // End of character in if
              break;
       case SW_Serial_Start:
              // Send out LSB.
              digitalWrite(SW_Serial_Pin, bitRead(SW_Serial_Hold, 0)); // Set output.
              SW_Serial_BitCount = 1; // Next bit to send.
              SW_Serial_State = SW_Serial_Data; // Move to data sending state.
              break;
       case SW_Serial_Data:
              if (SW_Serial_BitCount < 8) // if not past all bits.
              {
                     // Send next bit and increment bit count.
                     digitalWrite(SW_Serial_Pin,
                            bitRead(SW_Serial_Hold, SW_Serial_BitCount++));
              }
              else  // past all bits, Send Stop bit.
              {
                     digitalWrite(SW_Serial_Pin, HIGH); // Set for stop bit.
                     SW_Serial_State = SW_Serial_Idle;  // Next pass, look for another
character.
              } // End of bit count if
              break;
       } // End of state switch
}// End of SW_Serial_ISR
```

```
void SW_Serial_Initialize(int BaudRate, int pin)
{
	int BitTime_ms = 1000000 / BaudRate; // Compute bit time in microseconds.
	SW_Serial_Pin = pin;  // Save pin number for later use.
	pinMode(SW_Serial_Pin, OUTPUT);     // Set pin to output
	digitalWrite(SW_Serial_Pin, HIGH); // and high or RS232-idle state.

	// Set up timer to run ISR at bit time.
	Timer1.initialize(BitTime_ms);
	Timer1.attachInterrupt(SW_Serial_ISR, BitTime_ms);

	// Initialize buffer.
	SW_Serial_Flag = 0;
}

// This function will place a character into the circular buffer.
void SW_Serial_Transmit(char ch)
{
	// Insert ch at current in pointer.
	SW_Serial_Buffer = ch;
	// Advance in pointer
	SW_Serial_Flag = 1;
}
#endif
```

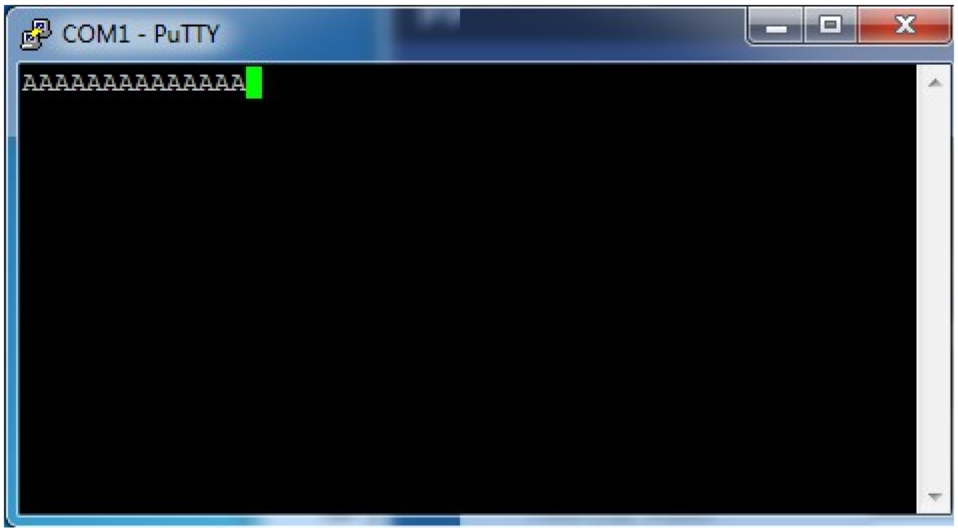
State Transition Diagram for Transmitter.


!Flag
Flag/Send Start Bit and Flag=Ø
Idle
Start
Always /Send Bit Ø and Count=1
Data
Count=8 /Send Stop Bit
Count<8 /Send Next Bit and Increment Count

Arduino Code to test SW_Serial

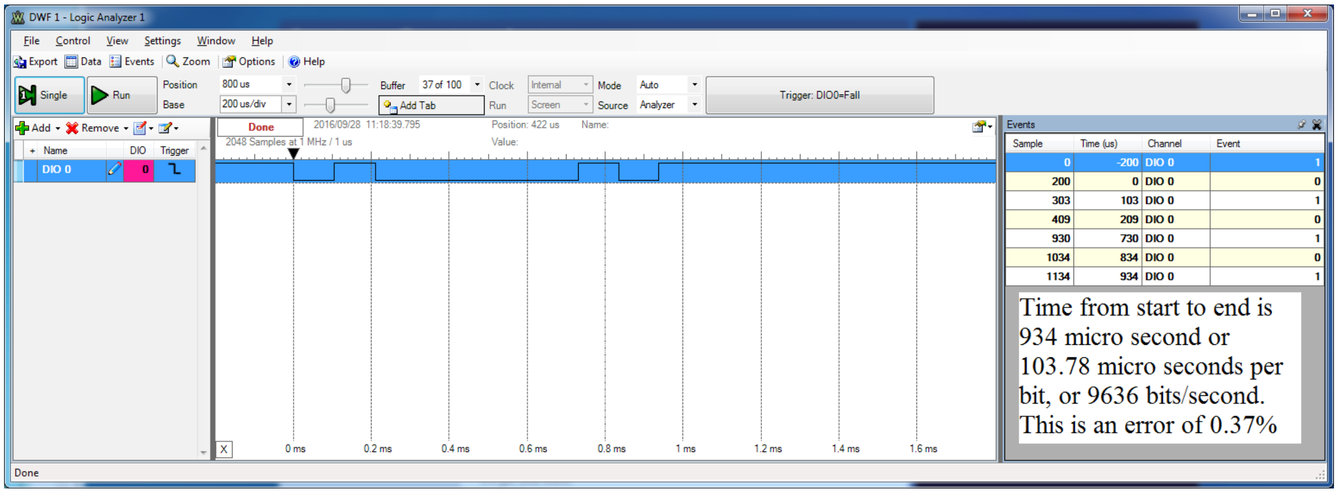
```
#include "SW_Serial.h" // Header file defining software serial system

unsigned long SendTimer = 0; // one second timer.

// setup code, ran once:
void setup()
{
	pinMode(13, OUTPUT);
	pinMode(12, OUTPUT);
	SW_Serial_Initialize(9600, 12); // Set Serial to 9600 baud
	// and pin 12.

	// Timer  used to run TimerService every second.
	SendTimer = millis();
}

// main code, ran repeatedly:
void loop()
{
	// Check for one second
	if (millis() - SendTimer >= 1000)
	{
		// send character to transmitter.
		SW_Serial_Transmit('A');
		SendTimer += 1000;
	}
}// End of loop.
```

Results on Terminal.



```
AAAAAAAAAAAAAA
```

Logic analyzer reading of serial output.



Position: 800 us
Base: 200 us/div
Buffer: 37 of 100
Clock: Internal
Run: Screen
Mode: Auto
Source: Analyzer
Trigger: DIO0=Fall

Done 2016/09/28 11:18:39.795 Position: 422 us Name:
2048 Samples at 1 MHz / 1 us Value:

Name	DIO	Trigger
DIO 0	0	⌐

Events

Sample	Time (us)	Channel	Event
0	-200	DIO 0	1
200	0	DIO 0	0
303	103	DIO 0	1
409	209	DIO 0	0
930	730	DIO 0	1
1034	834	DIO 0	0
1134	934	DIO 0	1

Time from start to end is 934 micro second or 103.78 micro seconds per bit, or 9636 bits/second. This is an error of 0.37%

Done