



CTI-P301

Advanced Programming Concepts

Getting to Know You

First Name on both sides

On one side:

- What company do you work for?
- What is your favorite place to visit?



Guidelines

- Classroom
 - **Respect others**, suspend judgment
 - **Share and learn** through discussions
 - **Ask questions** to get what you need from the course
 - **“Show up”** & participate
 - **Be punctual** from breaks
 - Set **Cell phones** to a lowered ring tone

Logistics

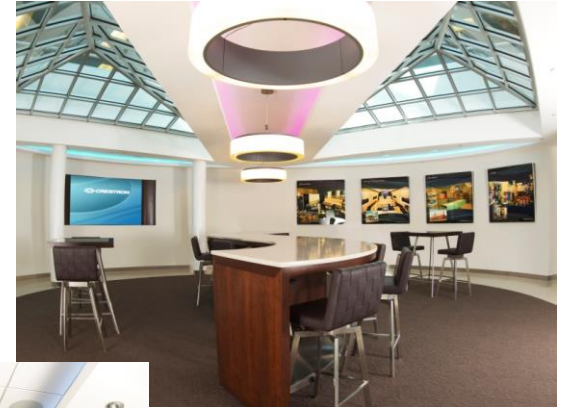
- Breaks/ Lunch
- Restrooms
- Phones/ messages
- Smoking
- Building escorts
- Return name badge holders at the end of the day
- Dinner (3 drink limit)



CTI-P301 Advanced Programming Concepts

Recommended PC Configuration:

- i7 processor
- 8.0 GB RAM
- 1280x800 screen resolution
- Windows 7, 8.1 or 10
- Administrative privileges
- Adobe® Flash® v10 or later
- Ethernet Port
- USB port
- External Mouse



Handouts

- FTP site for Classroom use only
 - In IE type ftp://10.100.0.2
 - Username: Student
 - Password: Training
 - In IE use Alt-View | Open in File explorer for a faster method of coping files to your local hard disc drive
 - I recommend C:\Crestron\CTI-P301\Class\

Agenda

Day 1

- > Introductions
- > Programming Fundamentals
- > Introduction to SIMPL+
- > Events
- > Variables
- > String manipulation & Arrays

Agenda

Day 2

- Math operations: Application of mathematic formulas
- String data formats and String functions
- Numeric bases and operations
- Byte deconstruction and long number reconstruction
- Loops, Flags
- File operations
- Direct socket connections
- Parameters

Agenda

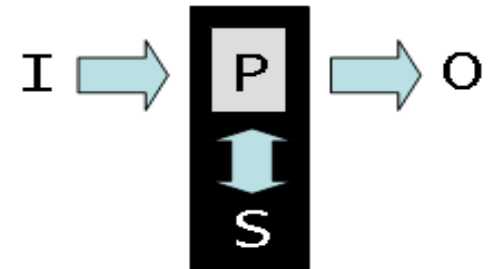
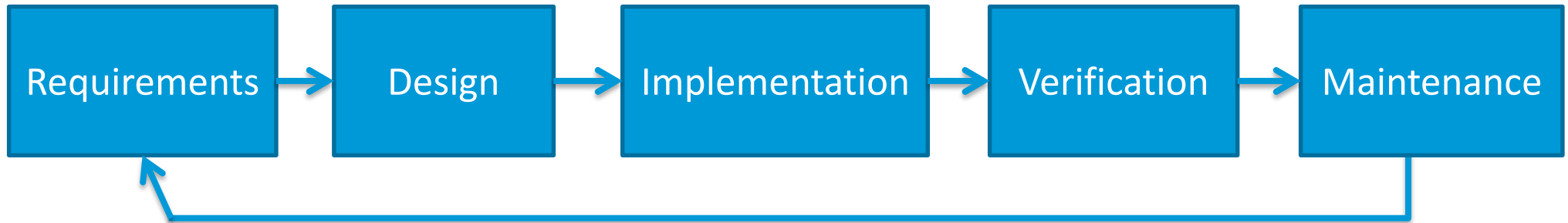
Day 3

- > Comparisons
- > Multicast
- > Advanced Debugging
- > SIMPL# class in SIMPL / SIMPL+

SHTHFSTH

Something has to happen for
Something to happen.

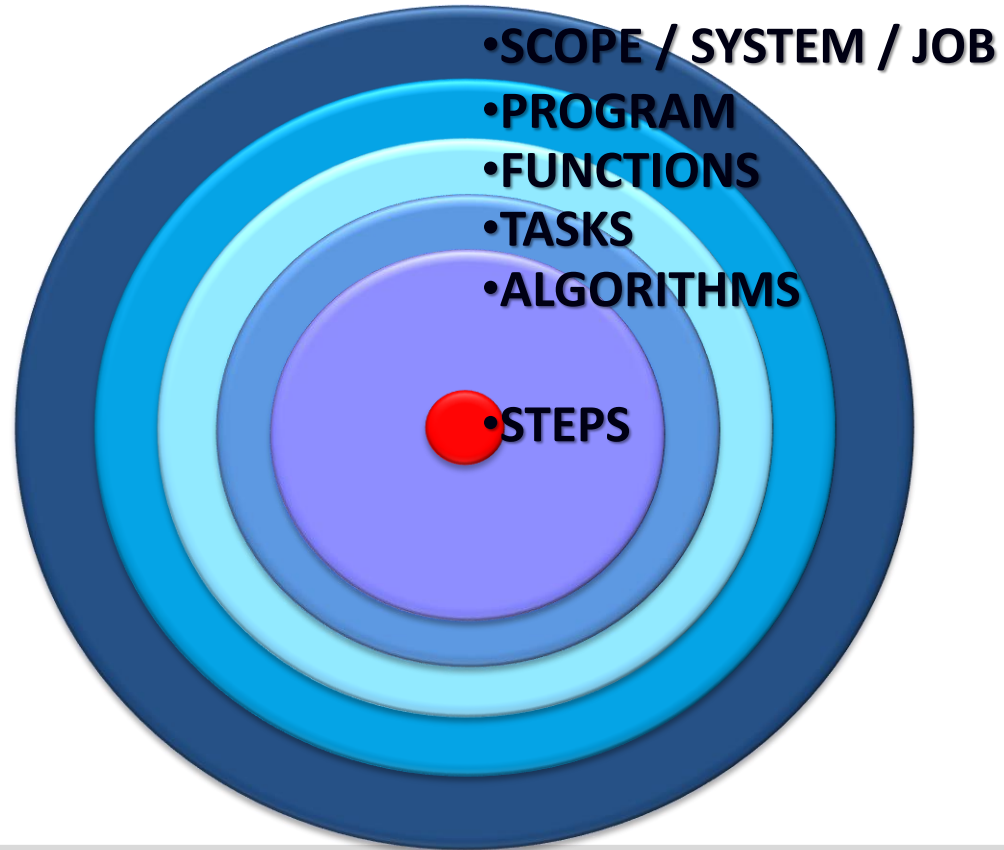
Input – Process – Output Model



Pseudo Logic Challenge



Layers of Programming



There are no User Errors

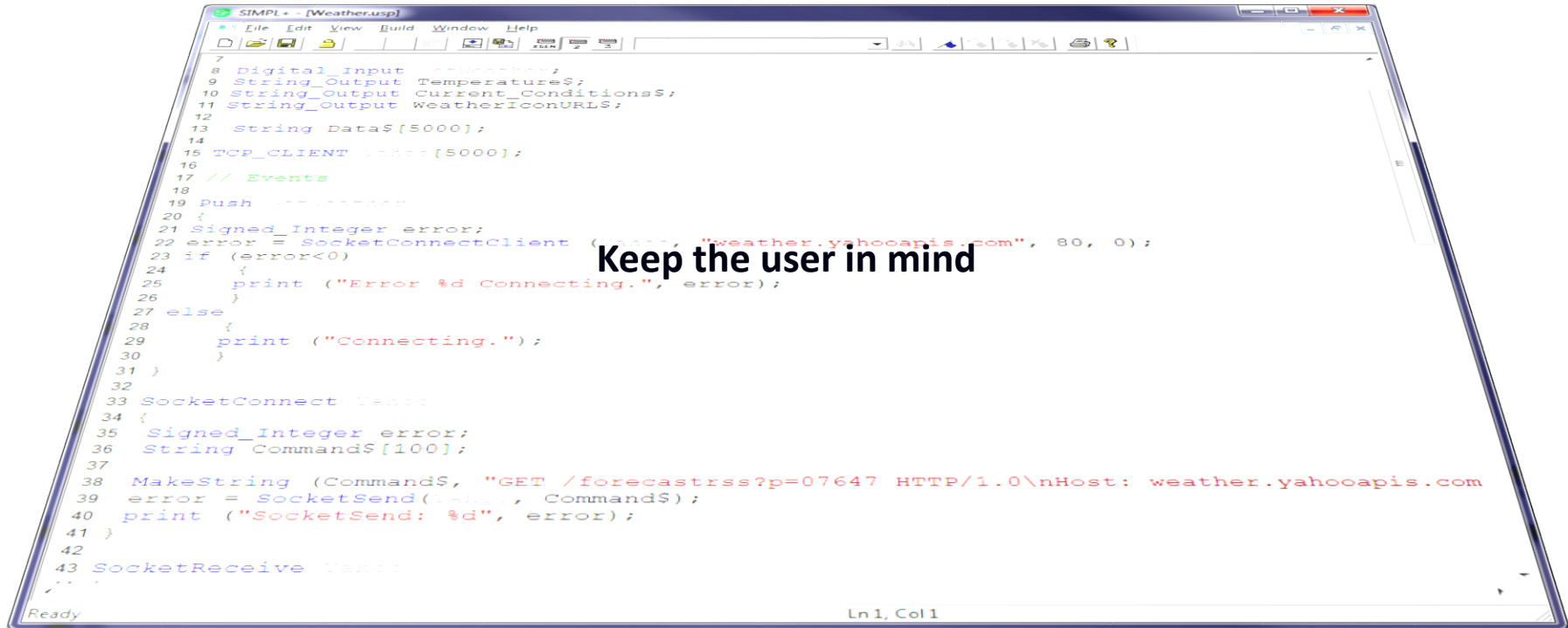




CTI-P301

Advanced Programming Concepts

Introduction to Simpl+



```
7
8 Digital_Input  Temperature;
9 String_Output Temperature$;
10 String_Output Current_Conditions$;
11 String_Output WeatherIconURL$;
12
13 String Data$(5000);
14
15 TCP_CLIENT sock[5000];
16
17 // Events
18
19 Push 0
20 {
21 Signed_Integer error;
22 error = SocketConnectClient (sock, "weather.yahooapis.com", 80, 0);
23 if (error<0)
24 {
25     print ("Error %d Connecting.", error);
26 }
27 else
28 {
29     print ("Connecting.");
30 }
31 }
32
33 SocketConnect Yahoo
34 {
35 Signed_Integer error;
36 String Command$(100);
37
38 MakeString (Command$, "GET /forecastrss?p=07647 HTTP/1.0\nHost: weather.yahooapis.com
39 error = SocketSend(sock, Command$);
40 print ("SocketSend: %d", error);
41 }
42
43 SocketReceive Yahoo
44 {
45     ...
46 }
```

