- CO3009 -

- Efficient C Programming & Additional Exercise -

Problems with #define

```
#define PI_PLUS_1        3.14 + 1
.
.
x = 5 * PI_PLUS_1;   // The compiler sees the statement as
                     // x = 5 * 3.14 + 1
                     //So, it will be resolved as follows:
                     // x = (5 * 3.14) + 1
                     //Which is not what we want!
                     // Solution: #define PI_PLUS_1  (3.14 + 1)
                     //Moral: Beware of the "()" while dealing
                     //with the #define statement
```

anhpham@hcmut.edu.vn

Problem with Macros (1)

```
#define ADD(a,b) a + b
.
.
c = 2 + ADD(1,2);    // Result is 5 → Correct

d = 2 * ADD(1,2);    // Result is 4 → Incorrect
```

```
#define ADD(a,b) (a + b)
.
.
c = 2 * ADD(1,2);    // Result is 6 → Correct
```

Moral: *Again, beware of the "()" while dealing with the #define statement*

Problem with Macros (2)

```
#define MULT(a,b)  (a * b)
.
.
c = 3 + MULT(1,2);          // Result is 5 → Correct

d = 3 + MULT(1+1,2+2);      // Result is 8 → Incorrect
```

```
#define MULT(a,b)  ((a) * (b))
.
.
d = 3 + MULT(1+1,2+2);      // Result is 11 → Correct
```

Moral: I told you! Beware of the "()" while dealing with the #define statement

Playing around with Increment

- Example 1:

```
a = 2;
b = a++;
//Values after:      a = 3, while b = 2
```

- Example 2:

```
a = 2;
b = ++a;
//Values after:      a = 3, while b = 3
```

- Example 3:

```
a = 5;
b = 2;
c = a+++b;
//Values after:      a = 6, b = 2, while c = 7
```


Bit Manipulation (1)

- Detect if two integers have opposite signs:

```
int x, y;              // input values to compare signs bool
f = ((x ^ y) < 0);     // true iff x and y have opposite signs
```

- Determine if an unsigned integer is zero or a power of 2:

```
unsigned int v;      // we want to see if v is zero or a power of 2
bool f;              // the result goes here
f = (v & (v - 1)) == 0;
```

- Determine if an unsigned integer is a power of 2:

```
f = v && !(v & (v - 1));
```


Bit Manipulation (2)

- Merge bits from two values according to a mask:

```
unsigned int a;     // value to merge in non-masked bits unsigned
int b;              // value to merge in masked bits unsigned
int mask;       // 1 where bits from b should be selected; 0 where from a.
unsigned int r;     // result of (a & ~mask) | (b & mask) goes here
r = a ^ ((a ^ b) & mask);
```

- Counting bits set:

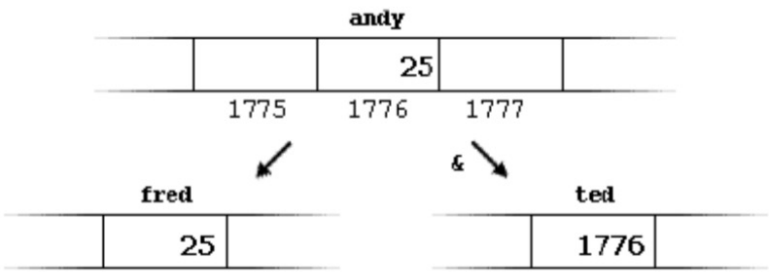
```
unsigned int v; // count the number of bits set in v
unsigned int c; // c accumulates the total bits set in v
for (c = 0; v; v >>= 1)
        c += v & 1;
```


Pointers

- Reference to a data object or a function
- Helpful for "call-by-reference" functions and dynamic data structures implementations
- Very often the only efficient way to manage large volumes of data is to manipulate not the data itself, but pointers to the data

