**START CODE: J18CUSA8E6N062315014880T**

**We are explaining back and forward to better understand our logic.**

**Separate code into subcategories and validate (check If 24 of size and the hours and dates):**

Format Code: **J18C** [0, 3] “UPU identifier [0,0] + Format Identifier [1, 3]”

Issuer Code: **USA** [4,6]

Equipment Identifier: **8E6** [7, 9]

Item Priority: **N** [10,10]

Month: **06** [11, 12]

Day: **23** [13, 14]

**SERIAL NUMBER**

Hour: **15** [15, 16]

Minute in increments of 10: **0** [17, 17] (ranges from 0 to 5)

Item Number Part: **14880** [18, 22] (can’t be higher than 16383)

Tracking Indicator: **T** [23, 23]

**Convert code to binary representation using these different tables (maps on begin of cpp):**

* Predefined code conversion table for converting the Issuer Code (z to a, 9 to 0) (**cc**)
* Converting the Item Priority (**pc**)
* Converting the Tracking Indicator (**tc**)

# **J18C** -> 0010 (1) / 4 bits

    # **USA** | From the cc table: U: 5, S: 7, A: 25 | 1600\*5+40\*7+25 -> 0010000001110001 (2) / 16 bits

    # **8E6** | Direct conversion from hexadecimal to binary -> 100011100110 (3) / 12 bits

    # **N** | From the pc table -> 00 (4) /2 bits

    # **06 23 15 0 14880** | The serial number is converted with a series of calculations |

16384 \* (5120 \* (**06**-1) + 160x**23** + 6 \* **15** + **0**) + **14880** -> 011100101011101011101000100000 (5) / 30 b

    # **T** | From the tc table -> 00 (6) / 2bits

RESULT is: (1).(2).(3).(4).(5 from the most significant bit count 20 bits).(6).(the rest of the bits from 5) ->

001000100000011100011000111001100001110010101110101110001000100000 /66 b

**Finish the code to binary conversion:**

leftsync = "010110";               #22 decimal

rightsync = "100110";              #38 decimal

* Predefined codes to be inserted in the binary code.

From the least significant bit, count 12 bits and insert the rightsync:

001000100000011100011000111001100001110010101110101110100110001000100000

From the most significant bit, count 12 bits and insert the leftsync:

001000100000010110011100011000111001100001110010101110101110100110001000100000 / 78 bi

**Convert the binary code to decimal**

Create a list of 12 zeros: {0,0,0,0,0,0,0,0,0,0,0,0}

Starting from the least significant bit, in groups of 6 bits, convert the binary groups to decimal numbers and append them to the list:

100000 -> 32

001000 -> 8

….

Final result:

{0,0,0,0,0,0,0,0,0,0,0,0, 32, 8, 38, 46, 46, 50, 33, 57, 24, 28, 22, 32, 8} / 13 non zero numbers

Then, until the list has exactly 63 elements, zeros are appended to the list:

C: {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 32, 8, 38, 46, 46, 50, 33, 57, 24, 28, 22, 32, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

5th) Calculate Reed-Solomon error correction

Final result (R): {16, 43, 31, 52, 29, 49, 6, 31, 7, 8, 13, 35}

Then convert back to binary:

8, 32, 28, 24, 57, 33, 50, 46, 46, 8, 32 -> 001000100000011100011000111001100001110010101110101110001000100000

TODO: CONVERT USING REED-SOLOMON ERROR DETECTION

**Converting the serial number back:**

011100101011101011101000100000 -> 481212960 in decimal

481212960 / 16 384 = 29 370 + 14 880 / 16 384 -> Item Number

29 370 / 5120 = 5 + 3770 / 5120 -> month -1

3 770 / 160 = 23 + 90 / 160 -> day

90 / 6 = 15 + 0 / 16 -> hour and minute

# GALOIS FIELD – Reed-Solomon Basics

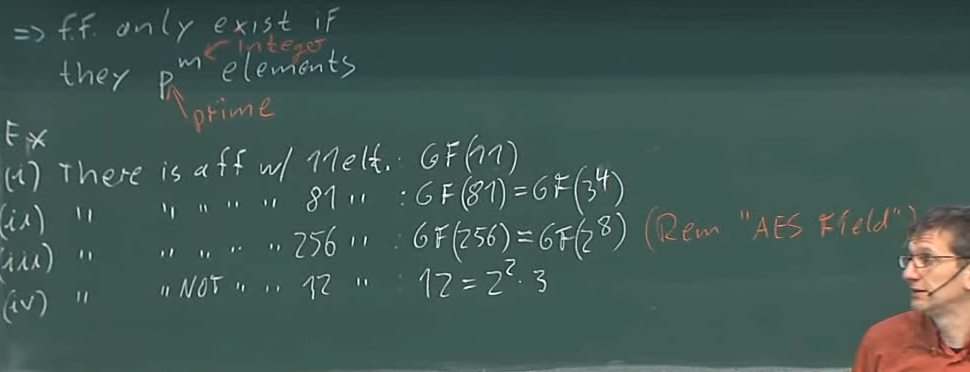


Figure 1 - Finite field dimensions.