Keep the user in mind

Ready Ln1, Col1



First Rule of SIMPL+

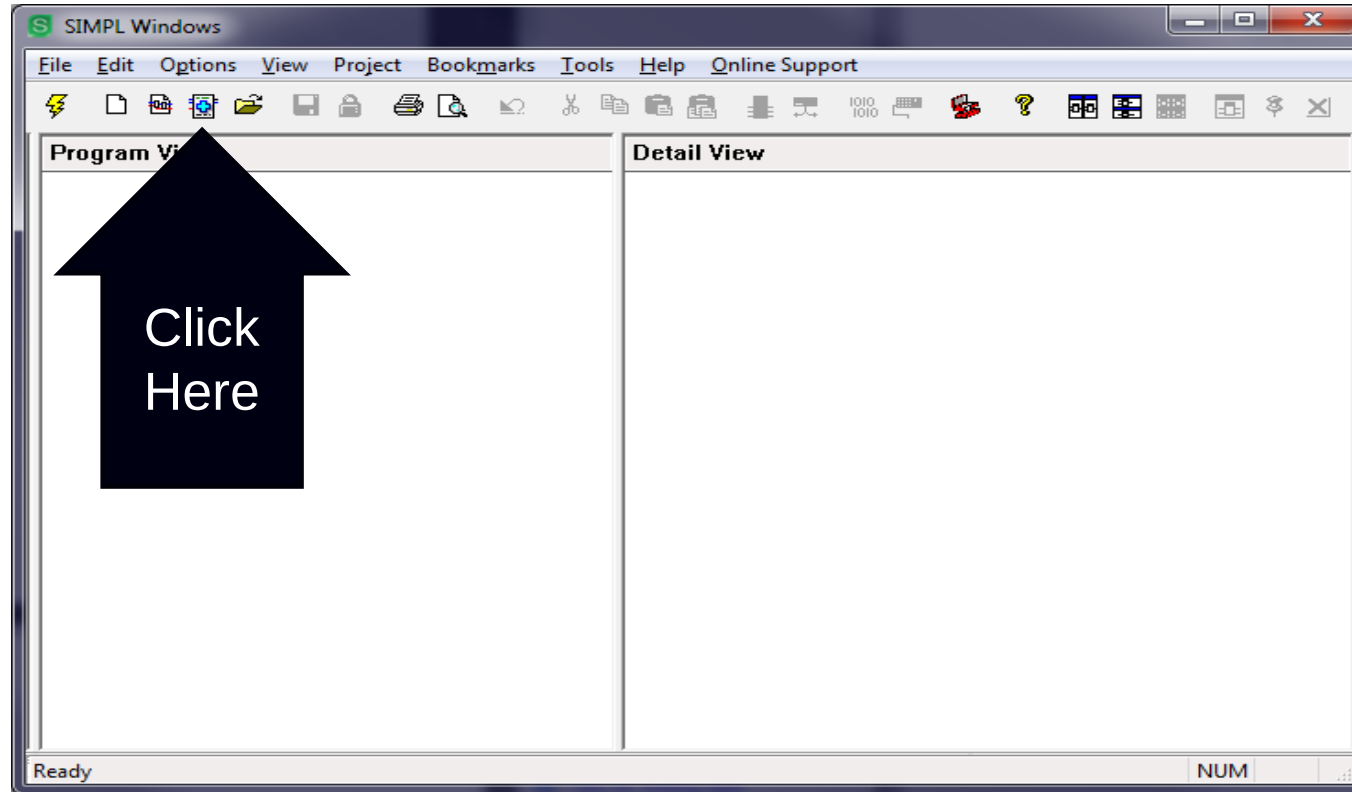
First Rule of Simpl+

> **SHTHFSTH**

> Don't Use SIMPL+

> When you have SIMPL Windows logic that can do the same thing already!

Let's get started!



> **SHTHFSH**

- > Don't Use SIMPL+(when you have SIMPL Windows logic that can do the same thing)
- > Watch for [] () { } ; ,
 - > Syntax.
 - > Key is to look at the line with the error
 - > As well as looking above the line with the error as it may be syntax issue with previous lines of code.

> **SHTHFSTH**

- > Don't Use SIMPL+ (when you have SIMPL Windows logic that can do the same thing)
- > Watch for [] () { } ; ,
- > Programmers are lazy
 - > Efficient!

AV Switcher Control



Manufacturer's Protocol

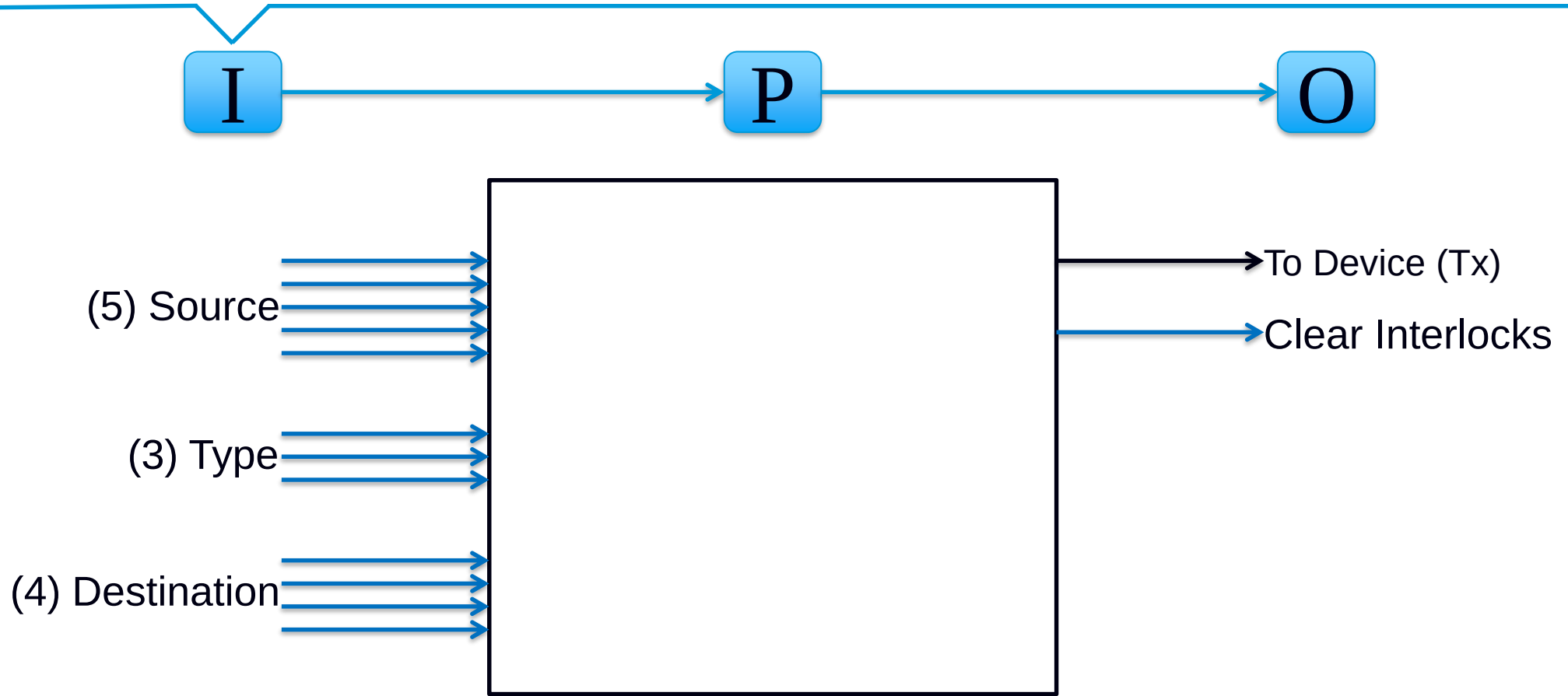
<input>,<output>,<type>[CR]

input = single digit number

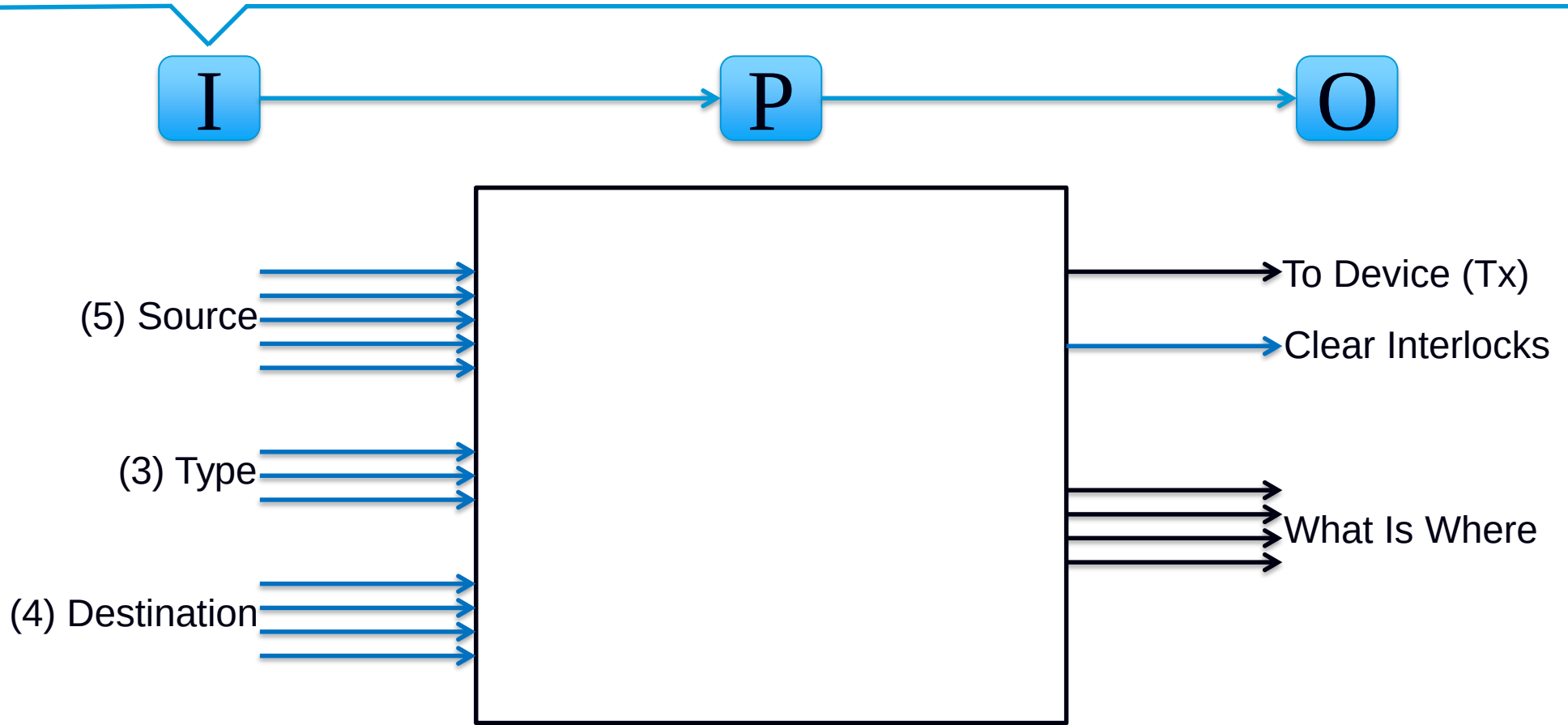
output = single digit number

type = '^' Audio+Video / '@' Audio / '&' Video

AV Switcher Control: Pseudoblock



AV Switcher Control: Pseudoblock



Questions ???