Example:

```
andy = 25;
fred = andy;
ted = &andy;
```



anhpham@hcmut.edu.vn

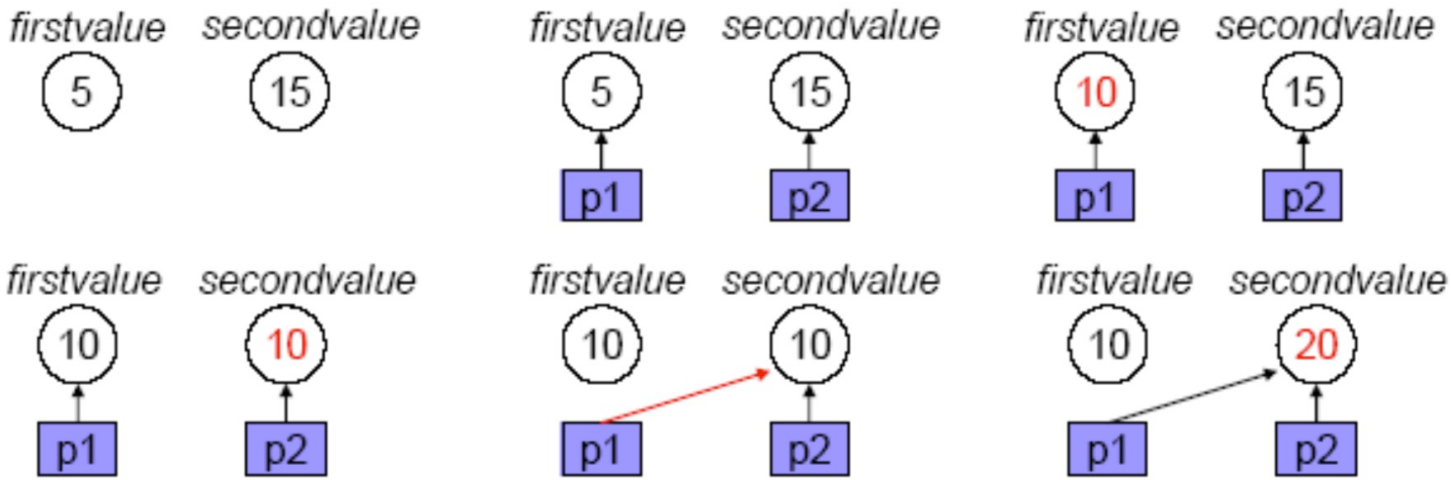
Pointers Example

```
int firstvalue = 5, secondvalue = 15;
int * p1, * p2;

p1 = &firstvalue;  // p1 = address of firstvalue
p2 = &secondvalue; // p2 = address of secondvalue
*p1 = 10;          // value pointed by p1 = 10
*p2 = *p1;         // value pointed by p2 = value pointed by p1
p1 = p2;           // p1 = p2 (value of pointer is copied)
*p1 = 20;          // value pointed by p1 = 20
```

