

C Programming

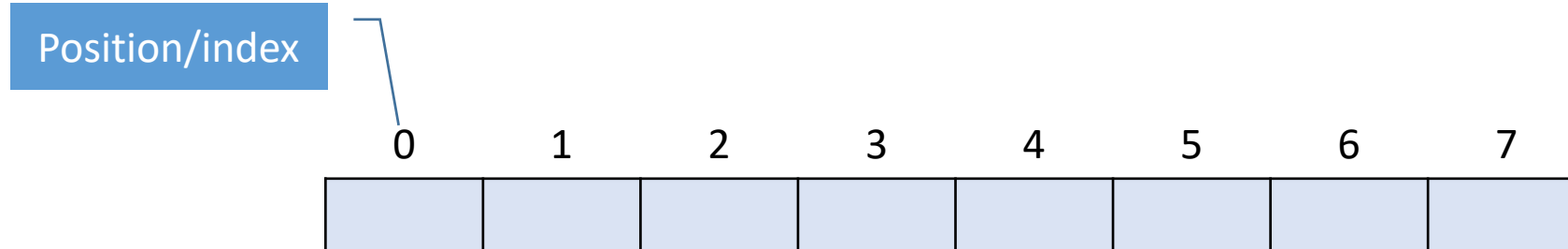
1DT301

Lecture 3

[OPNOVA]
ENGINEERED INNOVATION

Arrays

Layout

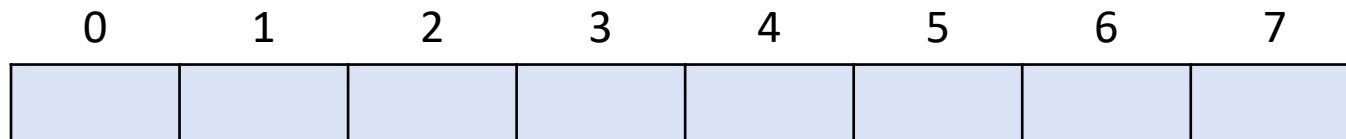


Group of consecutive cells. Compare to the actual data memory!
Each box has the same size (depends on its declared type).
Each box has a position (*starts at 0!*).

Creating an array

```
type variable_name[number_of_boxes];
```

```
int ivec[8]; // Each box is an int (2 bytes)  
char cvec[8]; // Each box is a char
```



Accessing Content

`array_name[position]`

```
int my_arr[8];  
:  
my_arr[2] = 45;  
int my_int = my_arr[1];
```