CTI-P301 Day 2 Math Day





Quiz Time

Math Operators

Addition: +

Subtraction: -

Multiplication: *

Division: /

Division: MOD

Exponents: ^

Math Evaluators

Equal To: =

Unequal: <>

Less than: <

Greater than: >

Less than or Equal: <=

Greater than or Equal: >=

16 bits =

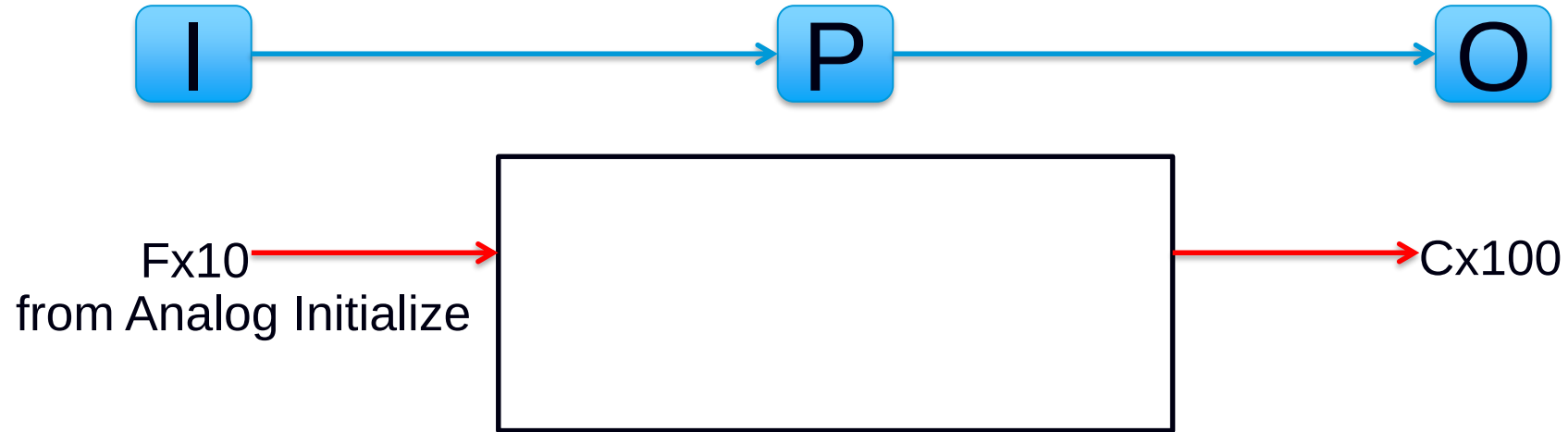
0000000000000000 to 1111111111111111

16 bit integers = **0 to 65535**

16 bit signed integers = **-32768 to +32767**

Decimals: None

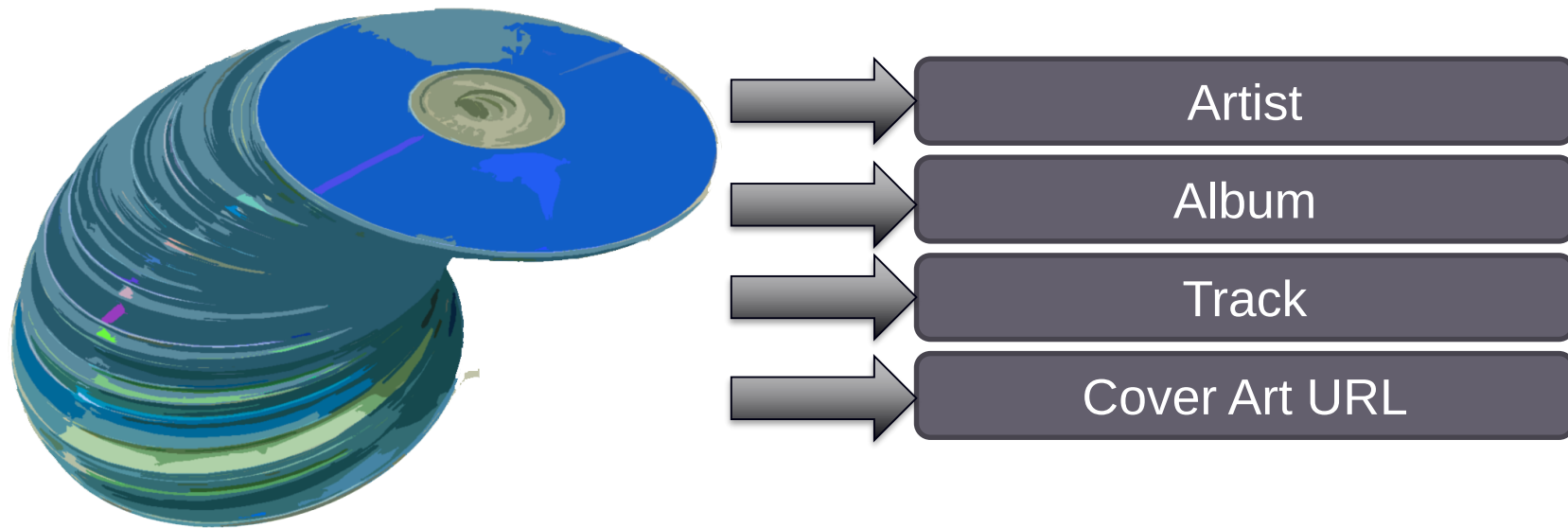
FtoC: Pseudoblock



Math Exercise: Workgroups

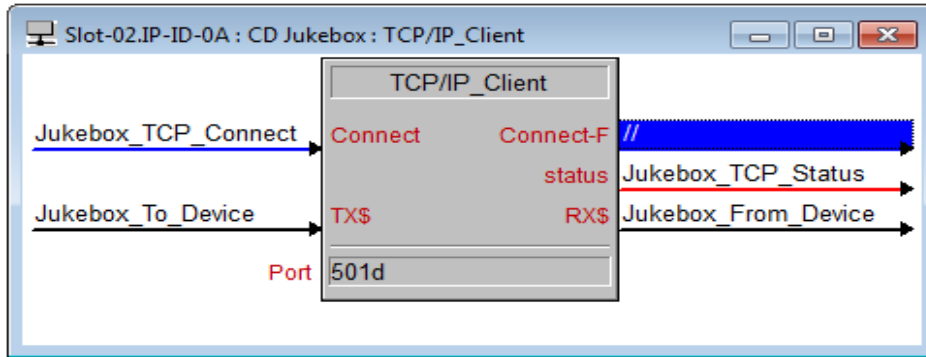
- Converting Fahrenheit to Celsius
 - Input: Degrees Fx10 from program or keyboard.
 - Output: Degrees Cx100

CD Jukebox



Connecting to the Jukebox

Address: TCP 10.0.7.77



Student Number	TCP Port Number
1	501
2	502
3	503
4	504
5	505
6	506
7	507
8	508
9	509
10	510
11	511
12	512

Manufacturer's Protocol

<command>[20]<value>[CR]

command = 'CD' Disc / 'TR' Track

value = '+' advance / '-' previous

Example: CD[20]+[0D]

Data Analysis



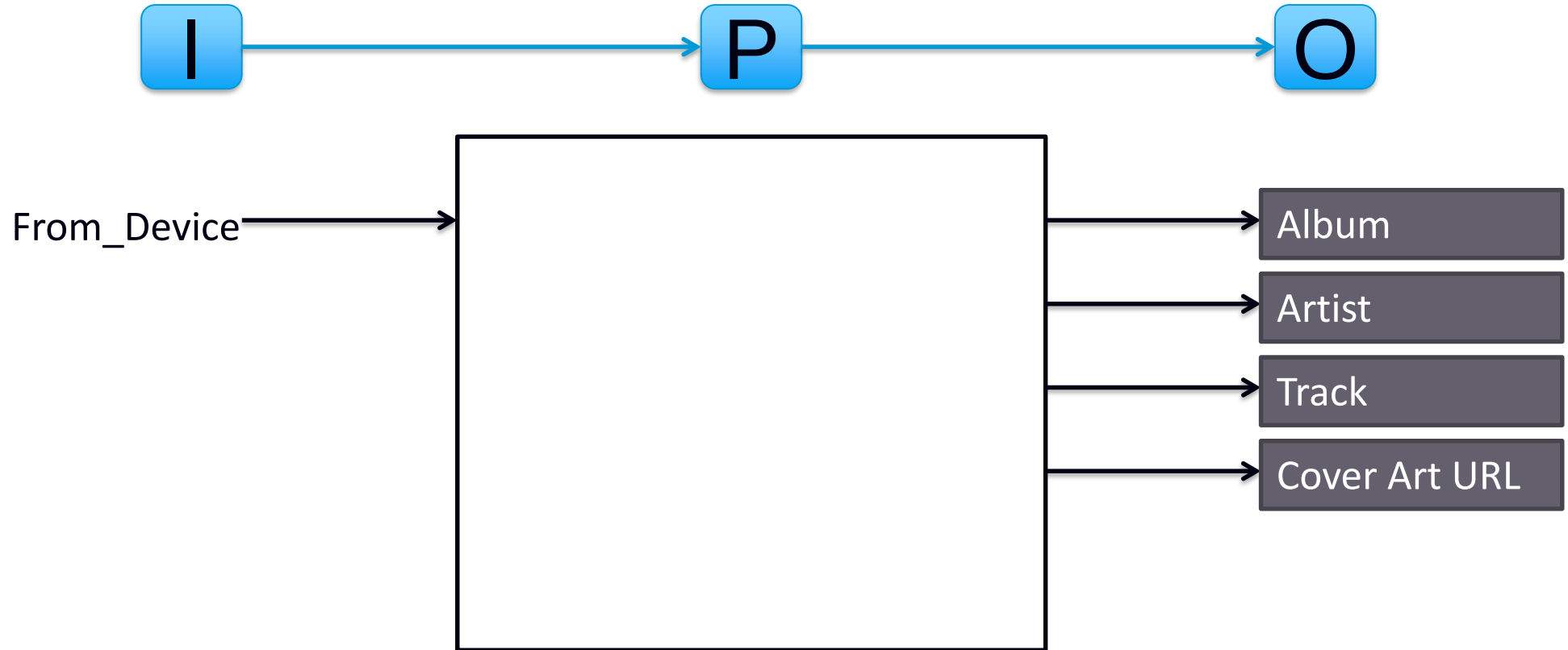
- > Is the **record** a constant length?
- > Are the **fields** constant lengths?
- > Is the **record** consistently delimited?
- > Are the **fields** consistently delimited?