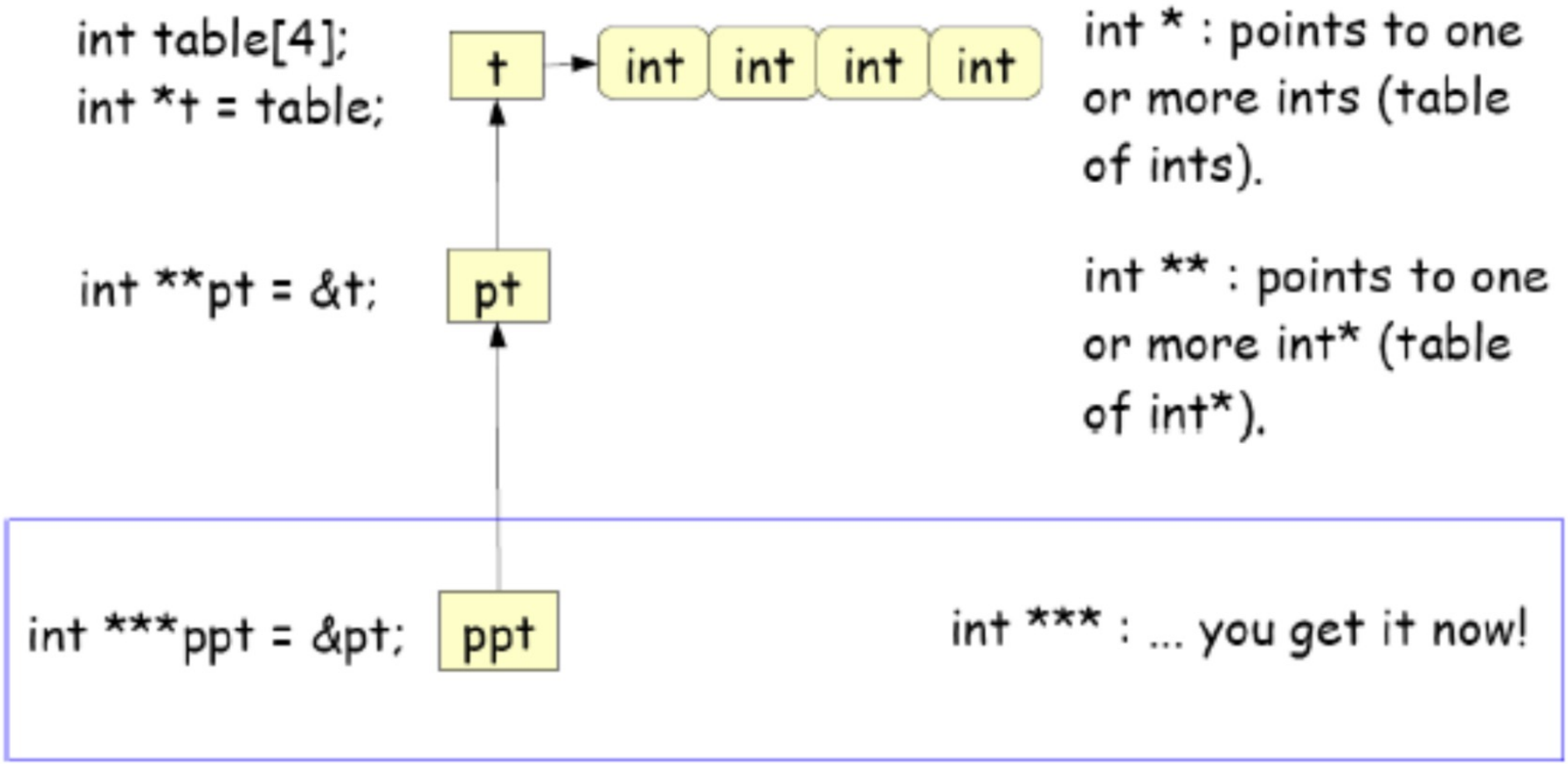
firstvalue = ?
secondvalue = ?

Pointers Example

```
int firstvalue = 5, secondvalue = 15;
int * p1, * p2;

p1 = &firstvalue;  // p1 = address of firstvalue
p2 = &secondvalue; // p2 = address of secondvalue
*p1 = 10;          // value pointed by p1 = 10
*p2 = *p1;         // value pointed by p2 = value pointed by p1
p1 = p2;           // p1 = p2 (value of pointer is copied)
*p1 = 20;          // value pointed by p1 = 20
```