You can’t make, for example, a finite field of 12 element because it doesn’t follow the rule: p^n where p is a prime number. There exist prime fields (size p) and extension fields (size p^n).

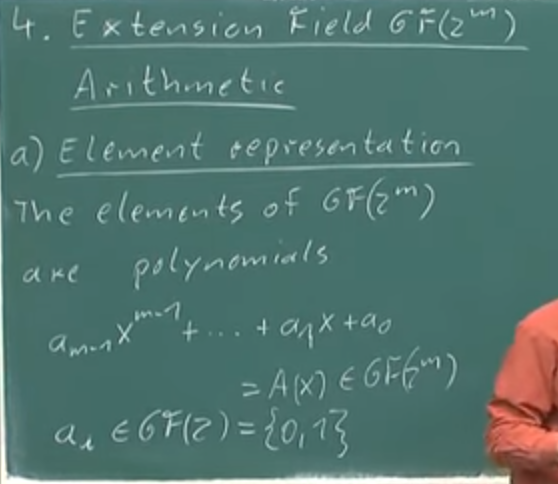


Figure 2 - Extension fields element representation.

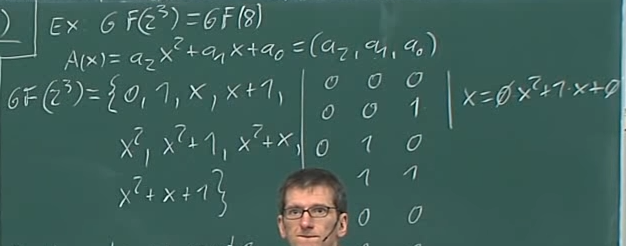


Figure 3 - Example of a extension field.

Irreducible polynomials are “prime polynomials” or polynomials that you can’t divide. In a prime field, multiplication and division is done with a modulo of a prime number. In an extension field, you use the irreducible polynomials. The irreducible polynomials can only be divided by 1 and themselves.

## Extension fields operations.

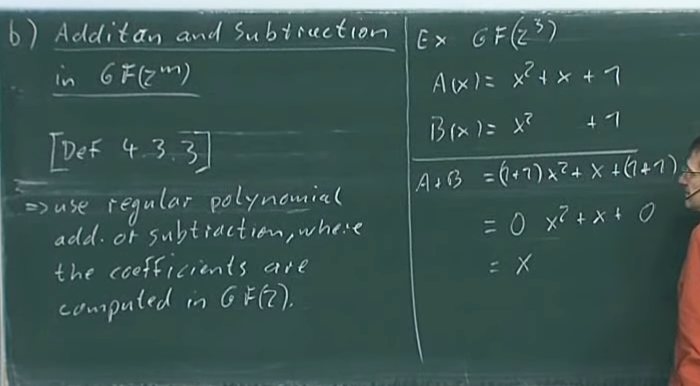


Figure 4 - How to add and subtract in an extension field.

Adding and subtracting are the same operations, and are very easy to compute. (EXCLUSIVE OR / XOR between two elements).