Data Analysis



- Is the **record** a constant length? **No**
- Are the **fields** constant lengths? **No**
- Is the **record** consistently delimited? **Yes**
- Are the **fields** consistently delimited? **Yes**

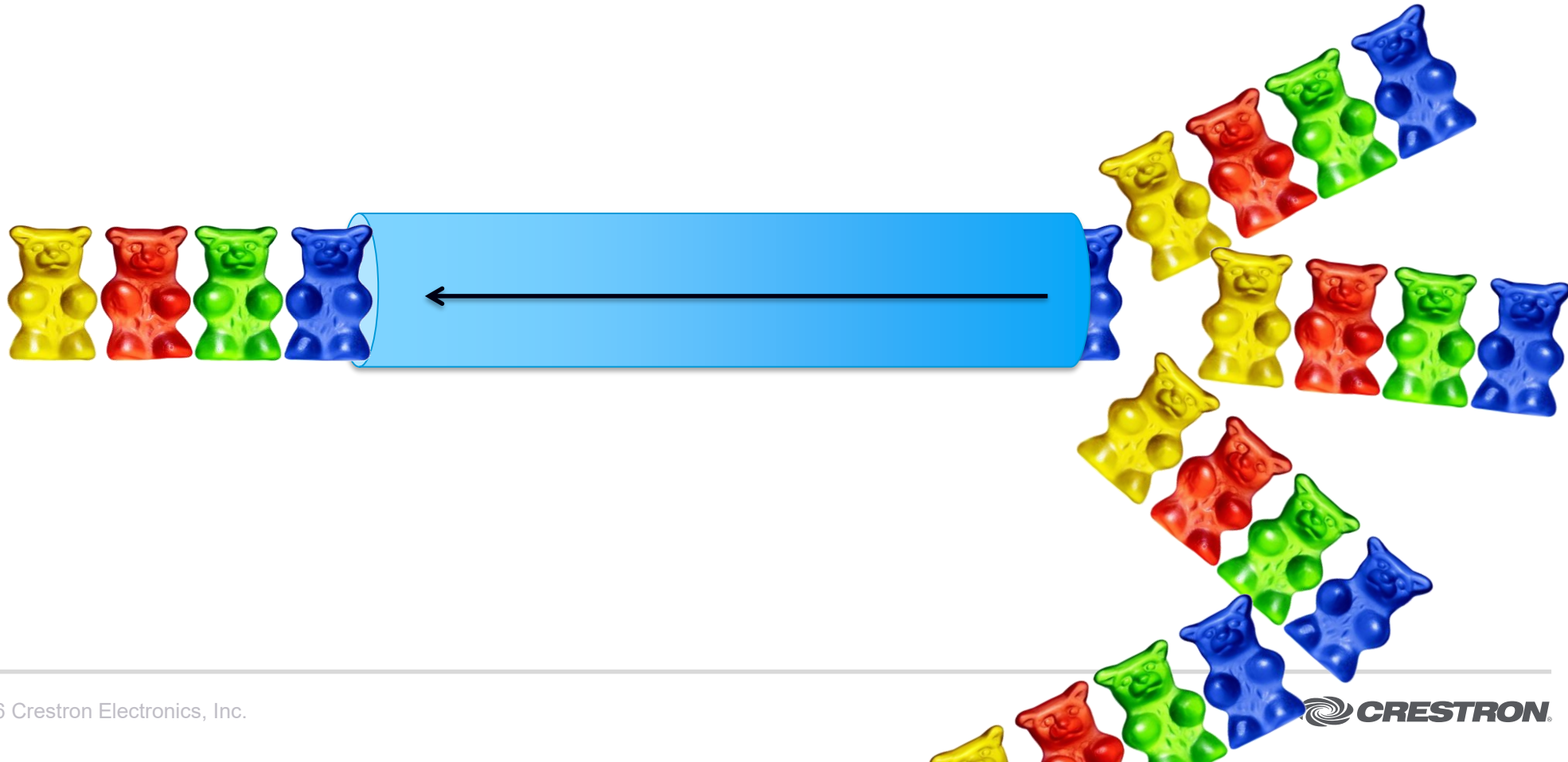
CD Jukebox: Pseudoblock



FIFO Buffer



Buffer Overflow



Data Analysis



+ Sample Data Grid

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Q	u	e	e	n			C	o	l	l	e	c	t	i	o	n			B	i	c	y	c	l	e			1	5	1	.	j	p	g	0D

Data Analysis: Artist



+ Sample Data Grid

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
Q	u	e	e	n			C	o	l	l	e	c	t	i	o	n			B	i	c	y	c	l	e			1	5	1	.	j	p	g	0D

p1



+ Artist is the first 5 characters on the left,
Therefore: Characters = 6 – 1.

Data Analysis: Album



+ Sample Data Grid

1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3
									0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
Q	u	e	e	n			C	o	l	l	e	c	t	i	o	n		B	i	c	y	c	l	e			1	5	1	.	j	p	g	0D	

p1

p2

Calculator

View Edit Help



- > Album is from 8 to 17 (10 characters)
- > Therefore: Left Position = 6+2
- > Characters = 18-6-2

Data Analysis: Track

Record

Artist

||

Album

||

Track

||

Cover Art

\x0D

+ Sample Data Grid

1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	
									0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
Q	u	e	e	n			C	o			e	c	t	i	o	n			B	i	c	y	c		e			1	5	1	.	j	p	g	0D

p1

p2

p3

Calculator

View Edit Help



- > Track is from 20 to 26 (7 characters)
- > Therefore: Left Position = 18 + 2
- > Characters = 27 - 18 - 2

Data Analysis: Cover Art

Record

Artist

||

Album

||

Track

||

Cover Art

\x0D

+ Sample Data Grid

1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3		
									0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
Q	u	e	e	n			C	o			e	c	t	i	o	n			B	i	c	y	c	l	e			1	5	1	.	j	p	g	0D
p1					p2															p3											?				

Deriving the final pointer...

Data Analysis: Cover Art

Record

Artist

||

Album

||

Track

||

Cover Art

\x0D

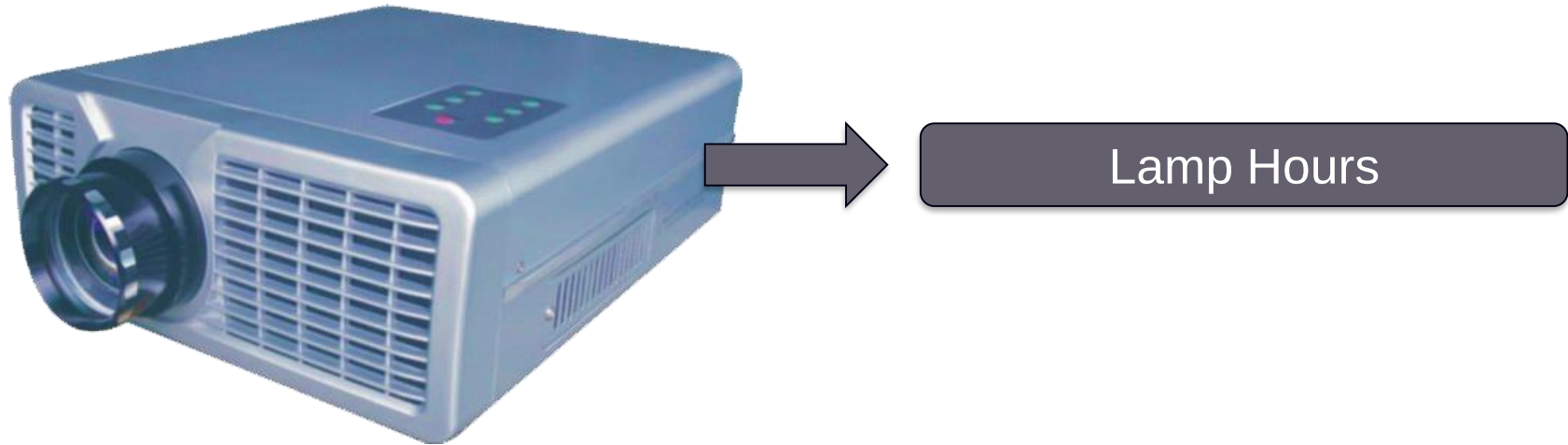
+ Sample Data Grid

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Q	u	e	e	n			C	o	l	l	e	c	t	i	o	n			B	i	c	y	c	l	e			1	5	1	.	j	p	g	0D
p1					p2													p3							len										



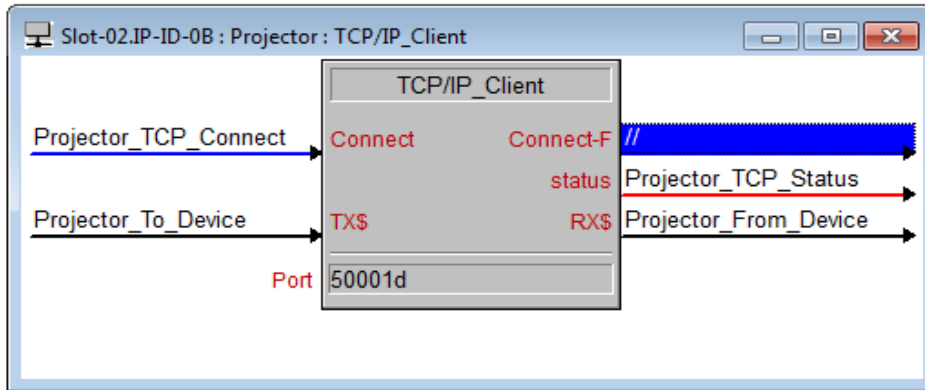
- > Cover Art is from 29 to 35 (7 characters)
- > Therefore: Left Position = 27 + 2
- > Characters = 36 - 27 - 2

Projector Lamp Hours



Connecting to the Projector

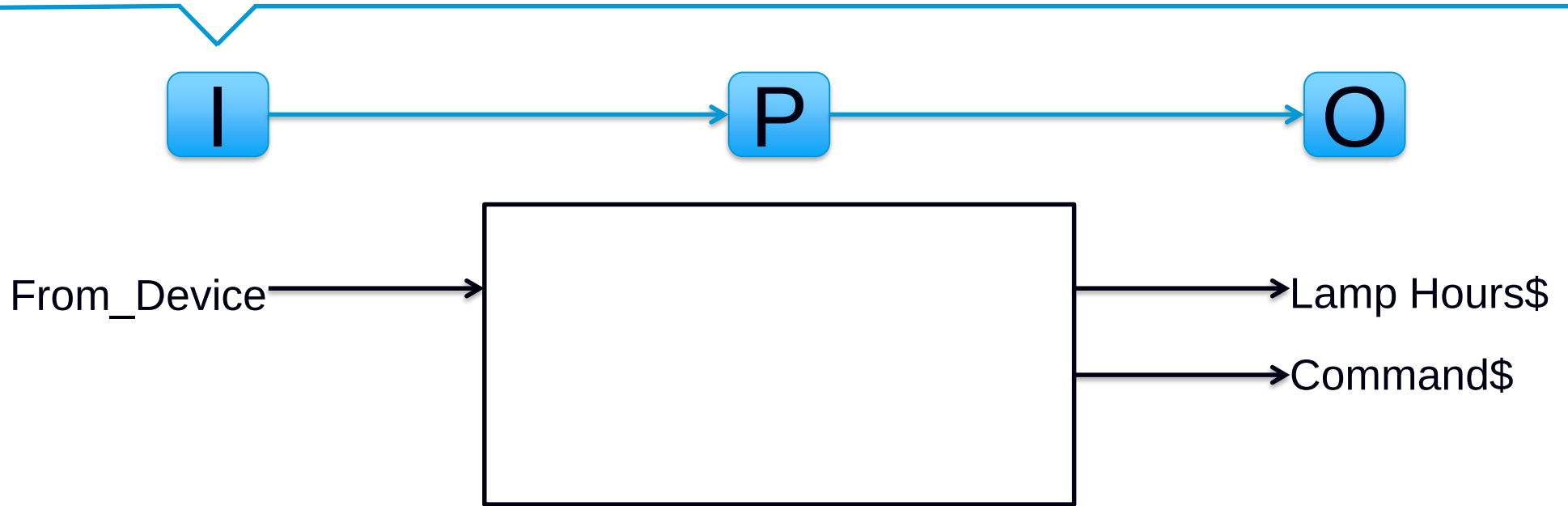
Address: EMULATOR



Manufacturer's Protocol
PollAdvanced
[03][8C][00][00][00][8F]
Reset
[02]RESET[03]

Student Number	TCP Port Number
1	50001
2	50002
3	50003
4	50004
5	50005
6	50006
7	50007
8	50008
9	50009
10	50010
11	50011
12	50012

Projector: Pseudoblock

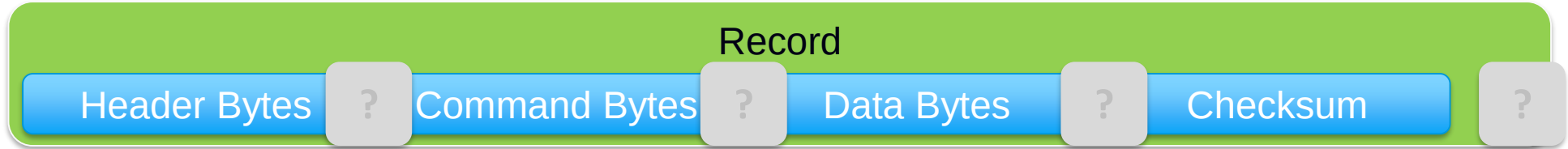


Data Analysis



- Is the **record** a constant length?
- Are the **fields** constant lengths?
- Is the **record** consistently delimited?
- Are the **fields** consistently delimited?

