Message Parsing on a Microcontroller

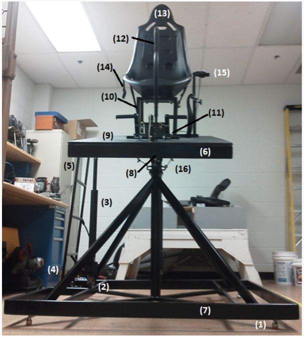
Using Finite State Machines

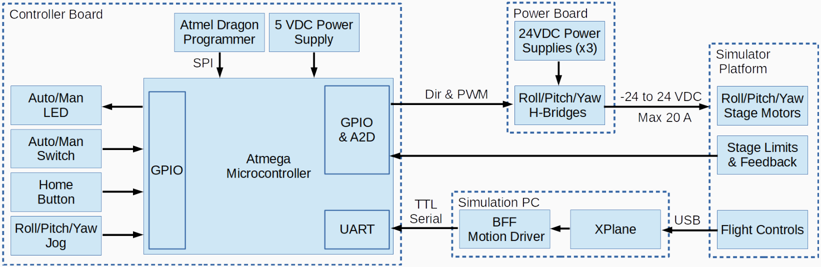
David J. Broderick, Ph.D.

February 10, 2019

Introduction

- Interfacing peripheral devices is a necessary task in robotic control systems
- How do we write code to receive a message?
- Relying on libraries is limiting





Controller Board

Atmel Dragon Programmer

5 VDC Power Supply

SPI

Auto/Man LED

Auto/Man Switch

Home Button

Roll/Pitch/Yaw Jog

GPIO

Atmega Microcontroller

GPIO & A2D

UART

Dir & PWM

Power Board

24VDC Power Supplies (x3)

Roll/Pitch/Yaw H-Bridges

-24 to 24 VDC

Max 20 A

Simulator Platform

Roll/Pitch/Yaw Stage Motors

Stage Limits & Feedback

Flight Controls

TTL Serial

Simulation PC

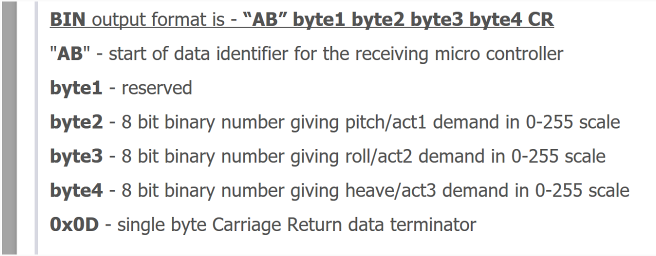
BFF Motion Driver

XPlane

USB

Problem Statement

- Objective: parse the message passed from the simulation PC to the Controller



BIN output format is - **“AB” byte1 byte2 byte3 byte4 CR**

"**AB**" - start of data identifier for the receiving micro controller

byte1 - reserved

byte2 - 8 bit binary number giving pitch/act1 demand in 0-255 scale

byte3 - 8 bit binary number giving roll/act2 demand in 0-255 scale

byte4 - 8 bit binary number giving heave/act3 demand in 0-255 scale

0x0D - single byte Carriage Return data terminator

Overview of Procedure

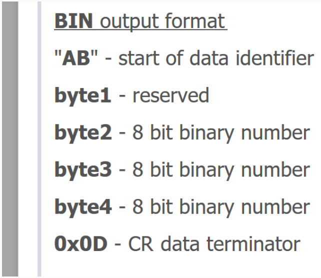
Procedure:

1. When will we service the state machine?
2. Define the states
3. Define the transitions
4. Taking action
5. Implementation

Assumptions:

1. UART Initialized (BAUD rate, Format etc.)
2. Interrupt Service Routine setup

Define States



BIN output format

"**AB**" - start of data identifier

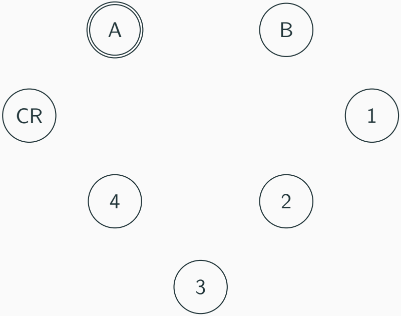
byte1 - reserved

byte2 - 8 bit binary number

byte3 - 8 bit binary number

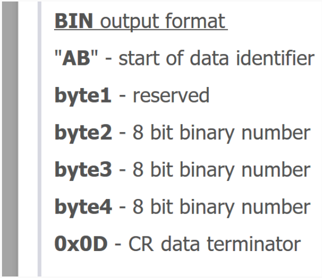
byte4 - 8 bit binary number

0x0D - CR data terminator



- "A" → 0x41
- "B" → 0x42

Defining Transitions



BIN output format

"**AB**" - start of data identifier

byte1 - reserved

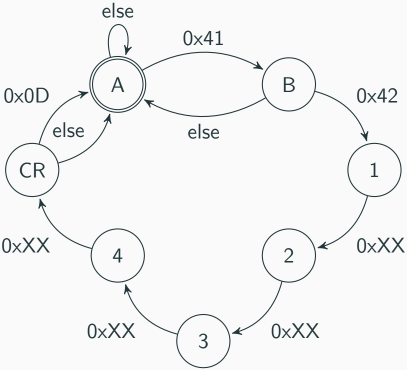
byte2 - 8 bit binary number

byte3 - 8 bit binary number

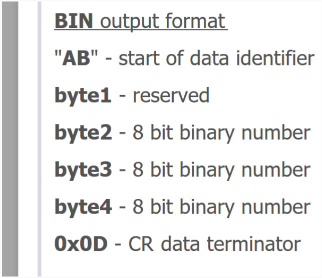
byte4 - 8 bit binary number

0x0D - CR data terminator

- “A” → 0x41
- “B” → 0x42



Taking Action on Transitions



BIN output format

"**AB**" - start of data identifier

byte1 - reserved

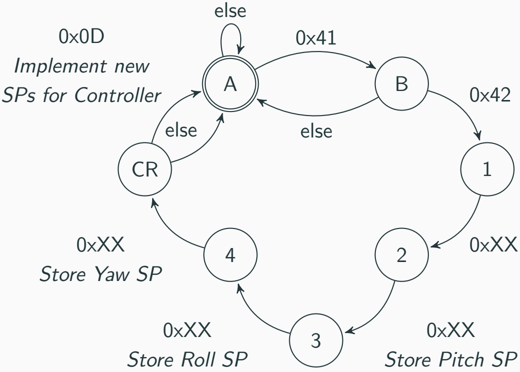
byte2 - 8 bit binary number

byte3 - 8 bit binary number

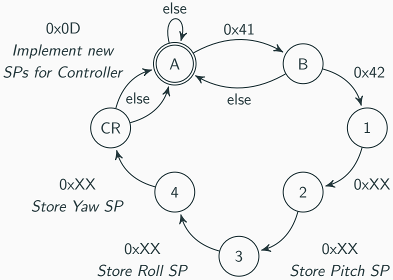
byte4 - 8 bit binary number

0x0D - CR data terminator

- “A” → 0x41
- “B” → 0x42



Implementation

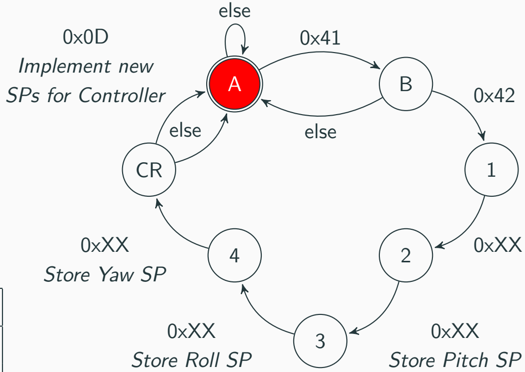


```
procedure UART RX INTERRUPT
    switch State do
        case A
            if 0x41 then Set State B
        case B
            if 0x42 then Set State 1
            else Set State A
        case 1
            Set State 2
        case 2
            Store Temp. Pitch SP
            Set State 3
        case 3
            Store Temp. Roll SP
            Set State 4
        case 4
            Store Temp. Yaw SP
            Set State CR
        case CR
            if 0x0D then Impl. SPs
            Set State A
```