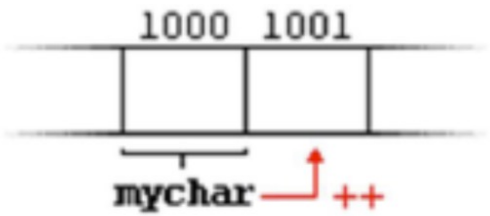
More Pointers Fun

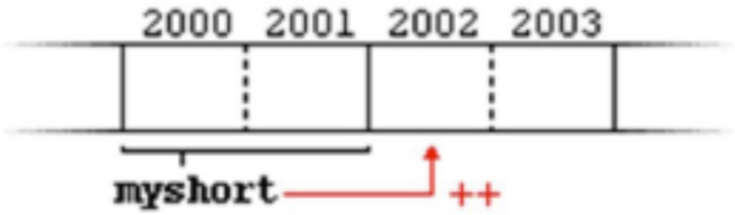


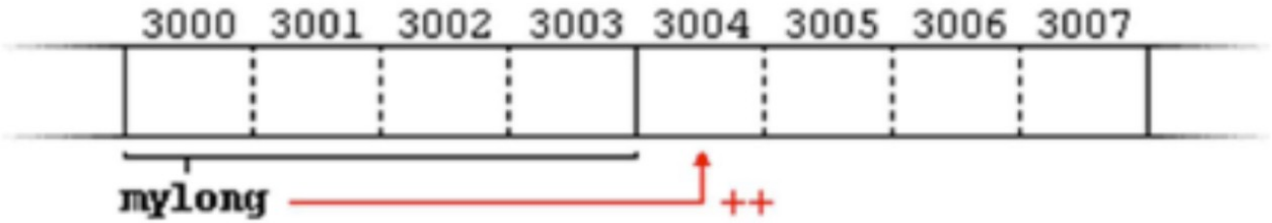
Pointers are Typed

```
char *mychar;
short *myshort;
long *mylong;
```

```
mychar++;
myshort++;
mylong++;
```





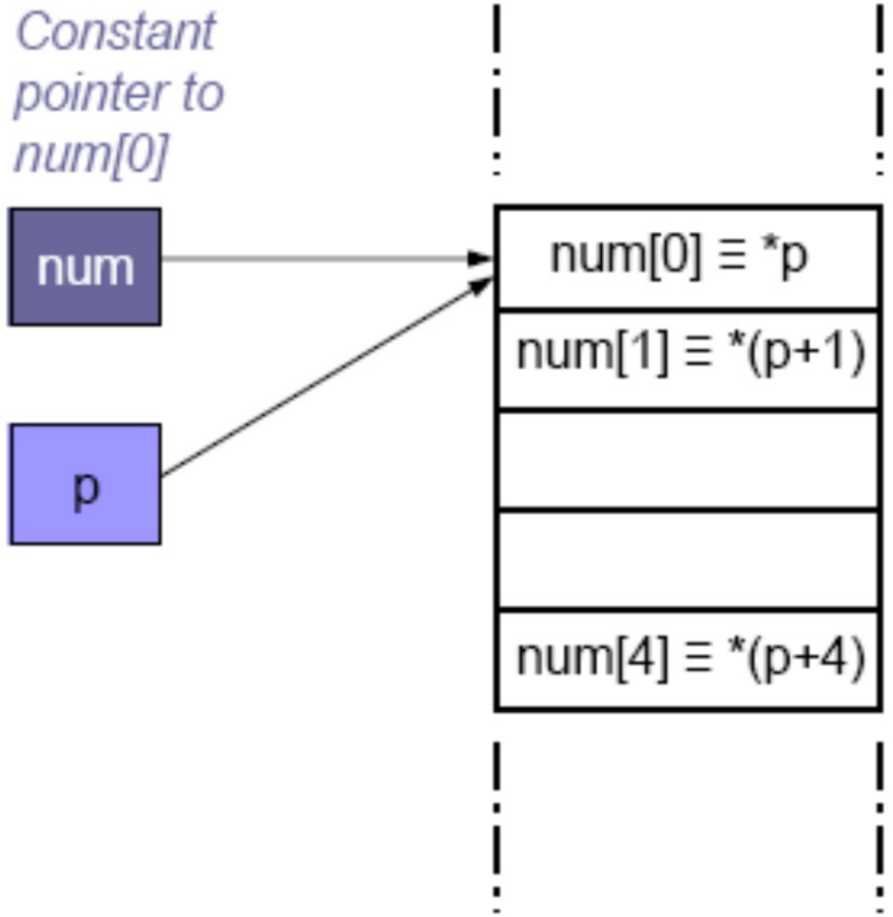


Pointers and Array

- *int num[5];*
- *int *p;*
- *p = num;*

~~*num = p;*~~

An array is a constant pointer



Pointers Precedence Issues

- *(array+i) ≡ array[i]
- *array+i ≡ array[0] + i
- *p++ ≡ *(p++)
- Notice the difference with (*p)++
- Better use parentheses to prevent mistakes
- int * ptr1, ptr2; Vs int * ptr1, * ptr2;