Data Analysis



- Is the **record** a constant length? **No.**
- Are the **fields** constant lengths? **No.**
- Is the **record** consistently delimited? **No.**
- Are the **fields** consistently delimited? **No**

Data Analysis: Returned Values

Record

Header Bytes

?

Command Bytes

?

Data Bytes

?

Checksum

?

Lamp Hours Request (HEX): 03 8C 00 00 00 8F

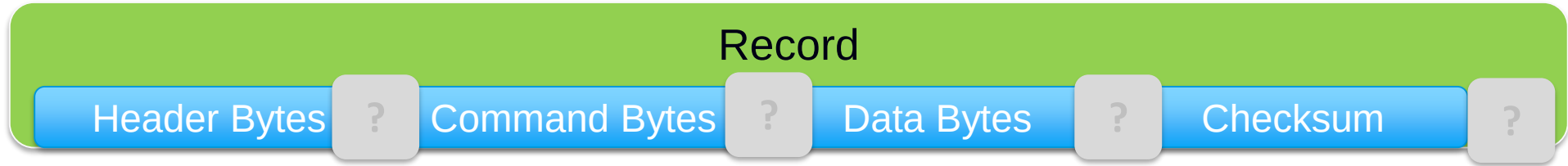
Lamp Hours Response:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
23	8C	01	20	10	38	4D	04	00	00	00	00	00	80	E0	62	00	00	DD	6D	00	75

Error Response:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
20	88	01	20	0C	80	01	00	00	00	00	00	00	00	00	00	00	AE				

Data Analysis: Lamp Hours Return



Lamp Hours Request (HEX): 03 8C 00 00 00 8F

Lamp Hours Response:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
23	8C	01	20	10	38	4D	04	00	00	00	00	00	80	E0	62	00	00	DD	6D	00	75	
					LS		MS															CS
					Lamp Seconds Bytes																	Checksum

Number Reassembly: Base 10

Starting Number: 1,234

Disassembled into four components (reverse ordered):



Number Reassembly: Base math problem

```
//pretend b6 = 0x04, b7 = 0x03, b8 = 0x02, b9 = 0x01
```

```
b6 = Byte(gsReceivedData, 6);
```

```
b7 = Byte(gsReceivedData, 7);
```

```
b8 = Byte(gsReceivedData, 8);
```

```
b9 = Byte(gsReceivedData, 9);
```

gsReceivedData comes in
reverse order!!

We want it to be 0x01020304

Number Reassembly: Why use Base math

- We use Base 10 Math, right?
 - What does that mean?
 - It takes 10 digits before we shift left and start a new digit
- If we have Base 5 what happens?
- What's interesting about multiplying something by 10 REGARDLESS of your base?
 - $4 \times 10 = 40$
 - That works for both base 10 AND base 5

Number Reassembly: Base math problem

- Using base 10 math, turn the numbers 9, 2, and 1 to 921
 - $X = 9 * 100$
 - $Y = 2 * 10$
 - $Z = 1 * 1$
 - $\text{Result} = X + Y + Z$

Number Reassembly: Base math problem

- Using base 5 math, turn the number 4, 2, and 3 to 324
 - $X = 3 * 100$
 - $Y = 2 * 10$
 - $Z = 4 * 1$
 - $\text{Result} = X + Y + Z$

Number Reassembly: Base math problem

```
//pretend b6 = 0x04, b7 = 0x03, b8 = 0x02, b9 = 0x01
```

```
b6 = Byte(gsReceivedData, 6);
```

```
b7 = Byte(gsReceivedData, 7);
```

```
b8 = Byte(gsReceivedData, 8);
```

```
b9 = Byte(gsReceivedData, 9);
```

```
r1 = b6; // r1 = 0x04
```

```
r2 = b7 * 0x100; // r2 = 0x0300
```

```
r3 = b8 * 0x10000; // r3 = 0x020000
```

```
r4 = b9 * 0x1000000; // r4 = 0x01000000
```

```
Result = r1 + r2 + r3 + r4; //0x01020304
```

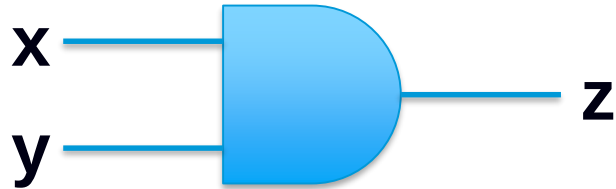
gsReceivedData comes in
reverse order!!

We want it to be 0x01020304

Byte returns 2 Hexadecimal
(base 16) numbers

Why 0x100 instead of 0x10?

Binary Math: AND



Inputs		Output
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

$$z = x \& y;$$

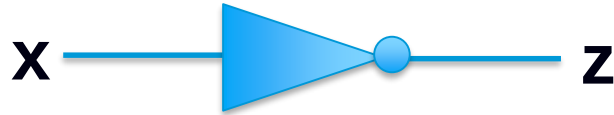
Binary Math: OR



Inputs		Output
x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

$$z = x \mid y;$$

Binary Math: NOT



Input	Output
x	z
0	1
1	0

$$z = !x;$$

Binary Math: Shifting Bits

00001100

slides right two places

= 000011

Function is: **$z = x \gg 2$**

00001100

slides left two places

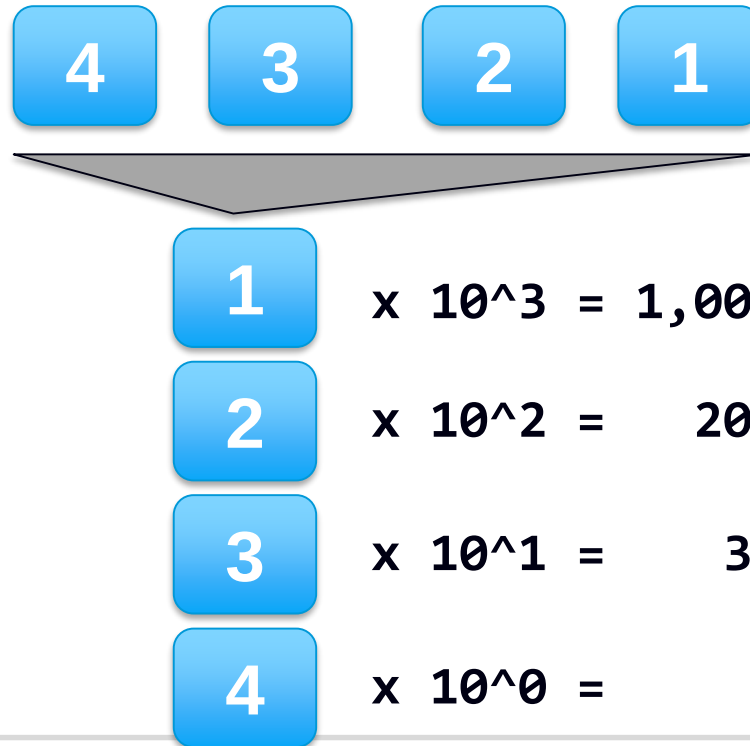
= 0000110000

Function is: **$z = x \ll 2$**

Number Reassembly: Base 10

Starting Number: 1,234

Disassembled into four components (reverse ordered):



Number Reassembly: Hexadecimal

Starting Number: [11] [22] [33] [44]

Disassembled into four components (reverse ordered):



Number Reassembly: Hexadecimal

Starting Number: [11] [22] [33] [44]

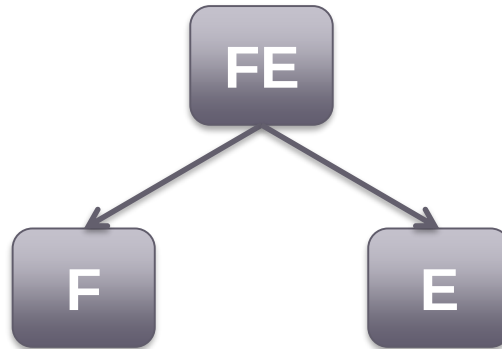
Disassembled into four components (reverse ordered):



11	$\times [10]^6 =$	[11] [00] [00] [00]
22	$\times [10]^4 =$	[22] [00] [00]
33	$\times [10]^2 =$	[33] [00]
44	$\times [10]^0 =$	[44]

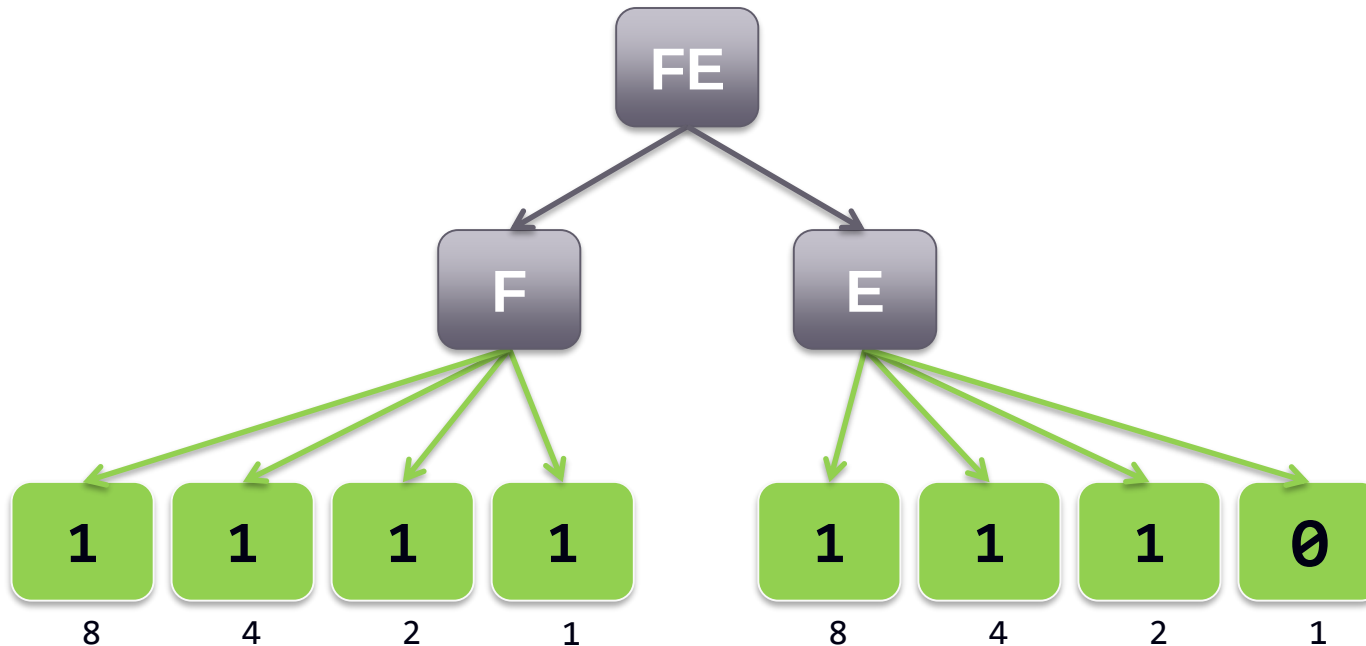
Number Reassembly: Binary

Hexadecimal “Bytes” contain two sets of 4 bits



Number Reassembly: Binary

Each set of 4 bits is a binary “nibble”



