# 0b00100100

Покрутил тут ардуино,точнее прикручивал к нему ЖКИ, для него хоть и есть готовая библиотека, но хотелось разобраться с даташитом и сделать по описанию (задача удалась процентов на 90, т.к. подглядывать в исходники библиотеки пришлось, но, как в итоге оказалось, из-за небольшой ошибки в схемотехнике,так бы моё работать начало).

Но суть не в том, там команды формируются битами и битами же в байте задаются её параметры. По классике жанра делаются примерно такие дефайны в шестнадцатеричном виде:

```
#define BIT0 0x01
#define BIT1 0x02
#define BIT2 0x04
#define BIT3 0x08
#define BIT4 0x10
#define BIT5 0x20
#define BIT6 0x40
#define BIT7 0x80
```

или в восьмеричном:

```
#define BIT0 0001
#define BIT1 0002
#define BIT2 0004
#define BIT3 0010
#define BIT4 0020
#define BIT5 0040
#define BIT6 0100
#define BIT7 0200
```

тут нетрудно проследить закономерности, разделив биты в группы по 4 (в hex представлении) или по 3 (начиная с нулевого, в oct представлении).

Но голова не резиновая, решил поискать, а не придумали что в C/C++ для непосредственного задания констант в битовом представлении.

Оказалось - придумали, по крайней мере в компиляторах GCC и GCC-AVR можно задавать непосредственно битовое число, используя префикс `0b`:

```
#define BIT0 0b00000001
#define BIT1 0b00000010
#define BIT2 0b00000100
#define BIT3 0b00001000
#define BIT4 0b00010000
#define BIT5 0b00100000
#define BIT6 0b01000000
#define BIT7 0b10000000
```

или (кому как более наглядно):

```
#define BIT0 0b1
#define BIT1 0b10
#define BIT2 0b100
#define BIT3 0b1000
#define BIT4 0b10000
#define BIT5 0b100000
#define BIT6 0b1000000
#define BIT7 0b10000000
```

Не ручаюсь абсолютно на совместимость с другими компиляторами (да и нет её скорее всего), но интересно было бы узнать как там обстоят дела с подобной bin-нотацией констант.

PS да, объединять потом в нужное число или исключать нужные биты:

```
unsigned char c1 = BIT0 | BIT4 | BIT7; /* включить биты */
unsigned char c2 = c1 ^ BIT4; /*выключить биты */
if(c1 & BIT4) {} /* проверить установку бита */
```

**UPD**:
Вот выдержка из документа по GCC, страница 556:

> **6.56 Binary constants using the ‘0b’ prefix**
> nteger constants can be written as binary constants, consisting of a sequence of ‘0’ and ‘1’ digits, prefixed by ‘0b’ or ‘0B’. This is particularly useful in environments that operate a lot on the bit-level (like microcontrollers).

Секция 6 этого документа зовется «Extensions to the C Language Family», в начале этой секции говорится:

> GNU C provides several language features not found in ISO standard C. (The ‘-pedantic’ option directs GCC to print a warning message if any of these features is used.) To test for the availability of these features in conditional compilation, check for a predefined macro GNUC, which is always defined under GCC.
> These extensions are available in C and Objective-C. Most of them are also available in C++. See Chapter 7 [Extensions to the C++ Language], page 547, for extensions that apply only to C++.
> Some features that are in ISO C99 but not C89 or C++ are also, as extensions, accepted by GCC in C89 mode and in C++.

сиречь, если захотите публиковать программу, скомпилируйте её с параметром `-pedantic`, проверьте все предупреждения и… или давайте рекомендации или используйте препроцессорные директивы.