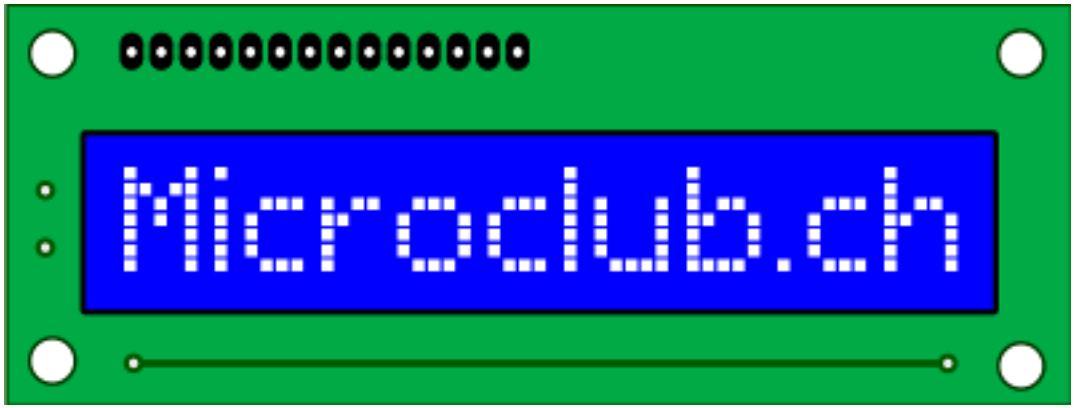

Finesses et astuCes

Microclub 22.02.2019

L. Francey

Finesses et astuCes

Un fichier écrit en langage C doit être compilé pour être traduit en un langage compréhensible par le processeur pour lequel il a été écrit.

Il est très intéressant de voir comment certaines parties du codes sont traduites. Certains compilateurs sont très ... très futés !

Voici ci-après quelques exemples illustrés et compilés avec AtmelStudio et le compilateur GCC for AVR.

AstuCes : boucles

```
#include <avr/io.h>
int main(void)
{
    uint8_t local_1 = 0;
    do {
        PORTB ^= local_1;
        local_1++;
    } while (local_1<100);
}
```

Code : 80 bytes

```
#include <avr/io.h>
int main(void)
{
    uint8_t local_1 = 100;
    do {
        PORTB ^= local_1;
        local_1--;
    } while (local_1>0);
}
```

Code : 78 bytes

AstuCes : boucles imbriquées

```
int main(void)
{
    uint8_t i, total = 0;
    uint8_t tmp[10] = {0};
    for (i=0; i<10; i++) {
        tmp [i] = ADCH;
    }
    for (i=0; i<10; i++) {
        total += tmp[i];
    }
    UDR = total;
}
```

Code : 148 bytes

```
int main(void)
{
    uint8_t i, total = 0;
    uint8_t tmp[10] = {0};
    for (i=0; i<10; i++) {
        tmp [i] = ADCH;
        total += tmp[i];
    }
    UDR = total;
}
```

Code : 80 bytes

AstuCes : if ... else ou switch case

```
tmp = ADCH;

    if (tmp == 10)
    {
        UDR = tmp;
        UDR = 10;
    }

    if (tmp == 20)
    {
        UDR = tmp;
        UDR = 20;
    }
```

Code : 86 bytes

```
tmp = ADCH;
switch (tmp)
{
    case 10 :
        {
            UDR = tmp;
            UDR = 10;
        }; break;
    case 20 :
        {
            UDR = tmp;
            UDR = 20;
        }; break;
}
```

Code : 92 bytes

AstuCes : variables 16 ou 8 bits

```
#include <avr/io.h>

int main(void)
{
    uint16_t local_1 = 10;
    do {
        PORTB ^= 0x80;
    } while (--local_1);
}
```

Code : 82 bytes

```
#include <avr/io.h>

int main(void)
{
    uint8_t local_1 = 10;
    do {
        PORTB ^= 0x80;
    } while (--local_1);
}
```

Code : 78 bytes

AstuCes : variables

```
#include <avr/io.h>

uint8_t global_1;

int main(void)
{
    global_1 = 0xAA;
    PORTB = global_1;
}
```