Efficient C Programming

- How to write C code in a style that will compile efficiently (**increased speed and reduced code size**) on ARM architecture?
  - How to use data types efficiently?
  - How to write loops efficiently?
  - How to allocate important variables to registers?
  - How to reduce the overhead of a function call?
  - How to pack data and access memory efficiently?

anhpham@hcmut.edu.vn

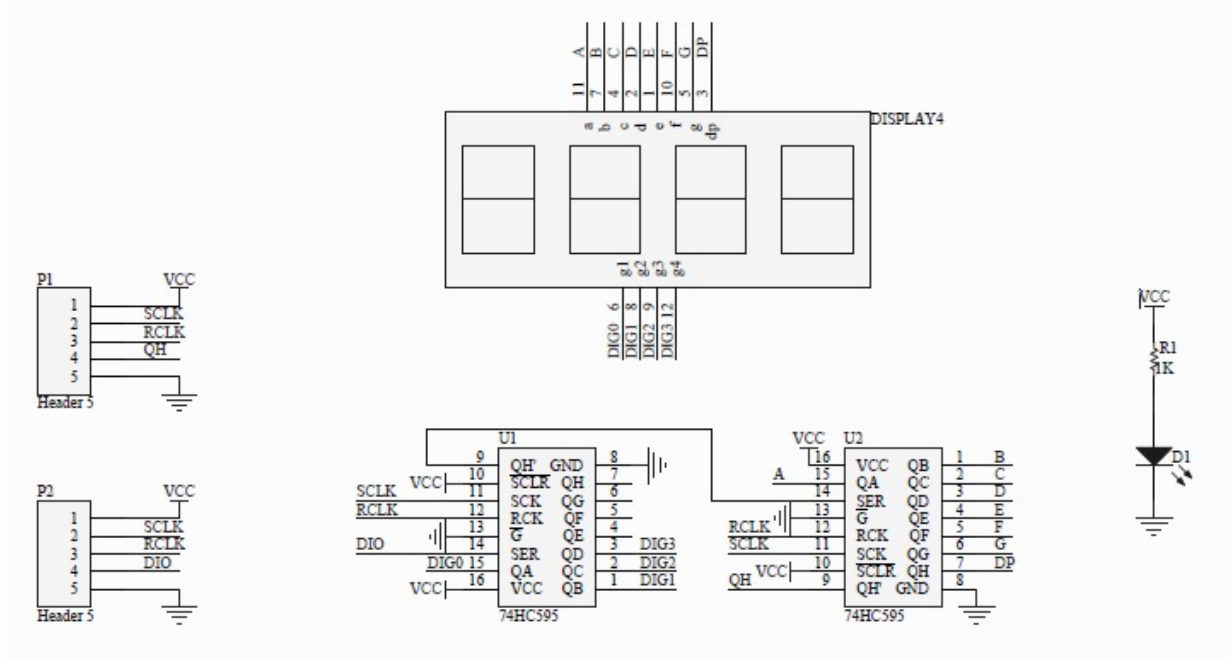
References

- A.N. Sloss, D. Symes, and C. Wright, “ARM System Developers Guide”

- CO3009 -

- Additional Exercises -

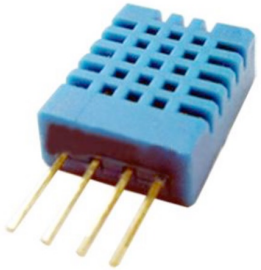
Four 7-Segment LEDs using 74HC595

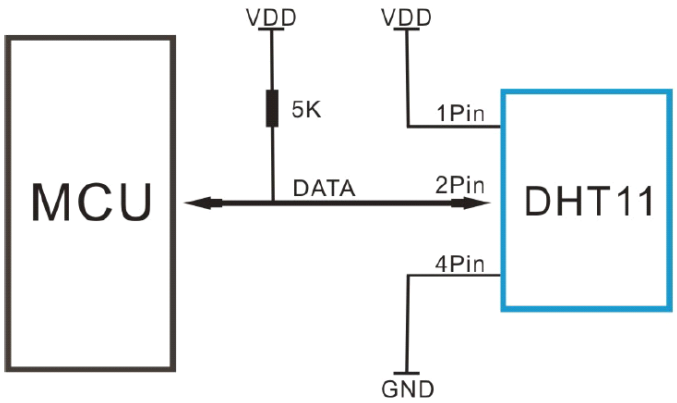


anhpham@hcmut.edu.vn

DHT11 Humidity & Temperature Sensor

- Your task is to read Humidity &Temperature data from DHT11
- Communication Process
  - Serial Interface (Single-Wire Two-Way)





