

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Белгородский государственный технологический университет
им. В. Г. Шухова»
Институт информационных технологий и управляющих систем
Кафедра технической кибернетики

Методические указания

Создание динамической библиотеки блока FNC среды GoodHelp.

Разработал:
Студент гр. УС-41
Черных А.А
Преподаватель:
Кижук А.С.

Белгород 2014 г.

Блок «С функция» (FNC), входящая в стандартную библиотеку алгоблоков в раздел «Обработка данных» представляет собой интерфейс для связи программного кода на языке C/C++ и средой Good Help. Программист может написать собственный алгоритм на языке «С» и подключить его к системе . На вход блока передается массив из пяти входных аналоговых параметров : Inp1 .. Inp5, сигнал разрешения (Enable) и сигнал сброса (Reset). Выходом блока служит массив из пяти аналоговых параметров : Out1 .. Out5.

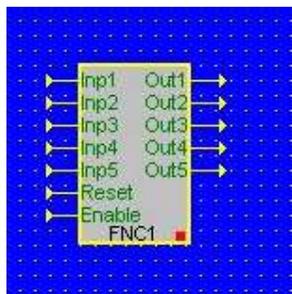


Рис 1. Блок «С функция»

В свойствах блока имеются флажок разрешения, который связан со входом разрешения, установка начальных значений для пяти аналоговых входов, период, обозначающий временной интервал через который блок будет срабатывать и имя функции.

The screenshot shows a dialog box titled 'Редактирование' (Editing) with a light green header. Inside, there is a checkbox labeled 'Разрешение' (Enable) which is currently unchecked. To its right is a text input field labeled 'Период' (Period) containing the number '0'. Below this is a section titled 'Начальные значения' (Initial values) containing five rows, each with a label 'Вход 1' through 'Вход 5' and a corresponding text input field, all containing the number '0'. At the bottom, there is a text input field labeled 'Имя функции' (Function name) which is empty. Below the input fields are two buttons: 'Да' (Yes) and 'Нет' (No).

Рис 2. Свойства блока FNC

Для взаимодействия пользовательского кода и среды GoodHelp код должен быть правильно оформлен и скомпилирован в динамическую библиотеку непосредственно, или в *.bin файл посредством утилиты поставляемой вместе со средой GoodHelp. Рассмотрим код примера на языке C++ :

```
#pragma inline
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include <time.h>
#ifdef __cplusplus // связывание имен в стиле C.
extern "C" {
#endif
// Внимание
// Запрещена запись в глобальные переменные
// только чтение.
char Ts[]{"TZ=EST0"};
char hex_to_ascii[]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
void WINAPI _export Primer(float Inp[5], //Входной массив
                           float Out[5], //выходной массив
                           float Data[10], // Массив промежуточных значений
                           int Init, //Инициализация блока
                           int Reset, //Сброс
                           struct tm far *GTime1,
                           float far *Ret[]
                           )
{
// Локальные переменные
time_t dummy;
struct tm *GTime; //Установка локального времени

if(Init)
{
//Если происходит инициализация блока
Out[0]=Out[1]=Out[2]=Out[3]=Out[4]=0;
putenv(Ts); //Установить переменные среды
tzset(); // инициализирует информацию о преобразованиях времени
return;
};
if(Reset)
{
//Если происходит сброс
Out[0]=Out[1]=Out[2]=Out[3]=Out[4]=0;
}
time(&dummy);
GTime = localtime(&dummy);
{...} //Тело функции.
return;
}
#ifdef __cplusplus
}
#endif
```

Рассмотрим код программы более подробно:

```

void WINAPI _export Primer(float Inp[5], //Входной массив
float Out[5], //выходной массив
float Data[10], // Массив промежуточных значений
int Init, //Инициализация блока
int Reset, //Сброс
struct tm far *GTime1,
float far *Ret[]
)

```

Как видно процедура void WINAPI _export Primer имеет 7 параметров:

- float Inp[5] - Входной массив, в блоке FNC ему соответствуют входы Inp1 .. Inp5.
- float Out[5] – Выходной массив и соответственно выходы Out1 .. Out5
- float Data[10] – массив из десяти промежуточных переменных. Программист может использовать их по своему усмотрению.
- int Init – Входной сигнал инициализации. Устанавливается в «1» при первом запуске.
- int Reset – Сигнал сброса. Соответствует входу Reset
- struct tm far *GTime1 – Структура для работы с локальным временем контроллера
- float far *Ret[] – Адрес возврата. Если программа желает выдать сообщение на терминал, она должна занести в Ret[0] номер сообщения из журнала, иначе она должна занести туда 0.