Combining two bytes

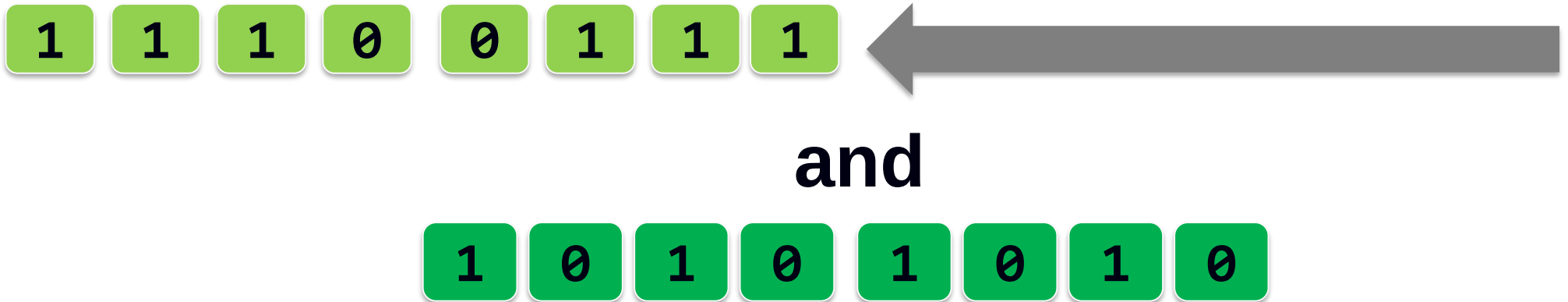
1 1 1 0 0 1 1 1

and

1 0 1 0 1 0 1 0

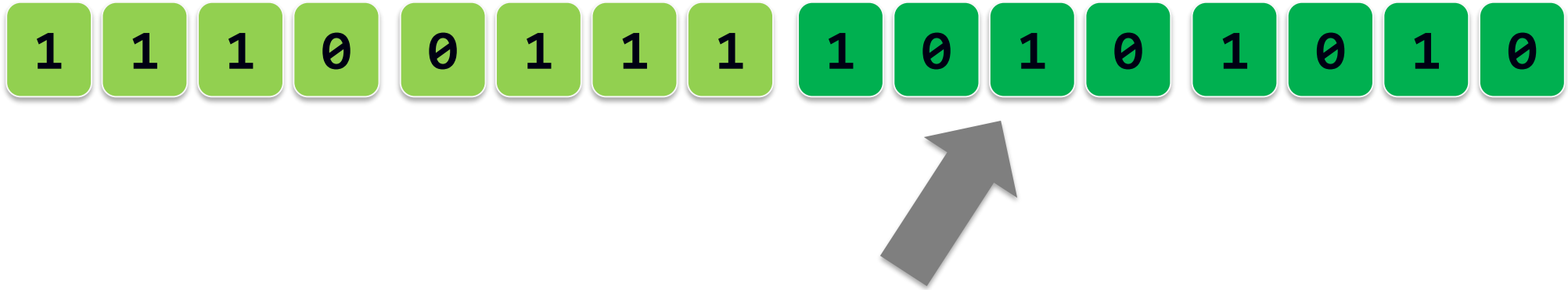
Number Reassembly: Binary

Combining two bytes



Number Reassembly: Binary

Combining two bytes



Binary Result Characteristics



Questions ???



CTI-P301
Day 3
Fun Day!