	0	1	2	3	4	5	6	7
my_arr:	2	55	45	6	64	1	9	456
my_int:	55							

N-Dimensional arrays

```
int matrix[3][8];  
int *p_matrix = matrix;
```

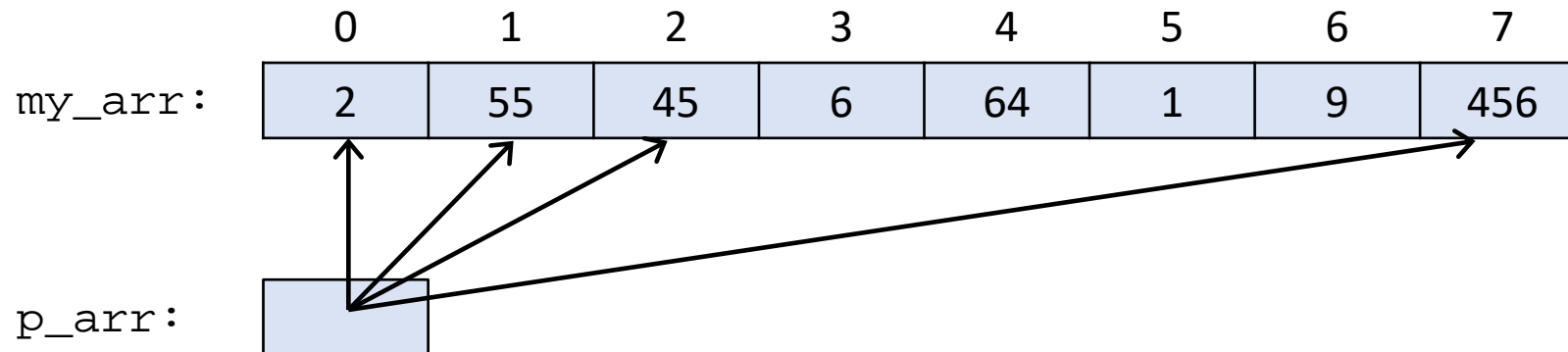
0	1	2	3	4	5	6	7



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

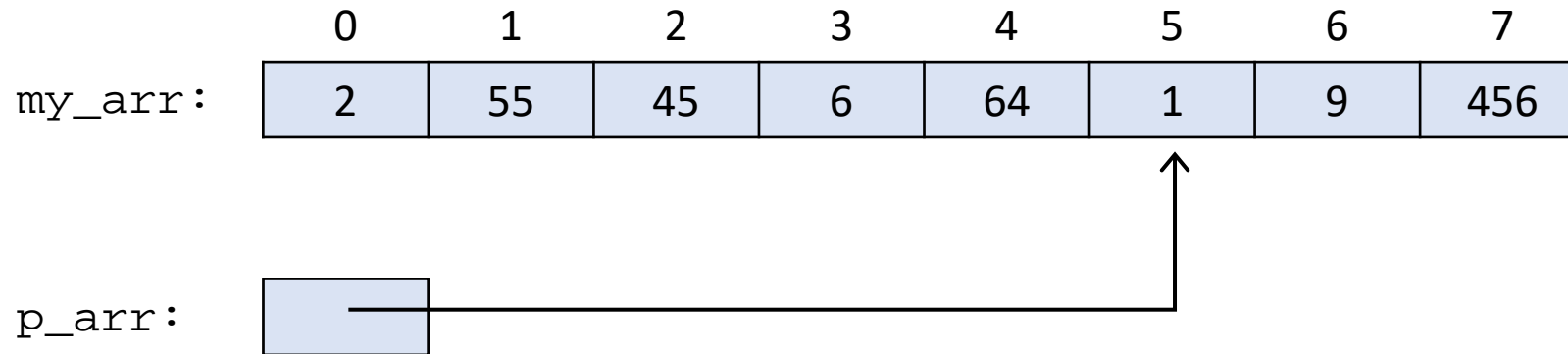
Arrays and Pointers

```
int my_arr[8];
int *p_arr = my_arr;
:
int cntr = 0;
while( cntr < 8)
{
    printf("%i", *p_arr);
    p_arr++;
    cntr++;
}
```



Referencing a Location

```
int *p_arr = &my_arr[5];
```



String

```
char *str = "c is fun";
```

0	1	2	3	4	5	6	8	8
'c'	' '	'i'	's'	' '	'f'	'u'	'n'	'\0'

Array of chars having '\0' as it's termination marker.

Character

- Encoded 'internally' by an 8-bit integer (ASCII)
 - 'A' → 65
 - 'a' → 97
- Have a look at <http://www.asciitable.com/>
- Type is char
- There are other encodings
 - UTF-8, UTF-16 etc

String functions (string.h)

- `strcpy()` - Copy string.
- `sprintf()` - Formatting a string (print to buffer/string).
- `strcat()` - Concatenate.
- `strlen()` - Get string length.
- ...

Example

- Convert a number to hex and as a string.
- Sum all even numbers in an integer array.
- Compute the number of a's in a string. Print the number as a hexadecimal integer.

Bithandling

Bitwise Operators

Operator	Meaning
~	Negation (not)
&	And
	Or
^	Exclusive or (xor)

Shift Operators

Operator		
<<	Left shift	Bit-pattern << number-of-bits-to-shift
>>	Right shift	Bit-pattern >> number-of-bits-to-shift

Note!

Arithmetic shift vs. Logical shift.

Arithmetic → Preserves sign bit .

Logical → Always shifts in 0's.

Compiler dependent!

Overflowing bits are discarded (usually).