Программист может производить обработку пяти входных чисел (Inp[0].. Inp[5]) по своему усмотрению, в зависимости от решаемых им задач. Полученные результаты следует занести в выходной массив Out, что приведет к появлению их на соответствующих выходах блока FNC.

```

if(Init)
{
//Если происходит инициализация блока
Out[0]=Out[1]=Out[2]=Out[3]=Out[4]=0;
putenv(Ts); //Установить переменные среды
tzset(); // инициализирует информацию о преобразованиях времени
return;
};
if(Reset)
{
//Если происходит сброс
Out[0]=Out[1]=Out[2]=Out[3]=Out[4]=0;
}

```

В этом участке кода каждый программный цикл (что значит каждый раз когда выполняется функция) проверяется значение переменных `init` и `Reset`. `Init` не равен нулю только при первом выполнении функции, а следовательно действия расположенные внутри блока условия выполняются один раз при первом программном цикле. Переменная `Reset` считывается с входа блока FNC и если её значение не равно нулю производятся действия расположенные в блоке условия, а именно обнуление всех выводов блока FNC. После чего функция продолжает выполняться.

Пример: Получить на нулевом выводе блока сумму всех его входных величин, на первом выводе произведение этих величин, на втором наибольшее из этих чисел и на третьем наименьшее.

Решение:

```
#pragma inline
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include <time.h>
#ifdef __cplusplus // связывание имен в стиле C.
extern "C" {
#endif
// Внимание
// Запрещена запись в глобальные переменные
// только чтение.
char Ts[]={"TZ=EST0"};
char hex_to_ascii[]={ '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
float max_inp(float Arr[5]){ //Функция для нахождения наибольшего значения в массиве из пяти
элементов
    float buffer=Arr[0];
    int i;
    for(i=1;i<5;i++){ if(buffer<Arr[i]){buffer=Arr[i];} }
    return buffer;
}
float min_inp(float Arr[5]){ //Функция для нахождения наименьшего значения в массиве из пяти
элементов
    float buffer=Arr[0];
    int i;
    for(i=1;i<5;i++){ if(buffer>Arr[i]){buffer=Arr[i];} }
    return buffer;
}
//функция с именем Test для решения поставленной задачи.
void WINAPI _export Test(float Inp[5], //Входной массив
    float Out[5], //выходной массив
    float Data[10], // Массив промежуточных значений
    int Init, //Инициализация блока
    int Reset, //Сброс
    struct tm far *GTime1,
    float far *Ret[]
    ){
    if(Init) {
        //Если происходит инициализация блока
```

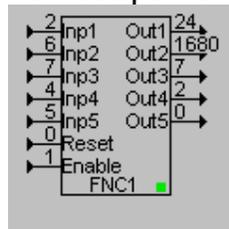
```

Out[0]=Out[1]=Out[2]=Out[3]=Out[4]=0;
putenv(Ts); //Установить переменные среды
tzset(); // инициализирует информацию о преобразованиях времени
return;
};
if(Reset){
//Если происходит сброс
Out[0]=Out[1]=Out[2]=Out[3]=Out[4]=0;
}
Out[0]=Inp[0]+Inp[1]+Inp[2]+Inp[3]+Inp[4]; //На нулевой вывод выводим сумму пяти входов
Out[1]=Inp[0]*Inp[1]*Inp[2]*Inp[3]*Inp[4]; //На первый вывод выводим произведение пяти входов
Out[2]=max_inp(Inp); //На второй вывод максимальное из пяти входных значений
Out[3]=min_inp(Inp); //на третий вывод минимальное из пяти входных значений
return; //Выход из функции
}

#ifdef __cplusplus
}
#endif

```

Результат работы представлен на изображении



Такой способ подходит, когда нужно обрабатывать только текущие значения входов. Если же требуется использовать предыдущие значения (например, для дифференцирования или интегрирования) то сохранять эти значения следует в массиве Data. Значения в этом массиве сохраняются на протяжении работы программы и не обнуляются в каждом программном цикле, как в случае с локальными переменными.

В одной библиотеке fnc.dll может быть несколько функций. Оформляются они одинаково. Блок FNC Будет выполнять ту функцию, имя которой записано в поле «Имя функции».

Компиляцию программы следует производить в модели памяти ti-ny (максимальный размер 64К).

Для компиляции используется Borland C Compiler (BCC) Версии 5.5.

Компиляция производится следующей командой:

```
"путь_до_папки_bin\bcc32.exe" -tWD -v -efnc.dll Имя_исходного_кода.cpp
```

Для компиляции DLL в строке ключей (параметров) компилятора используется ключ -tWD. При необходимости для указания пути к папкам, содержащим заголовочные и библиотечные файлы используются следующие ключи

```
- IC:\Borland\BCC55\INCLUDE
```

`-LC:\Borland\BCC55\Lib`

В результате выполнения данной команды будет получен файл fnc.dll , который следует переписать в каталог edchart/disk7188.

Скачать компилятор BCC, Среду GoodHelp и данные методические рекомендации можно на сайте кафедры Технической кибернетики по адресу <http://tk.bstu.ru/students/materials> в разделе Проектирование систем управления