FPP Notes –

Blue – Sections, Green – Part of Examples, Yellow – Don’t know much about it, Red – Limitations/Errors

Section 3.1

Defining Constants ex: constant ultimateAnswer = 42

Can’t have it undefined

Section 3.2

You can put a $ in front of reserved words ex: $constant

Can’t have two symbols (variables) with the same name : constant c = 0, constant c = 1. The only time that you can have the same symbols is if the are in two different modules, struct and array may not have same name but array and constant can.

Section 3.3

Primitive values: decimal integer, hexadecimal, floating-point literal, Boolean

String uses “” ex: constant s “ This is a string”

Having \ or “ use \ ex: “\”This is a quotation within a string,\” he said.” Output: ”This is a quotation within a string,” he said.

Multi line uses “”” ex:

“””

This is a multiline string

Hello

“””

Array values ex: constant a = [1,2,3], cannot be empty ex: constant a =[], up to 256 elements, and elements

Structs ex: constant s = { x = 1, y= “abc”}, order does not matter and it can be empty. They can struct members as arrays

Arithmetic: negative, +, -, *,/,()

Constant a = 9 is the same as constant b = [3,3]

Section 3.4

You can put the same definitions on the same line separated by a semi-colon

Ex: constant a=1; constant b =2

Section 4.1-4.2

Commits use #, annotations use @

Section 5

Modules: module M {} like namespaces/classes in c++, refer back to it in variable like constant a = M.a

Ex: module M {

Constant a = 1

}

Constant a = M.a

XML limitations with nested modules

Section 6.1

Array Type Definitions : array A = [numElements] U32?

Unsigned integers ex: U8,U16,U32,U64, Signed integers: I, Floating type: F32,F64,

Defaults are like filling an array with a default number or array of numbers to another array

Format strings are how the variable gets produced ex: `array WheelSpeeds = [3] U32 format "{} RPM" , 100 RPM`

Replacement Fields: {c} Character, {d} decimal, {x} hexadecimal, {o} octal, {e} rational value ex: 1.234e2, {f} Rational value fixed ex: 123.4, {g} rational value in general format

Can have Array of arrays

Section 6.2

Defining Struct ex: struct S { x:U32, y: string}

Defaulting can default only one value because it will automatically default them to a value if called and cannot default to a value not in the struct

Format? 6.2.5

Struct types the order matters, this means serializing to deserializing will produce garbage

Section 6.3

Abstract Types?

Section 7.1

Enum has three decisions: Yes, No, Maybe

Section 7.2

Using Enum ex: enum State {ON, OFF}

Constant initialState = State.OFF

Defaulting: default State.OFF

Section 7.3

Like C++ with Boolean Yes has value 0, B has 1 and Maybe has 2. Can provide specific value but have to specify all and they cant be the same number

Can use enum as a number for an operation if you call it ex: enum E {A=5}, constant c = E.A + 1

Section 7.4

Types for enum have to written enum Small : U8 {A,B,C}

Section 8.1

Ports that carry no data is useful to sending and receiving a triggering event

Section 8.2

Valid types of an abstract type defining ex: type T