An Example

Received Data

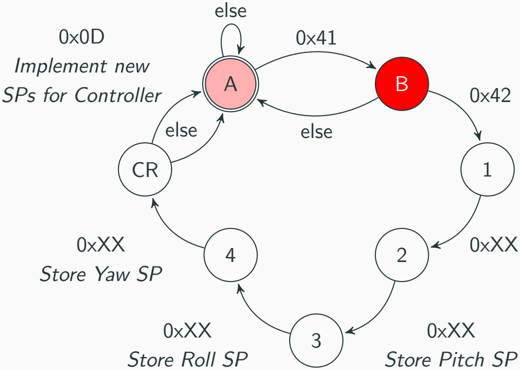
	Temp	SP
Pitch		
Roll		
Yaw		



An Example

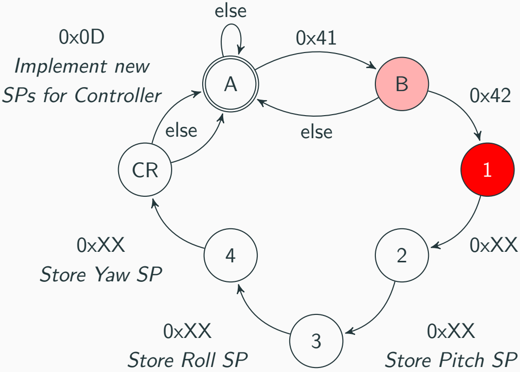
Received Data
0x41

	Temp	SP
Pitch		
Roll		
Yaw		



An Example

Received Data
0x41
0x42

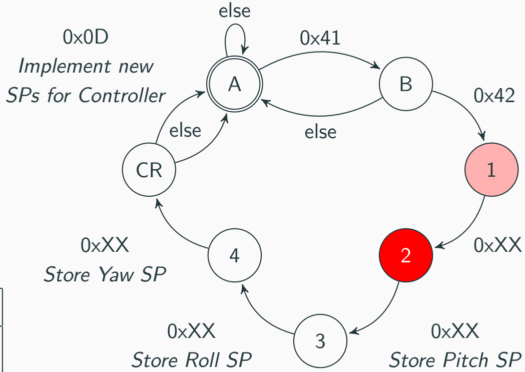


	Temp	SP
Pitch		
Roll		
Yaw		

An Example

Received Data
0x41
0x42
0x00

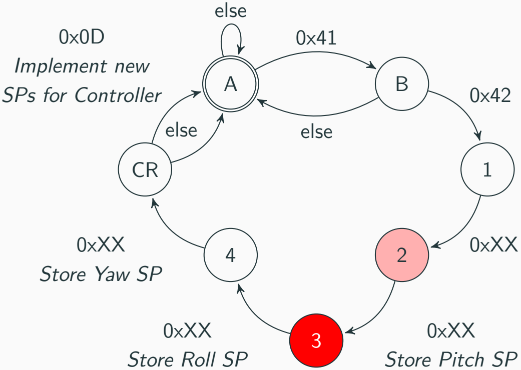
	Temp	SP
Pitch		
Roll		
Yaw		



An Example

Received Data
0x41
0x42
0x00
0x7F

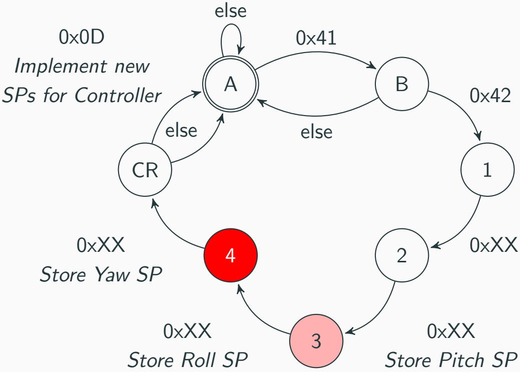
	Temp	SP
Pitch	0x7F	
Roll		
Yaw		