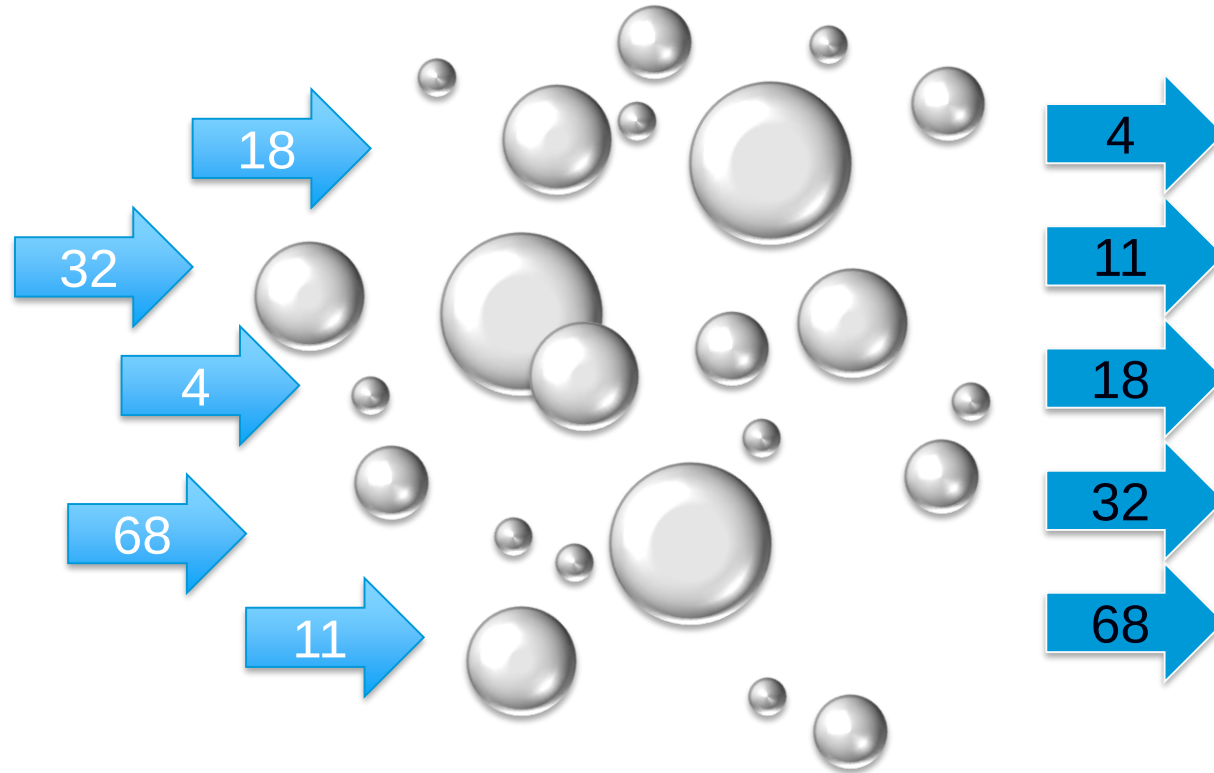
Quiz Time

01010001 01110101 01101001

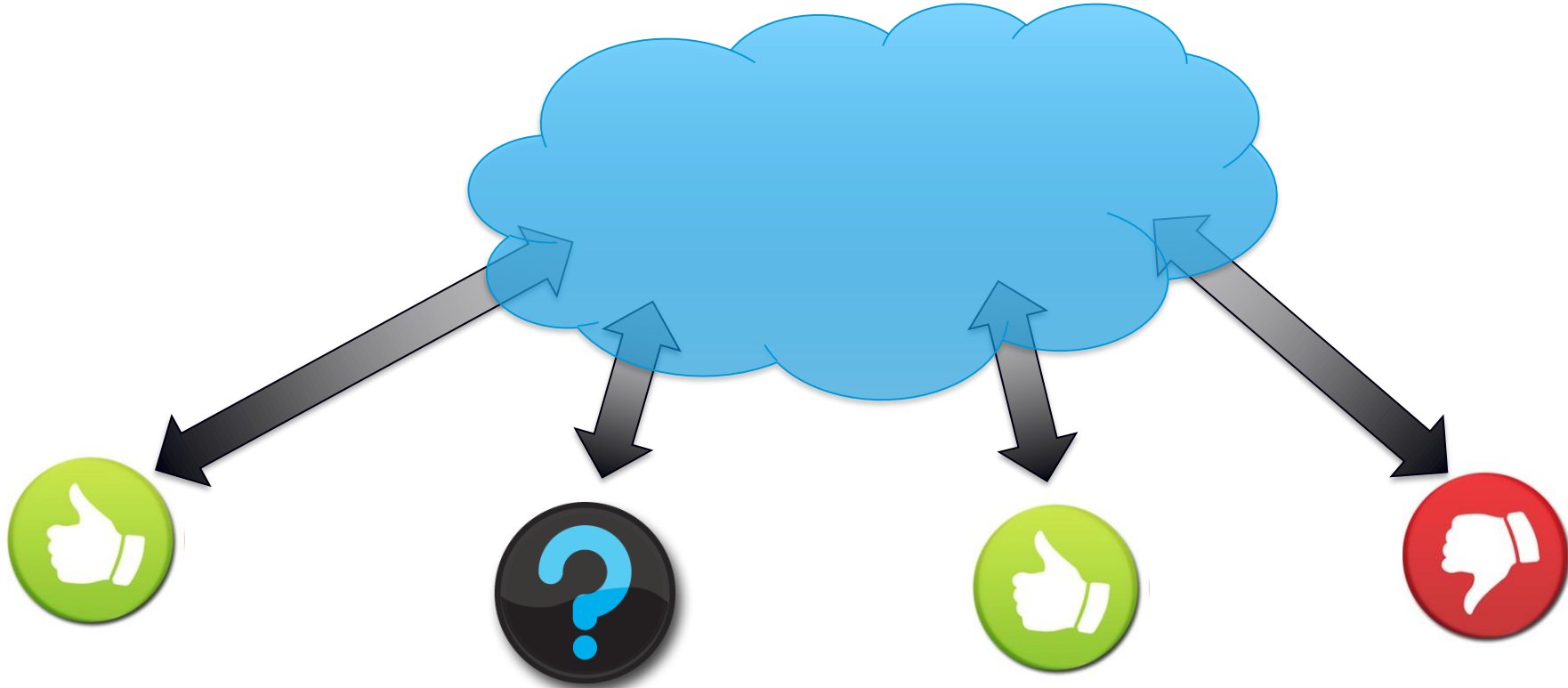
01111010 00100000 01010100

01101001 01101101 01100101

Bubble Sort



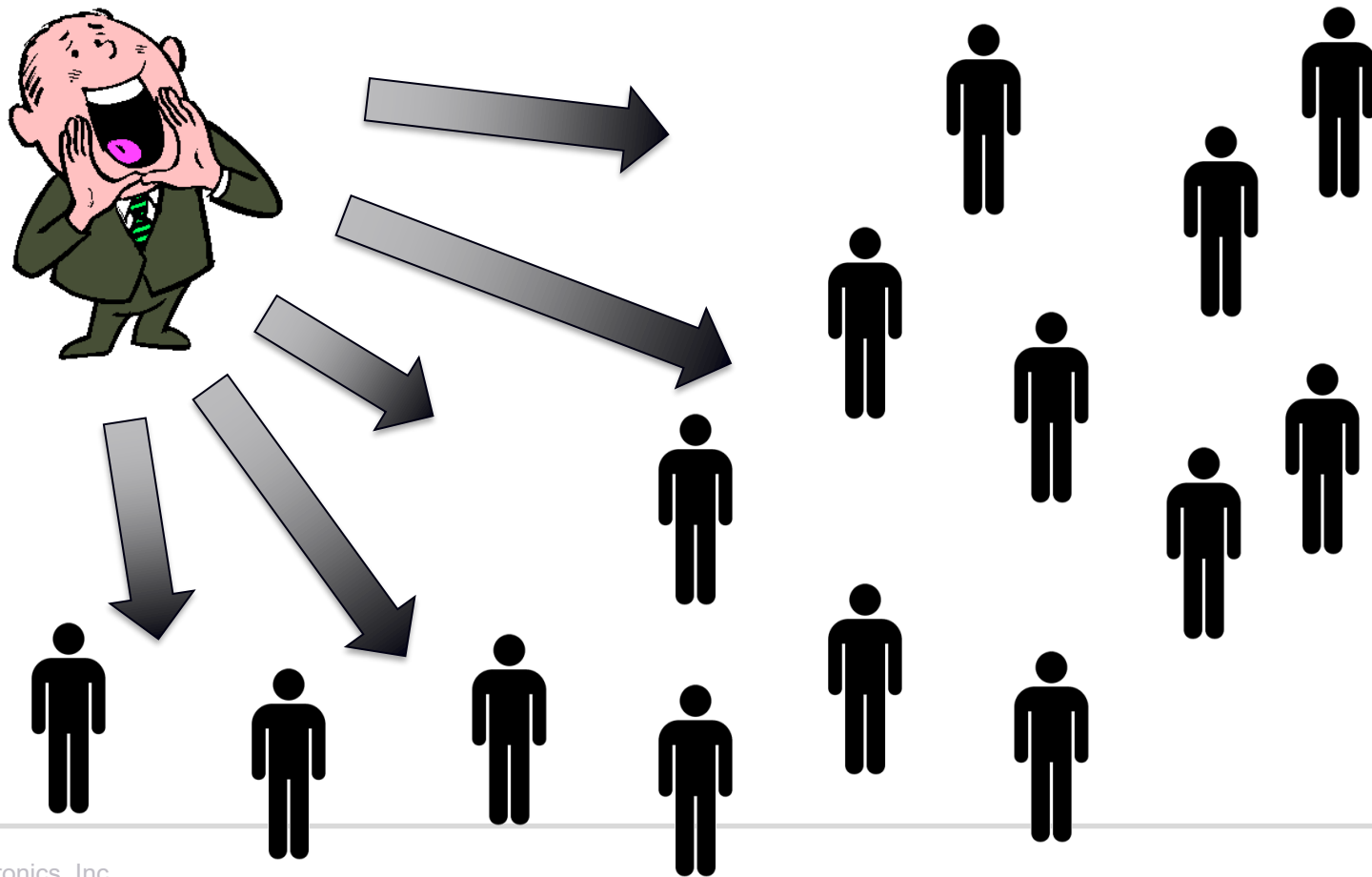
Approval Ratings – UDP Broadcast



TCP Communications Model



UDP Communications Model



Bubble Sort: Pseudoblock

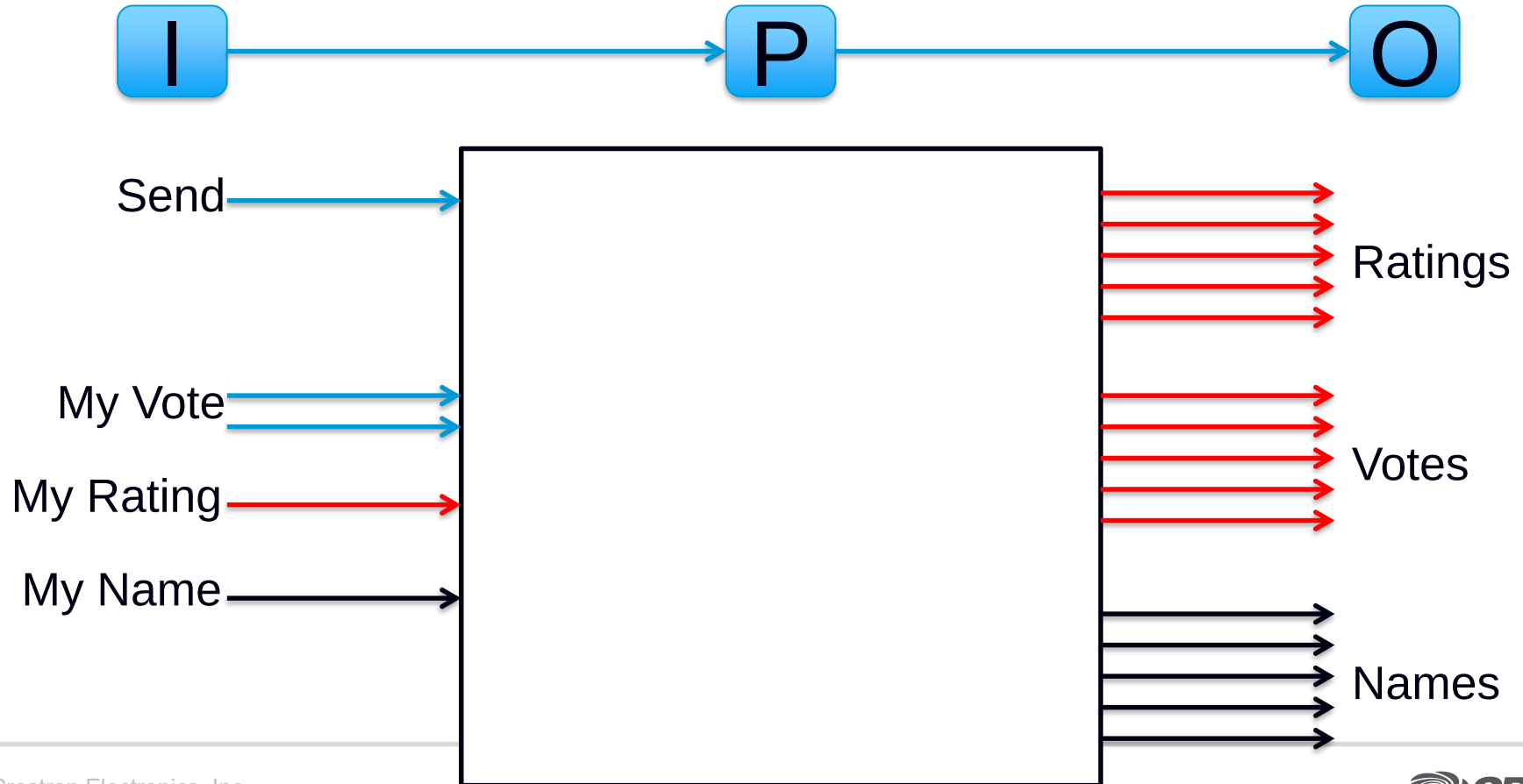


Approvals: Our own Protocol

[F0] <ID Byte> <Vote Byte> <Rating Byte> <Name>[F1]

Byte Number	Description	Data
1	Start Byte	Unique, must be \xF0
2	ID Byte	Player ID number (one through sixteen) in hexadecimal
3	Vote Byte	Byte indicating your vote 0x00 = Undecided , 0x01 = Yes , 0x02 = No
4	Rating Byte	Byte value indicating your rating, from 0d – 100d
5 - ?	Name	Freeform (variable) string field, up to 20 bytes.
Last	End Byte	Unique, must be \xF1

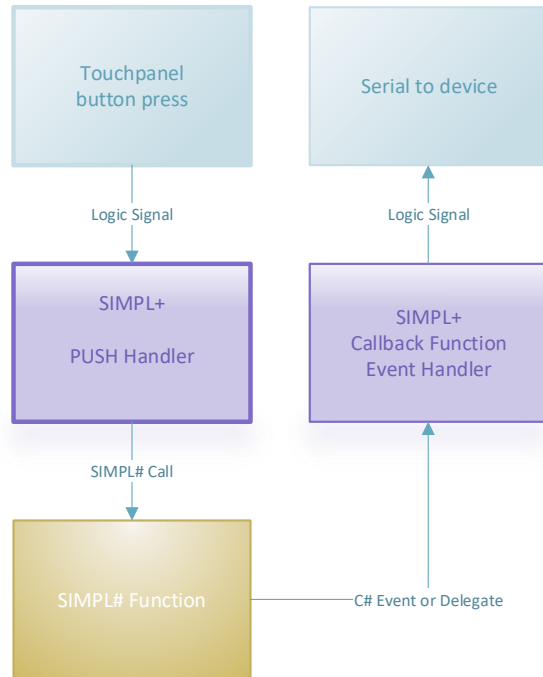
Approval Module: Pseudoblock



SIMPL#

- > Do I need to know C#
- > Example of SIMPL#
 - > MySSH.clz
 - > #User_SimplSharp_Library
- > SIMPL# Class
 - > Crestron API - Right Click Menu in SIMPL+ | Open API

SIMPL#: Best Practices



- > Use SIMPL+ as an interface between logic and SIMPL# only
- > In SIMPL+ do **nothing** other than passing data to/from SIMPL#
 - > One exception: `Gather()`
- > SIMPL+ program cannot have the **same name** as CLZ library

SIMPL#: Basic of C# Constructs

- > Access to class
 - > Dotted notation, like SIMPL+ structure

```
SSHClient myClient;
```

```
Connected_FB = myClient.IsConnected; //Read value
```

- > Case sensitive

```
SSHClient mySSHClient;
```

```
mySSHClient.isConnected;
```

```
mySSHClient.IsConnected; // These two are different
```

SIMPL#: Basics C# Constructs

- Fields / Member variables
 - Like SIMPL+ structure variables
- Properties
 - Like field with distinct difference:
 - Read-only
 - Write-only
 - Read-Write

SIMPL#: Basics C# Constructs

- Delegates
 - Function pointer with pre-specified signature
 - Can assign multiple
 - Allows SIMPL# to call SIMPL+ function

```
public delegate void KeyReceivedDelegate (String Key,  
String KeyName, String Fingerprint);  
...  
public KeyReceivedDelegate KeyReceived;  
...  
KeyReceived ("Key", "Keyname", "Fingerprint");
```

SIMPL#: Basics C# Constructs

- Events
 - Delegate with specific signature
 - Can assign multiple
 - Allows SIMPL# to call SIMPL+ EventHandler
 - Sounds a lot like a delegate! Difference is just in C#:
 - Protected – you can only add / remove yourself

```
public delegate void myEvent(Object sender, EventArgs e);
```

```
...
```

```
Public event myEvent onEventTriggered;
```

SIMPL#: Using SIMPL# Library in SIMPL+

- + What is a CLZ?
 - Container with all files required
- + Where do I put it?
 - Project folder
 - Anywhere else

SIMPL#: Let's do it

- + Copy SSHClient.CLZ to project folder
- + Create a SIMPL+ Module
 - Make sure to use a different name for the SIMPL+ Module
 - Use the standard Template we have been using
- + #INCLUDEPATH
 - Tell S+ where to find the CLZs
 - You can have more than one
 - Not needed when CLZ is in project folder so we won't add this # define this time
 - #INCLUDEPATH "C:\\MyCLZs"
- + #USER_SIMPLSHARP_LIBRARY
 - Tell SIMPL+ what CLZ to load
 - No path / extension
 - Do the following in the compiler directive location of SIMPL+
 - #USER_SIMPLSHARP_LIBRARY "SSHClient"

SIMPL#: Now Check the API

- How do you know what to access in the CLZ?
 - Add your Compiler Directives for the class
- Now save your SIMPL+ file to the same directory as the CLZ.
 - Remember that it cannot have the same name
 - Call it something like mySSHClientP301.usp
- Compile your SIMPL+ file to confirm your directive is correct.
- Now right-click in SIMPL+ editor
 - Select “Open API For ...”
 - This the Application Programming Interface for the class that is generated when the programmer compiles the CLZ from the C#.

