Описание битового флага у карточки складского учета.

/* bits for SMCard.Flags */
#define SMCARDFLAG_WHRQPACKS SMSQLLONG(1) /*if set then automatic store orders generation will round up quantities to pack sizes*/
#define SMCARDFLAG_CASHNOSCALE SMSQLLONG(2) /*if set then scale info is not loaded to cash desk, flag ignored if article has no scale attached*/
#define SMCARDFLAG_PRICERSFORTENTH SMSQLLONG(4) /* цены на ценнике за 0.1 единицы измерения*/
//#define флаг свободен SMSQLLONG(8) /* ранее обозначал "Спец. товар", убран в версии 1.027.5*/
#define SMCARDFLAG_IGNORE_RATE SMSQLLONG(16) /* Скорость продаж игнорируется при генерации заказов (т.е. как если бы она была равна 0)*/
#define SMCARDFLAG_FIXED_PRICE SMSQLLONG(32) /* Фиксированная цена на артикул. Артикул не участвует в ценообразовании.*/
#define SMCARDFLAG_NONINGREDIENT SMSQLLONG(64) /* запрещен к использованию в производстве*/
#define SMCARDFLAG_ONEXCISE SMSQLLONG(128) /* акцизный товар*/
#define SMCARDFLAG_CERTREQUIRED SMSQLLONG(256) /* требуется сертификат соответствия (в документе прихода)*/
#define SMCARDFLAG_INGREDIENT SMSQLLONG(512) /* артикул может использоваться как ингредиент*/
#define SMCARDFLAG_GIFT SMSQLLONG(1024) /* разрешена безвозмездная передача: артикулы с таким флагом могут грузиться в кассу даже с нулевой ценой*/
#define SMCARDFLAG_FOOD SMSQLLONG(2048) /*продовольственный товар*/
#define SMCARDFLAG_MARK_EAC SMSQLLONG(4096) /*Маркировка товара EAC*/
#define SMCARDFLAG_MARK_CTM SMSQLLONG(8192) /*Маркировка товара CTM*/
#define SMCARDFLAG_MARK_KVI SMSQLLONG(16384)/*Маркировка товара KVI*/
#define SMCARDFLAG_MAX SMSQLLONG(16384)/*максимальный текущий флаг артикула*/
// next is 4096 etc...

Битовый флаг

Замечали ли вы, что в функциях, например FileOpen, WinGetState, стили в GUICreate используется число определяющее некоторую опцию или включающую некоторый стиль? И эти числа имеют закономерность они как бы образованные многократным умножением на 2. Чтобы понять, почему используется именно такой способ, а не 1, 2, 3, 4, 5 и т.д. попробуем разобраться.

При использовании в качестве параметра функции 1, 2, 3, 4, 5 и т.д. можно задать выбор только одной опции из всех доступных и использовать переключатель Switch для определения флага. Битовый флаг, состоящий из такой последовательности доступных чисел 1, 2, 4, 8, 16, 32 и т.д. позволяет хранить в одном параметре несколько опций, которые имеют состояние включено или выключено. Например максимальное беззнаковое целое число 0xFFFFFFFF может содержать 32 битовых флага. Число в двоичной системе состоит из 0 и 1, соответственно каждый бит можно рассматривать как флаг включено или выключено, а позицию бита рассматривать как одну из опций. В данном случае смущает только десятичный вид представления флагов, которые кажутся не очевидно понятными, но в двоичной системе это в чистом виде набор флагов.

Битовый флаг

Dec	Bin	Степень
1	00000001	2 ^ 0
2	00000010	2 ^ 1
4	00000100	2 ^ 2
8	00001000	2 ^ 3
16	00010000	2 ^ 4
32	00100000	2 ^ 5

64	01000000	2 ^ 6
128	10000000	2 ^ 7

Как видно из выше указанной таблицы, степень числа определяет позицию флага и очевидно, что 2 ^ 31 = 2 147 483 648 является последним доступным флагом.

Комбинация нескольких битовых флагов

Dec	Флаги	Bin
12	8 + 4	00001100
44	32 + 8 + 4	00101100

Из таблицы видно, что добавление битового флага просто переключает 0 на 1 в соответствующей позиции. Для работы с флагами используются функции BitAND и BitOR.

С помощью функции BitAND проверяется наличие флага, его включенное состояние.

```
$iNum = 44
If BitAND($iNum, 8) Then
   MsgBox(0, 'Сообщение', 'Да, этот флаг включён')
Else
   MsgBox(0, 'Сообщение', 'Нет, этот флаг выключен')
EndIf
```

С помощью BitOR можно безопасно объединять флаги

```
$iNum = 12 ; 8 + 4
$iNum = BitOR($iNum, 4) ; Добавляет флаг 4
MsgBox(0, 'Сообщение', $iNum) ; Число 12 не изменилось, так как флаг был включён прежде
```

То есть допустимо суммировать битовые флаги без использования BitOR, например 8 + 4 + 2, но иногда при использовании констант, таких как стиль для GUICreate можно совершить ошибку. Некоторые константы уже содержат комбинацию битовых флагов и при добавлении флага, который уже присутствует в константе можно получить совершенно другую комбинацию битовых флагов. Например если попытаться к 14 (8 + 4 + 2) добавить флаг 4, то получится число 18 (16 + 2), а это уже совсем другая комбинация флагов. Совершенно очевидно, что любое десятичное число при разложении на битовые флаги однозначно даст только одну комбинацию флагов, так же как и любая комбинация флагов даёт только одно уникальное десятичное число, потому что это по сути это одно и тоже число только в десятичном или в двоичном представлении.

Использование битового флага

Пример функции с битовым флагом.

```
MsgBox(0, 'Сообщение', _Check(32 + 8 + 4))

Func _Check($iNum)
   Local $sText
   If BitAND($iNum, 1) Then $sText &= '1' & @LF
   If BitAND($iNum, 2) Then $sText &= '2' & @LF
   If BitAND($iNum, 4) Then $sText &= '4' & @LF
   If BitAND($iNum, 8) Then $sText &= '8' & @LF
   If BitAND($iNum, 16) Then $sText &= '16' & @LF
   If BitAND($iNum, 32) Then $sText &= '32' & @LF
   If BitAND($iNum, 64) Then $sText &= '64' & @LF
   If BitAND($iNum, 128) Then $sText &= '128' & @LF
   Return $sText
EndFunc   ;==>_Check
```

Способ исключения флага

```
$iNum = 12
If BitAND($iNum, 4) Then $iNum -= 4
MsgBox(0, 'Сообщение', $iNum)

$iNum = 12
$iNum = BitNOT(BitOR(BitNOT($iNum), 4))
MsgBox(0, 'Сообщение', $iNum)
```