An Example

Received Data
0x41
0x42
0x00
0x7F
0xC8

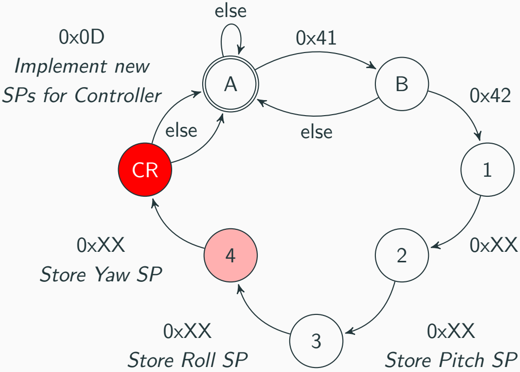
	Temp	SP
Pitch	0x7F	
Roll	0xC8	
Yaw		



An Example

Received Data
0x41
0x42
0x00
0x7F
0xC8
0x4A

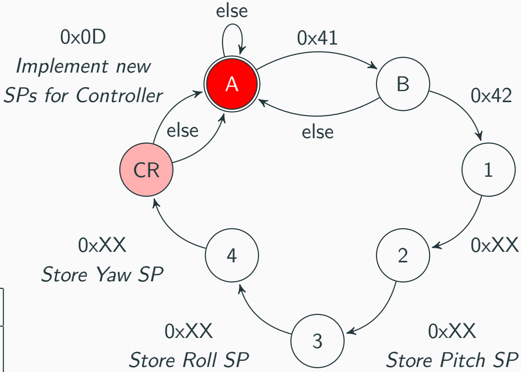
	Temp	SP
Pitch	0x7F	
Roll	0xC8	
Yaw	0x4A	



An Example

Received Data
0x41
0x42
0x00
0x7F
0xC8
0x4A
0x0D

	Temp	SP
Pitch	0x7F	0x7F
Roll	0xC8	0xC8
Yaw	0x4A	0x4A



Handling Data Loss

Things Will Go Wrong

Sources of data loss include:

- Noise
- Framing Errors
- BAUD rate mismatch

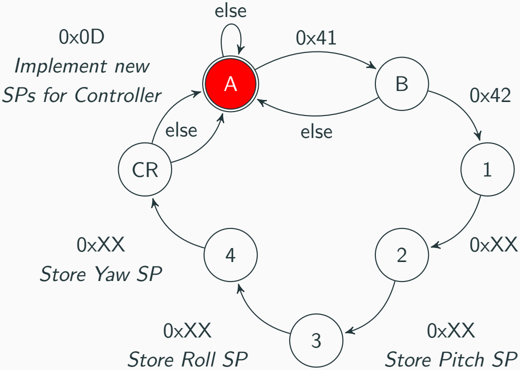
You must:

- Identify how the receiver will act
- Consider the system level effect

Handling Data Loss

Received Data

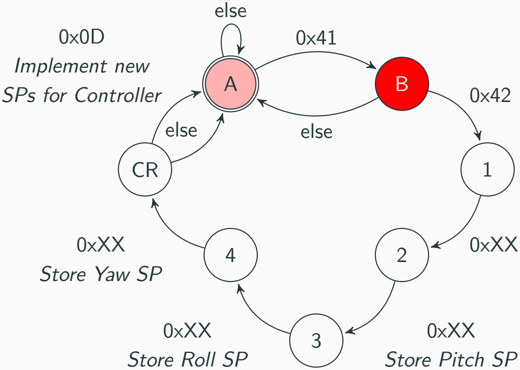
	Temp	SP
Pitch		
Roll		
Yaw		



Handling Data Loss

Received Data
0x41

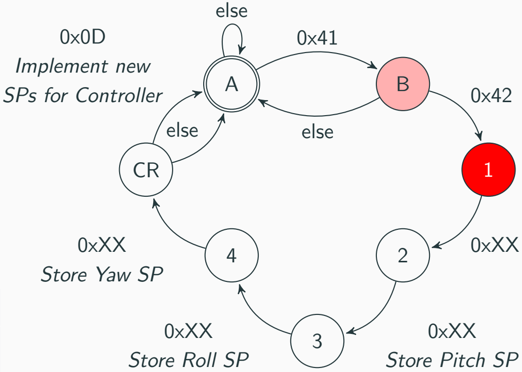
	Temp	SP
Pitch		
Roll		
Yaw		



Handling Data Loss

Received Data
0x41
0x42

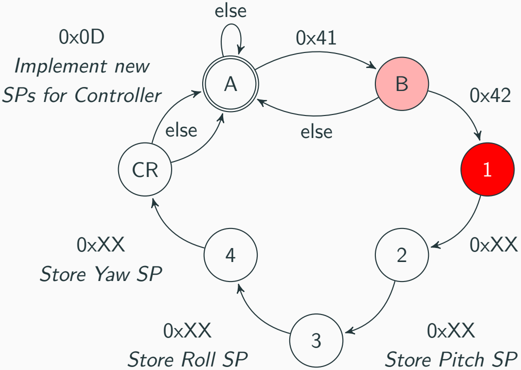
	Temp	SP
Pitch		
Roll		
Yaw		



Handling Data Loss

Received Data
0x41
0x42
~~0x00~~

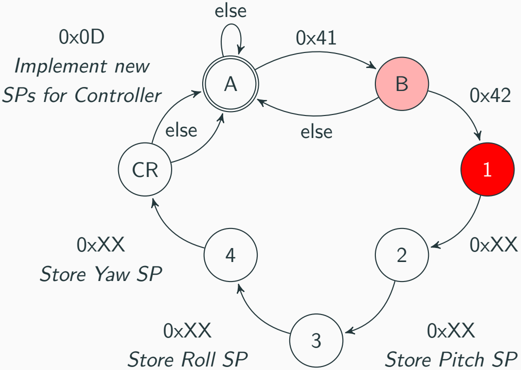
	Temp	SP
Pitch		
Roll		
Yaw		



Handling Data Loss

Received Data
0x41
0x42
~~0x00~~
~~0x7F~~

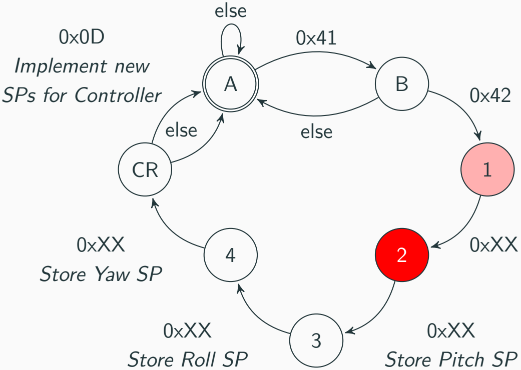
	Temp	SP
Pitch		
Roll		
Yaw		



Handling Data Loss

Received Data
0x41
0x42
~~0x00~~
~~0x7F~~
0xC8

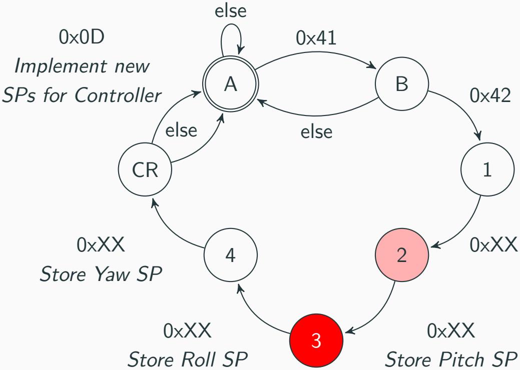
	Temp	SP
Pitch		
Roll		
Yaw		



Handling Data Loss

Received Data
0x41
0x42
~~0x00~~
~~0x7F~~
0xC8
0x4A

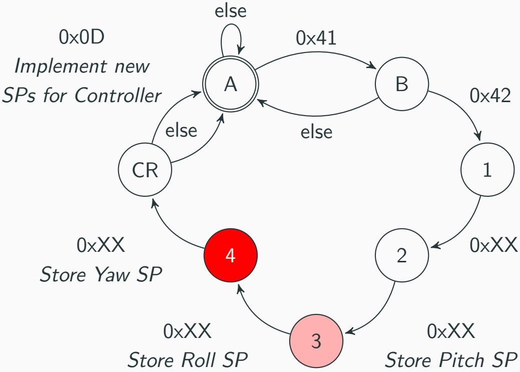