Shift example (logical)

1. `unsigned int ui = 149;`

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1

2. `ui = ui << 2;`

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1

 <<00

3.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	1	0	1	0	1	0	0

Shift Example (logical)

1. **unsigned char** ui = 149;

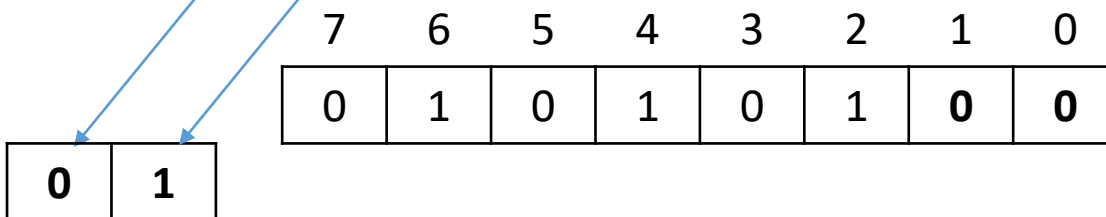
7	6	5	4	3	2	1	0
1	0	0	1	0	1	0	1

2. ui = ui << 2;

7	6	5	4	3	2	1	0
1	0	0	1	0	1	0	1

 <<00

3.



These are dropped!

Masking

- Getting or setting one or more individual bits in a byte, word etc
 - E.g is the bit at position 3 set or not ?

Example

22.9.2 UCSRnA – USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREN	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Check if both the RXC0 and UPE0 flags are set.

Assume register value is found in the UCSR0A variable
(of type unsigned char).

Functions

Functions

- A way to break a large program into smaller tasks
- Reusability
- Abstractions
- Make them small !
- All variables are local to the function!
- Need pointers to affect variables outside the function.

Formal

```
return_value_type  
function_name(parameter_list)  
{  
    declarations  
    statements  
}
```

Argument vs. Parameter

- Arguments are supplied when calling
- Parameters are the associated local variables

Type of
return value

Function
header

Function
body

Parameters

Arguments

```
int add(int i1, int i2)
{
    int i3;
    i3 = i1 + i2;
    return i3;
}
```

```
int main(void) {
    int a = 23;
    int b = 55;
    int ret = add(a, b);
    return EXIT_SUCCESS;
}
```


Function call & scope

```
int add(int i1, int i2)
{
    int i3;
    i1 = i1 + 3;
    i3 = i1 + i2;
    return i3;
}
```

```
int main(void) {
    int a = 23;
    int b = 55;
    int ret = add(a, b);
    return EXIT_SUCCESS;
}
```

When add() is called


i1 = a
i2 = b

i1, i2, i3 are local and do not exist outside the function
(call-by-value)!

Has no effect on a.

Affecting variables outside the function.

Means 'Not returning anything'.

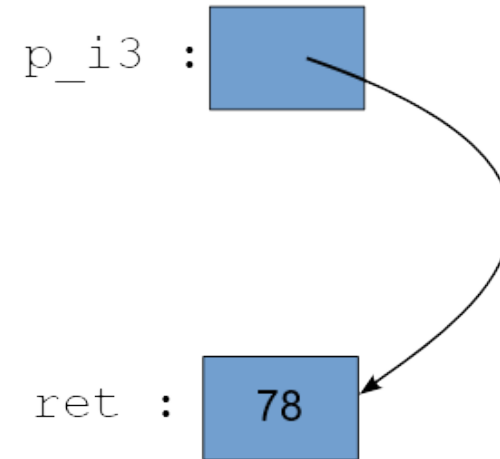


```
void add(int i1, int i2, int *p_i3)
{
    *p_i3 = i1 + i2;
}
```

```
int main(void) {
int a = 23;
int b = 55;
int ret;

    add(a, b, &ret);

    return EXIT_SUCCESS;
}
```



Example – swap ()

```
void swap(int *p1, int *p2)
{
    int temp;
        temp = *p1;
        *p1 = *p2;
        *p2 = temp;
}

int main(void) {

    int i1 = 23;
    int i2 = 77;
    swap(&i1, &i2);

    return EXIT_SUCCESS;
}
```

Example

- Create a function that calculates the square of a float value.
 - Using normal return
 - Using returning through params.