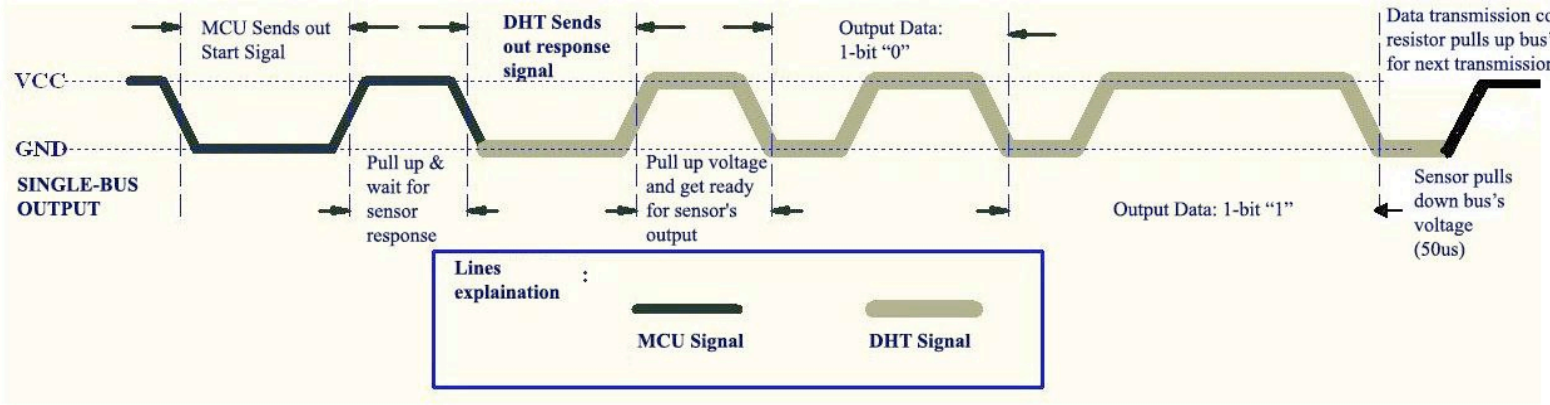
Single-Wire Two-Way (DHT11)

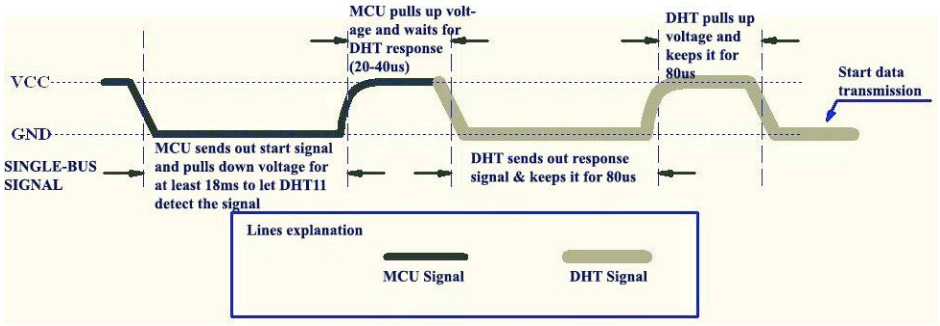
- Single-bus data format is used for communication and synchronization between MCU and DHT11 sensor.
- One communication process is about 4ms.
- Data consists of decimal and integral parts.
- A complete data transmission is **40bit**, and the sensor sends **higher data bit** first.
  - 8 bit integral humidity data
  - 8 bit decimal humidity data
  - 8 bit integral temperature data
  - 8 bit decimal temperature data
  - 8 bit check sum.
    - If the data transmission is right, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data".