Code : 80 bytes
Data : 1 byte

```
#include <avr/io.h>

int main(void)
{
    uint8_t local_1;

    local_1 = 0xAA;
    PORTB = local_1;
}
```

Code : 70 bytes
Data : 0 byte

AstuCes : constantes

```
#include <avr/io.h>

uint8_t string[12] = {"hello
world!"};

int main(void)
{
    UDR = string[10];
}
```

Code : 106 bytes
Data : 12 byte

```
#include <avr/io.h>
#include <avr/pgmspace.h>
const uint8_t string[12]
PROGMEM = {"hello
world!"};

int main(void)
{
    UDR =
pgm_read_byte(&string[10]);
}
```

Code : 86 bytes
Data : 0 byte

AstuCes : static function

```
void USART_TX(uint8_t data);

int main(void)
{
	uint8_t i = 0;
	while (i<12) {
		USART_TX(string[i++]);
	}
}

void USART_TX(uint8_t data)
{
	while(!(UCSRA&(1<<UDRE)));
	UDR = data;
}
```

Code : 142 bytes
Data : 12 bytes

```
static void USART_TX(uint8_t data);
int main(void)
{
	uint8_t i = 0;
	while (i<12) {
		USART_TX(string[i++]);
	}
}

void USART_TX(uint8_t data)
{
	while(!(UCSRA&(1<<UDRE)));
	UDR = data;
}
```

Code : 122 bytes
Data : 12 bytes

AstuCes : post or pré-incrément

```
#include <avr/io.h>
int main(void)
{
	uint8_t loop_cnt = 9;
	do {
		if (loop_cnt--) {
			PORTC ^= 0x01;
			} else {
			PORTB ^= 0x01;
			loop_cnt = 9;
		}
	} while (1);
}
```

Code : 88 bytes
Data : 0 bytes
Cycles : 75

```
#include <avr/io.h>
int main(void)
{
	uint8_t loop_cnt = 10;
	do {
		if (--loop_cnt) {
			PORTC ^= 0x01;
			} else {
			PORTB ^= 0x01;
			loop_cnt = 10;
		}
	} while (1);
}
```

Code : 86 bytes
Data : 0 bytes
Cycles : 61

AstuCes : boucle déroulée

```
#include <avr/io.h>
int main(void)
{
	uint8_t loop_cnt = 10;
	do {
		PORTB ^= 0x01;
	} while (--loop_cnt);
}
```

Code : 80 bytes
Data : 0 bytes
Cycles : 80

```
#include <avr/io.h>
int main(void)
{
	PORTB ^= 0x01;
	PORTB ^= 0x01;
	PORTB ^= 0x01;
	PORTB ^= 0x01;
	PORTB ^= 0x01;
	PORTB ^= 0x01;
	PORTB ^= 0x01;
	PORTB ^= 0x01;
	PORTB ^= 0x01;
	PORTB ^= 0x01;
}
```

Code : 128 bytes
Data : 0 bytes
Cycles : 61

AstuCes : if else ...

```
int main(void)
{
	uint8_t output;
	ad_result = readADC();
	if(ad_result <= 30)
		output = 0x6C;
	else if(ad_result <= 60)
		output = 0x6E;
	else if(ad_result <= 90)
		output = 0x68;
	else if(ad_result <= 120)
		output = 0x4C;
	else if(ad_result <= 150)
		output = 0x4E;
	else if(ad_result <= 180)
		output = 0x48;
	else if(ad_result <= 210)
		output = 0x57;
	else if(ad_result <= 240)
		output = 0x45;
	send(output);
}
```

Code : 152 bytes
Cycles : 58 (pire des cas)

```
if (ad_result <= 120){
		if (ad_result <= 60){
			if (ad_result <= 30)
				output = 0x6C;
			else output = 0x6E;
		}
		else{
			if (ad_result <= 90)
				output = 0x68;
			Else output = 0x4C;
			}
		}
	}
	else{
			.....
		}
	}
```

Code : 226 bytes
Cycles : 48 pire des cas

Finesses et astuCes

Conclusion

Lorsque vous devez optimiser un code pour la rapidité ou pour la place mémoire, pensez aussi à optimiser votre code et non seulement les options du compilateur !