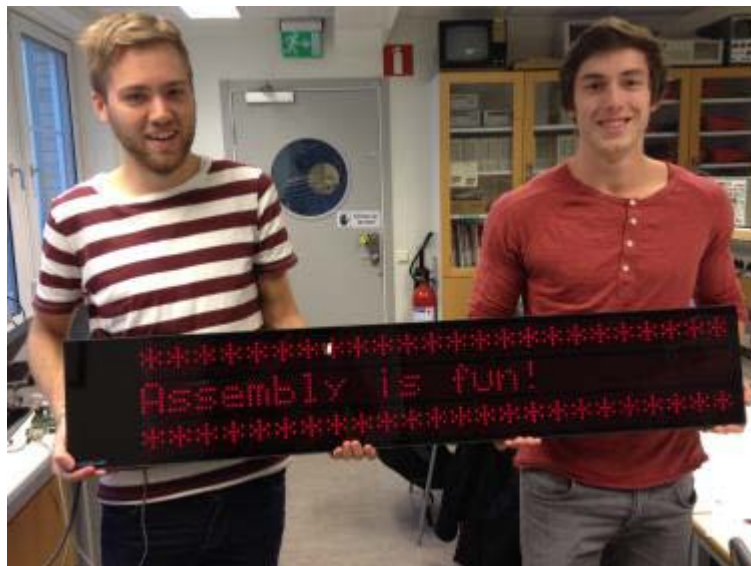
Function Prototype (older versions)

```
void add(int i1, int i2, int *p_i3);  
  
int main(void) {  
    int a = 23;  
    int b = 55;  
    int ret;  
    add(a, b, &ret);  
    return EXIT_SUCCESS;  
}
```

← Informs the compiler that add() is declared here but defined elsewhere.

← Makes the compiler succeed here.

If it isn't defined elsewhere the linker will fail.



Lab 6

Tasks

Task 1: Write a program that writes a character on the CyberTech Display.

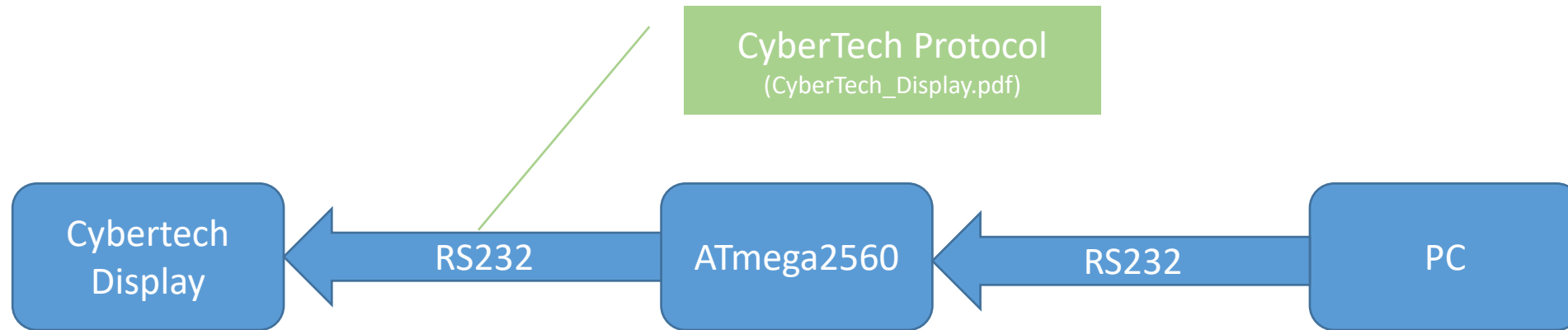
Task 2: Write a program that writes characters on all text-lines on the CyberTech Display.

Task 3: Write a program that change text strings on the display.

Task 4: Write a program that communicates with both the terminal and the display.

Task 5: Write a program for text input.

System Overview



Communications Protocol

A set of rules specifying how to send and receive commands/messages.

Information Message

Informatie frame

Walldisplay data wordt in frames verstuurd. Deze frames zijn altijd even lang of het nu een 1x8 of 2x24 walldisplay is. Het frame bestaat uit een START, ADRES, COMMANDO, CHECKSUM en STOP.

Voor het afbeelden van de informatie op een walldisplay is een extra AFBEELD frame nodig.

START	= 0x0D	ofwel Carriage return
ADRESS	= 'A'..'Y'	'A'=1 .. 'Y'=25
COMMANDO	= 'O0001'	Commando alleen anders bij Message software en AFBEELD commando.
CHECKSUM	= Sum(chars)	Alle bytes tot de checksum inclusief start opgeteld modulo 256 De checksum wordt in 2 karakters hexadecimaal 0..F verstuurd
EIND	= 0x0A	ofwel Linefeed

Een walldisplay frame wordt nu als volgt samengesteld:

[START][ADRESS][COMMANDO][48 karakters walldisplay tekst][CHECKSUM][EIND]

Image Message

Afbeeld frame:

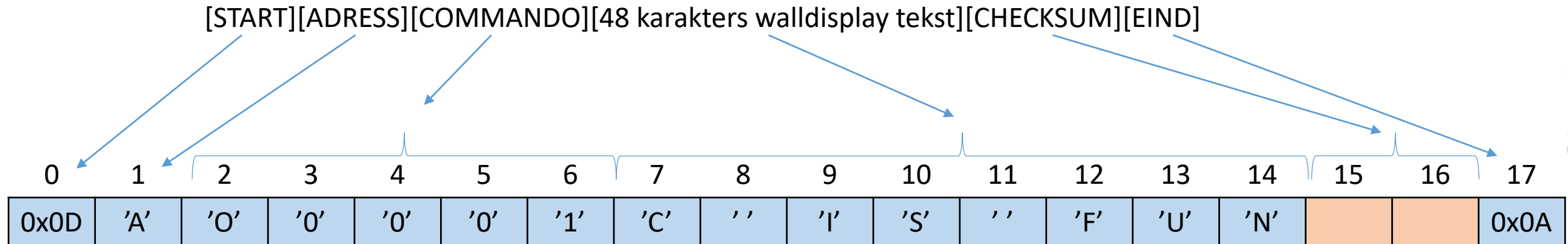
Een afbeeld frame wordt verstuurd na een informatie frame. Dit frame zorgt ervoor de walldisplay tekst uit het informatie frame op het walldisplay verschijnt. Voor dit commando wordt altijd het broadcast adres 'Z' gebruikt, verder heeft dit frame geen walldisplay karakters. Het frame is als volgt samengesteld:

[START][ADRESS][COMMANDO][CHECKSUM][EIND]

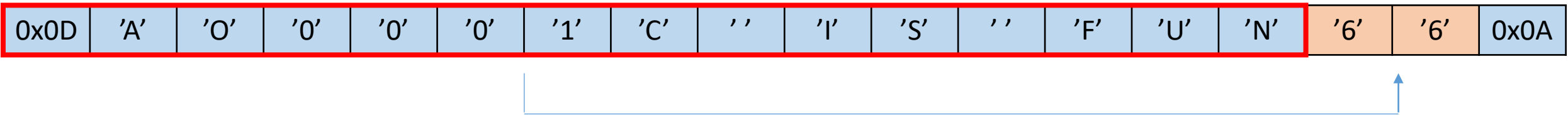
b.v. *[0x0D]['Z']['D001']['3C'][0x0A]*

Composing An Information Message - 1

Display the text *"C IS FUN"*.



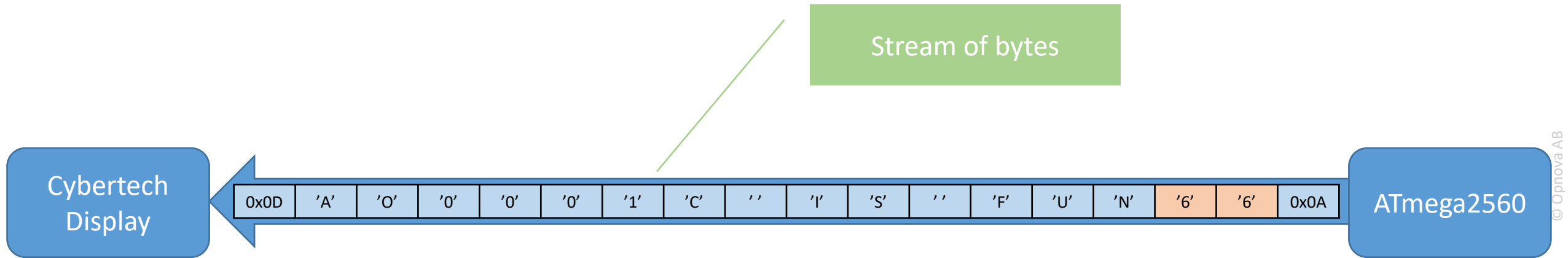
Composing An Information Message - 2



$(13+65+79+48+48+48+49+67+32+73+83+32+70+85+78) \text{ modulo } 256 = 102 \rightarrow 66_{16}$

Converting to Hex and as a string → Use `sprintf()`

Sending A Message



[START][ADRESS][COMMANDO][48 karakters walldisplay tekst][CHECKSUM][EIND]