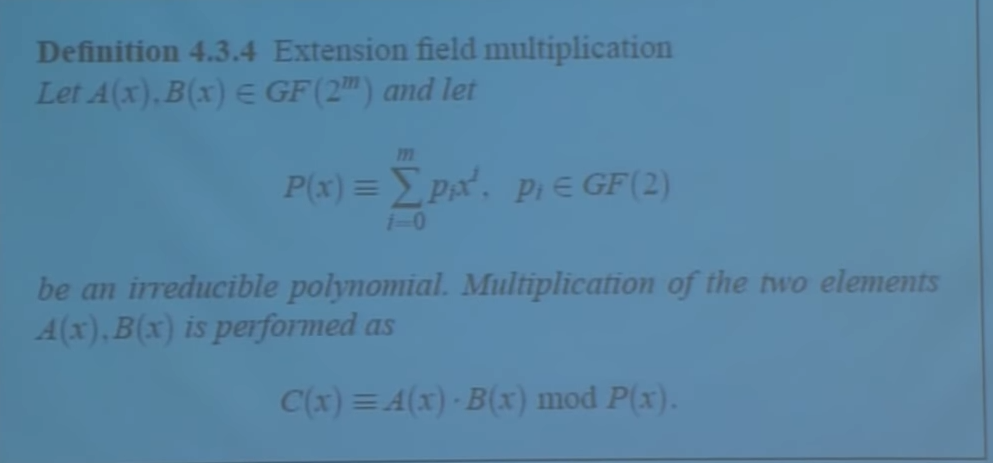
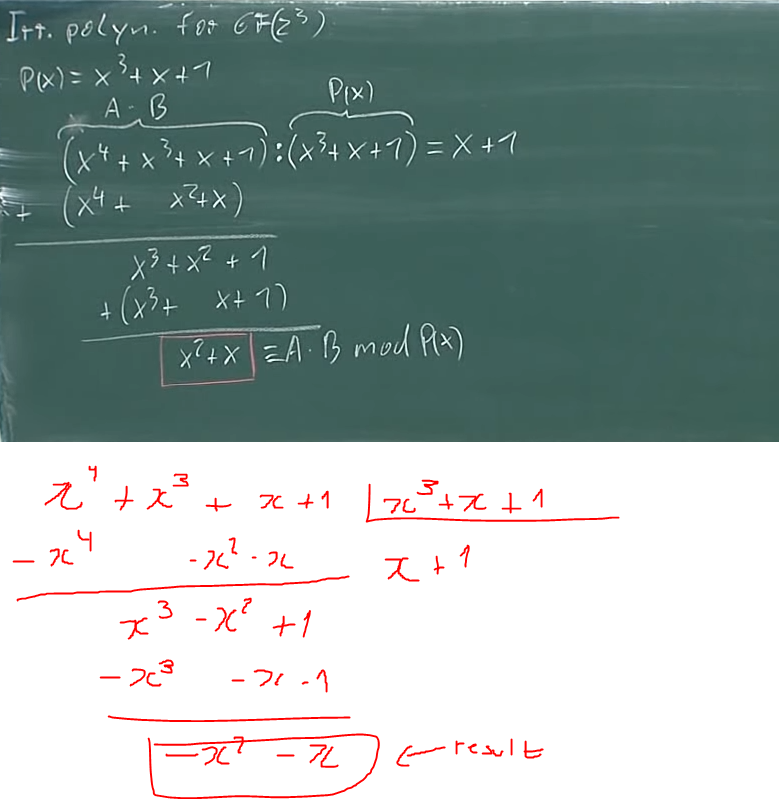
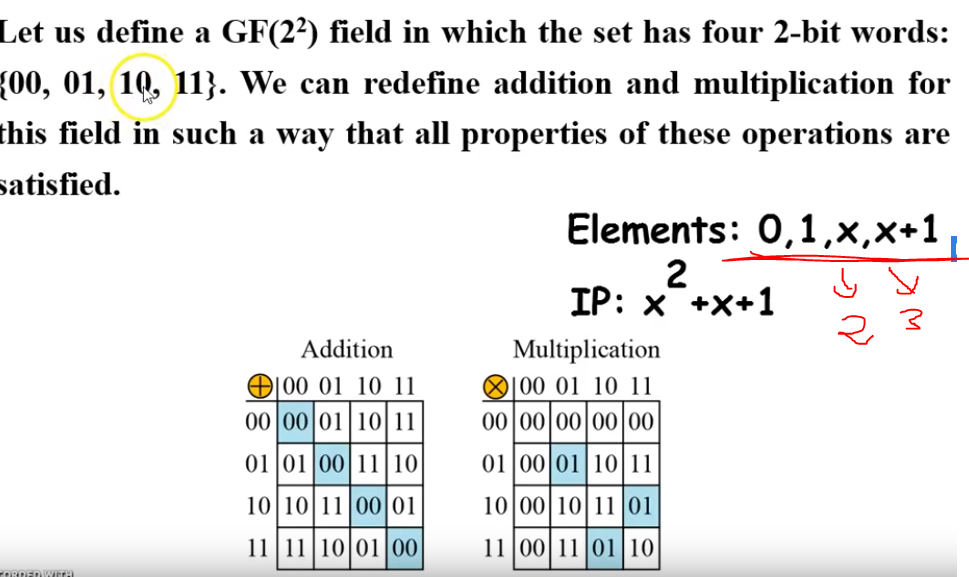


Figure 5 - Multiplication on extension fields ( P(x) is an irreducible polynomial that belongs in the field)



## Examples

Galois field polynomial representation example:



In this picture we can see the multiplication results don’t make sense, because up until now we have only seen prime galois fields. For example, if we do 10 \* 11 (mod 4(dec)) we get 6 mod( 4) = 2 which is not what we get on the multiplication table (the multiplication table shows 01 not 10). This is done so with the multiplication operation on extension fields.

## What does this mean for S18-C?

The encoder uses a galois field of the size 2^6 = 64 (each code word is comprised of 6 bits).

The irreducible polynomial is = x^6+x+1 = 67 = (2^6 + 2 +1)