	Temp	SP
Pitch	0x4A	
Roll		
Yaw		



Handling Data Loss

Received Data
0x41
0x42
~~0x00~~
~~0x7F~~
0xC8
0x4A
0x0D

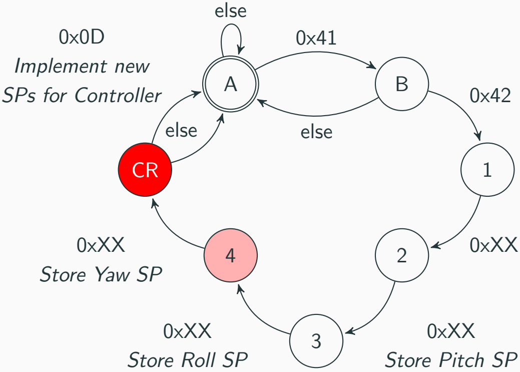
	Temp	SP
Pitch	0x4A	
Roll	0x0D	
Yaw		



Handling Data Loss

Received Data	
0x41	0x41
0x42	
~~0x00~~	
~~0x7F~~	
0xC8	
0x4A	
0x0D	

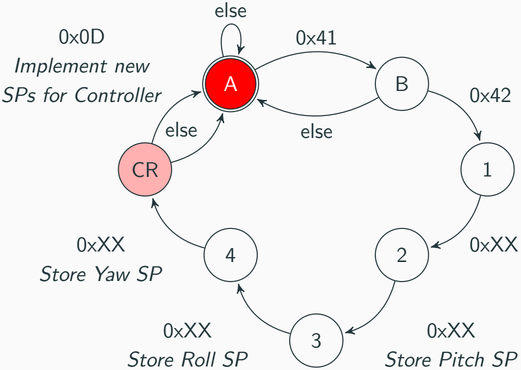
	Temp	SP
Pitch	0x4A	
Roll	0x0D	
Yaw	0x41	



Handling Data Loss

Received Data	
0x41	0x41
0x42	0x42
~~0x00~~	
~~0x7F~~	
0xC8	
0x4A	
0x0D	

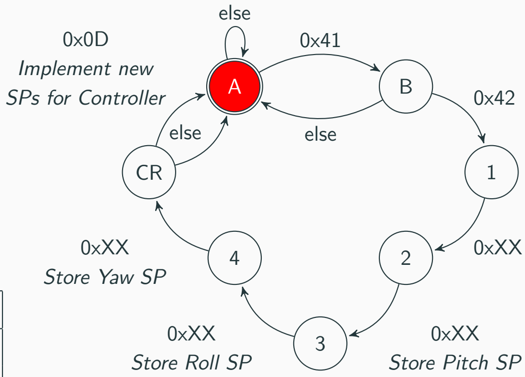
	Temp	SP
Pitch		
Roll		
Yaw		



Handling Data Loss

Received Data	
0x41	0x41
0x42	0x42
~~0x00~~	0x00
~~0x7F~~	0x82
0xC8	0xC4
0x4A	0x4A
0x0D	0x0D

	Temp	SP
Pitch		
Roll		
Yaw		

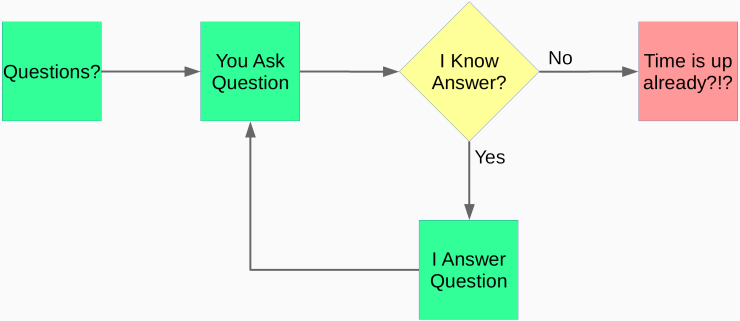


Limitations

To be considered:

- This is not the only approach
- Buffering
- Checksum
- Variable data length

Questions?


Questions?
You Ask Question
I Know Answer?
No
Time is up already?!?
Yes
I Answer Question