SIMPL#: Let's Review the API

```
class SSHClient
{
    // class delegates
    delegate FUNCTION KeyReceivedDelegate ( SIMPLSHARPSTRING Key , SIMPLSHARPSTRING KeyName , SIMPLSHARPSTRING Fingerprint );

    // class events
    EventHandler onCommandResponseReceived ( SSHClient sender, CommandResponseArguments e );
    EventHandler onShellStreamDataReceived ( SSHClient sender, AsynchronousDataArguments e );

    // class functions
    INTEGER_FUNCTION Connect ( STRING Host , STRING Username , STRING Password , INTEGER port );
    FUNCTION Disconnect ();
    FUNCTION SendCommand ( STRING command );
    FUNCTION InjectCommandToShell ( STRING cmd );
    STRING_FUNCTION ToString ();
    SIGNED_LONG_INTEGER_FUNCTION GetExitCode ();

    // class variables
    INTEGER StreamBufferSize;

    // class properties
    DelegateProperty KeyReceivedDelegate KeyReceived;
    INTEGER IsConnected;
};
```

The diagram illustrates the SIMPL# API structure for the `SSHClient` class. It highlights several key components:

- Events:** Indicated by a blue arrow pointing to the `EventHandler` events `onCommandResponseReceived` and `onShellStreamDataReceived`.
- Class:** Indicated by a green arrow pointing to the `class SSHClient` declaration.
- Delegates Functions:** Indicated by a blue arrow pointing to the `delegate FUNCTION KeyReceivedDelegate` definition.
- Methods:** Indicated by a blue arrow pointing to the `FUNCTION` methods like `Connect`, `Disconnect`, `SendCommand`, `InjectCommandToShell`, `ToString`, and `GetExitCode`.
- Fields:** Indicated by a blue arrow pointing to the `INTEGER StreamBufferSize` variable.
- Properties:** Indicated by a blue arrow pointing to the `DelegateProperty KeyReceivedDelegate KeyReceived` and `INTEGER IsConnected` property.

SIMPL#: Review SSHClient S+ Module

- > We have included Library SSHClient
 - > `#USER_SIMPLSHARP_LIBRARY` “SSHClient”
- > We have saved your SIMPL+ Module
 - > We made sure that the SIMPL+ is not the same name!
 - > We called something like “mySSHClientP301.usp”
- > We right clicked and reviewed the “Open API for SSHClient...”

SIMPL#: Create the SIMPL Windows I/O

- Let us deal with SIMPL Windows I/O first to set up our variables we will use later.
- Create in the SIMPL+ module the following Inputs:
 - **DIGITAL_INPUT** Connect, Disconnect;
 - **ANALOG_INPUT** Port;
 - **SERIAL_INPUT** Hostname[255], UserName[25], Password[25];
 - **BUFFER_INPUT** SendCommand\$[512], InjectToShell\$[512];
 - **DIGITAL_OUTPUT** Connected;
 - **STRING_OUTPUT** Command\$, Response\$;

SIMPL#: Create the SSH Client S+ Module

- › Create instance of that client in SIMPL+ and give it a SIMPL+ variable

```
SSHClient myClient;
```

```
// In C#, this is the equivalent to:
```

```
// SSHClient myClient = new SSHClient();
```

- › Always calls default constructor
- › There is no way to call overloaded constructors
- › Now you add global variable called MyClient of type SSHClient
 - › SSHClient MyClient;
- › Yes you can add more than one
 - › Just add a new SIMPL+ client global variable to do so but let us use just one for now.

SIMPL#: Create the SSH Client S+ Module

> Create SIMPL+ event stubs:

> PUSH Connect

```
{  
}
```

> PUSH Disconnect

```
{  
}
```

> THREADSAFE CHANGE SendCommand\$

```
{  
}
```

SIMPL#: SIMPL# Classes

> Calling methods

```
MyClient.Connect(Host, UserName, Password, Port);  
MyClient.SendCommand(Command);  
MyClient.Disconnect();
```

> Calling methods with return types

```
INTEGER Result;  
Result = MyClient.Connect(Host, UserName, Password, Port);
```

SIMPL#: Create the SSH Client S+ Module

> In Push Disconnect

```
{  
    Call MyClient.Disconnect();  
    If(!MyClient.IsConnected) Connected = 0;  
}
```

> In PUSH Connect

```
{  
    MyClient.StreamBufferSize = 65534;  
    MyClient.Connect(Host, Username, Password, Port);  
}
```

SIMPL#: Create the SSH Client S+ Module

```
THREADSAFE CHANGE SendCommand$
{
    STRING Command[512];
    WHILE (MyClient.IsConnected)
    {
        Try
        {
            Command = Gather("\n", SendCommand$);
            IF (MyClient.IsConnected)
                MyClient.SendCommand(Command);
        }
        Catch
        {
        }
    }
}
```

SIMPL#: Review What we done so far

- SIMPL Windows Interface is done.
- In PUSH Connect
 - Call Connect()
- In Push Disconnect
 - Call Disconnect()
- In THREADSAFE CHANGE SendCommand\$ / InjectToShell\$
 - While (1)
 - Try / Catch
 - Gather("\n", SendCommand\$) / Gather("\n", InjectToShell\$)
 - Call SendCommand() / InjectCommandToShell()

SIMPL#: Classes - Events

> Events

- > Create event handler to execute when event is triggered
 - > Copy from the API the event handler and adjust it.

```
EventHandler onCommandResponseReceived ( SSHClient sender, CommandResponseArguments e )  
{  
    PRINT("Command: %s - Response: %s\n", e.Command, e.CommandResponse);  
}
```

- > Register the event handler to the event added to Function Main()
 - > Must register all events before they are used. Main() will do this for us.

```
RegisterEvent(MyClient, onCommandResponseReceived, onCommandResponseReceived);
```

SIMPL#: Classes - Delegate

- Create delegate stub CALLBACK FUNCTION KeyReceivedHandler
- + **CALLBACK** FUNCTION KeyReceivedHandler (STRING Key , STRING KeyName , STRING Fingerprint)

```
CALLBACK FUNCTION KeyReceivedHandler ( STRING Key , STRING KeyName , STRING Fingerprint )  
{  
    Print("Key : %s - Keyname: %s - Fingerprint: %s\n", Key, KeyName, Fingerprint);  
}
```

- Just like events. We must register our delegate functions
 - **RegisterDelegate** (myClient, SIMPL# var, SIMPL+ var);

```
RegisterDelegate(MyClient, KeyReceived, KeyReceivedHandler);
```


SIMPL#: Add to SIMPL Windows

- Create SIMPL Windows program
- Drop your S+ module into it
- Add Symbols needed and Signal names.
- Compile / Upload
- Use SIMPL Debugger for testing:
 - Set hostname to PRO3 IP address
 - Username = Crestron
 - Password = *blank*
 - Port = 22

SIMPL#: Review

- You created a SIMPL+ module with
 - inputs and outputs from / to SIMPL Windows
 - Added SIMPL+ events to trigger our SIMPL#
 - Waited for SIMPL# events to trigger our SIMPL+ module.
- You created SIMPL Windows program
- You dropped in your SIMPL+ module into your program which included your SIMPL# calls and responses.
- You tested it on a processor with debugger.
- More information can be found at
 - Reference.Crestron.com
 - Chapter 2

Questions ???



Advance debugging

How best to troubleshoot a problem with a Crestron system.

- First, diagnose where the problem lies
 - Video / Audio (routing) issues
 - Touch Screen feedback issues
 - Delay / Performance issues
 - Unexpected behavior (lights turning on/off for no “apparent” reason)
 - Not controlling 3rd party devices

Advance debugging

Video / Audio issues should always be looked at by starting at an endpoint.

- Start with a source or a sync and work back toward the other end.
- If source is viewable on other sync in system then this can be a EDID issue for that sync.
 - Check the devices capabilities.
- If Source is not viewable on other sources connect it directly to small display to confirm output.
 - Check source setup menus for setting related to fixed EDID outputs

Advance debugging

Touch Screen feedback issues:

- Same “incorrect” feedback on all UI?
(i.e. is it just one touch screen, or all)
 - If Yes: Check your program
 - If No: Compare two Touch Screen symbols, possibly copy/paste error
 - If No: Run-as in VTPro, connect to the same IP ID.
 - Problem exists: Check program/symbol/project
 - Problem doesn't exist:
 - Wireshark, compare packets between XPanel and CS to Touch Screen and CS
 - DbgPktRx and DbgPktTx

Advance debugging

Delay / Performance Issues

- Touch Screen: Check behavior and see if it is program and project specific.
- Program/Control System
 - CPULoad (at least 3 times, ignore 1st value):
 - 100% is NOT a problem, unless there is a problem (3-Series will use as much CPU as is available by design)
 - RAMFree
 - < 80% : Healthy
 - 80-85%: Questionable
 - > 85%: Unhealthy, split programs over multiple processors.
 - Review Error log:

Advance debugging

Delay / Performance Issues

- Handle all S+ errors that are under your control
 - Thread Management
 - THREADPOOLINFO
 - Maximum number of threads:
 - $5 + 2 \times \text{Count}(\text{S+ modules}) + \text{NumberOf}(\text{running Gather})$
 - Active threads:
 - Number of threads in the pool ready to go
 - Waiting Callbacks
 - If growing, and not going down: threads are not exiting
 - THREADPOOLSIZExXX (max 400)
 - Threads currently executing
 - LISTACTIVEMODULES
 - Gives DLL names
 - SPLUSTASKS
 - Gives priorities and S numbers and runtime

Advance debugging

Delay / Performance Issues

+ Multiple Programs running

- Stop other programs, until you find the slow one.
 - If you don't find the slow one, you may have exceeded CPU performance in single box
 - Look at the built in diagnostic web page
- Work with any Toolbox Tool to diagnose the issue when the program is running.
 - Diagnostic web page can be used as well.

Questions

- Questions for You
- Questions from Crestron
- Become Certified!



Certification