Overall Communication Process

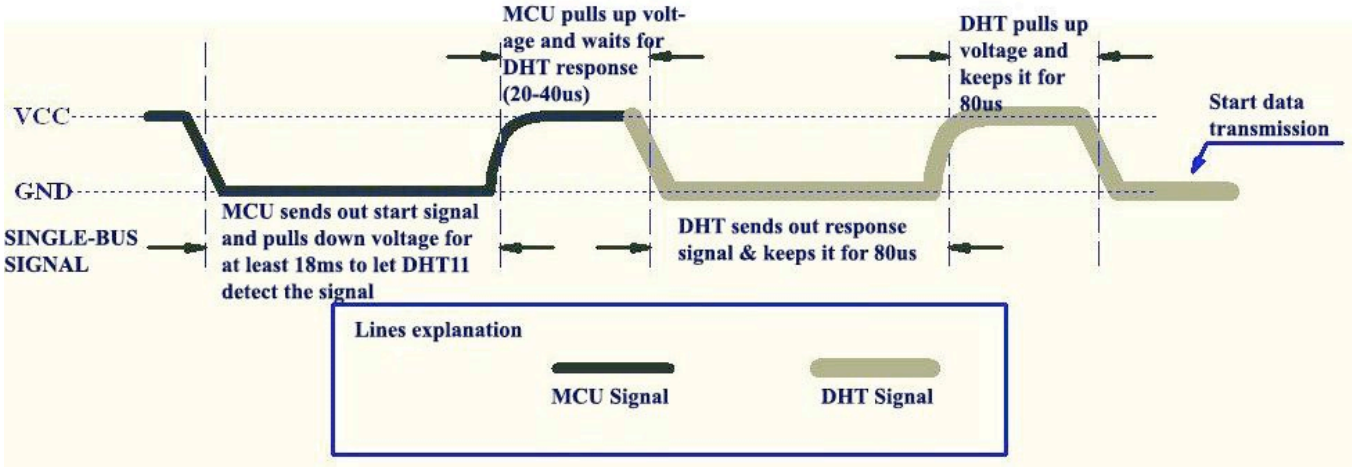
- When MCU sends a start signal, DHT11 changes from the low-power-consumption mode to the running-mode, waiting for MCU completing the start signal.
- Once it is completed, DHT11 sends a response signal of **40-bit data** that include the relative humidity and temperature information to MCU.
- Users can choose to collect (read) some data.
- Without the start signal from MCU, DHT11 will not give the response signal to MCU.
- Once data is collected, DHT11 will change to the low-power-consumption mode until it receives a start signal from MCU again.

MCU Sends out Start Signal to DHT

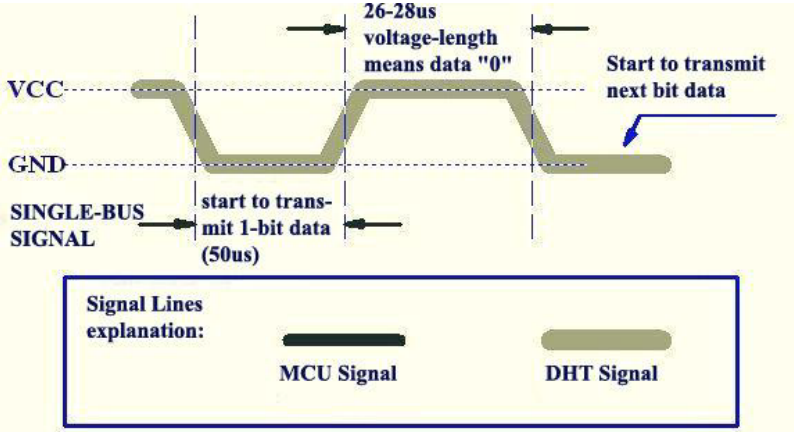




MCU Sends out Start Signal to DHT



DHT Responses to MCU



0 indication

1 indication