



Страница еще не переведена полностью.

Управление шпинделем по каналу Pulse-Dir

Независимый генератор импульсов, реализованный в myCNC, может быть замешан в канал оси "B".

Управление независимым генератором осуществляется через запись значений в глобальные переменные 8130-8133. Назначение регистров независимого генератора описано в таблице ниже.

Имя регистра	Адрес	Назначение
GVAR_GENERATOR_FRQ_RAW	8130	"Сырое значение" частоты задания независимого генератора импульсов, [ед.] 1 ед. = 0.0014549 Герц
GVAR_GENERATOR_ACCEL	8131	Ускорение (скорость изменения частоты) независимого генератора, [ед.] 1 ед. = 1.4549 1/с ²
GVAR_GENERATOR_FRQ_RATIO	8132	Множитель частоты независимого генератора
GVAR_GENERATOR_FRQ	8133	Заданная частота генератора. Величина (Частота*Множитель) посылается в регистр генератора и сохраняется как "сырое" значение.

При доступе из элементов GUI (кнопки, строки ввода и тд) удобно (и необходимо) использовать регистры множителя и частоты при задании частоты генератора. При изменении значения регистра 8133 (Заданная частота генератора) ПО myCNC автоматически пересчитает значение заданной частоты с учетом заданного множителя и прошлет эти данные в контроллер.

При доступе из Hardware PLC необходимо использовать запись в регистр "сырого" значения (8130) и самостоятельно (в коде PLC) учитывать множитель.

Изначально независимый генератор был реализован для управления системой импульсного полива, но его можно использовать и для других приложений, таких как управление шпинделем по каналу PULSE-DIR.

Пример реализации.

1. Добавить код включения генератора в Hardware PLC процедуру включения шпинделя - M03.plc. Код удобно добавить в конец процедуры перед вызовом exit(99)

```
//Установить ускорение генератора
gvarset(8131, 100000); timer=30;do{timer--;}while(timer>0); //Задержка на
30мс

//Преобразовать задание скорости шпинделя в частоту.
//Величина коэффициента подбирается таким образом, чтобы преобразовать
//12-битовое значение скорости шпинделя в частоту генератора
k=123456;
freq=eparam*k; //Посчитать "сырое" значение частоты генератора
```

```
//Послать значение частоты генератора
gvarset(8130,freq); timer=30;do{timer--;}while(timer>0); //Задержка на 30мс
exit(99); //normal exit
```

2. Добавить код включения генератора в Hardwre PLC процедуру регулировки скорости шпинделя - SPN.plc.

```
//Установить ускорение генератора
gvarset(8131, 100000); timer=30;do{timer--;}while(timer>0); //Задержка на
30мс

//Преобразовать задание скорости шпинделя в частоту.
//Величина коэффициента подбирается таким образом, чтобы преобразовать
//12-битовое значение скорости шпинделя в частоту генератора
k=123456;
freq=eparam*k; //Посчитать "сырое" значение частоты генератора

//Послать значение частоты генератора
gvarset(8130, freq); timer=30;do{timer--;}while(timer>0); //Задержка на 30мс

exit(99); //normal exit
```

3. Добавить код выключения генератора в Hardwre PLC процедуру выключения шпинделя - M05.plc. Код также удобно добавить в конец процедуры перед вызовом exit(99)

```
//Послать значение частоты генератора
gvarset(8130,0); timer=30;do{timer--;}while(timer>0); //Задержка на 30мс
exit(99);      //normal exit
```

Скриншоты примера реализации

SYS

PLC

Log

Stat

Info

Support

Cutchart

Config

CNC Settings

Axes/Motors

Inputs/Outputs/Sensors

Network

Motion

PLC

Hardware PLC

Hardware PLC: XML configs

PLC Configuration

Software PLC

G-codes settings

DXF import settings

Macro List

Macro Wizard

Probing Wizard

Preferences

Screen

Work Offsets

Parking Coordinates

Technology

Camera

5 axes RTCP

Panel/Pendant

Hardware

Common Hardware Settings

Encoders

Analogue Closed Loop

Pulse-Dir Closed Loop

ET2/ET4

Host Modbus

Advanced

Profile

Debug

UI Settings

PLC Sources

EST

M00

M01

M02

M03 (.)

M04

M05 (.)

M07

M08

M09

M108

M109

M140

M141

M142

M144

M145

M146

M162

M166

M167

M168

M169

M177

M180

M181

M182

M183

PLC Includes

func.h

mill-func.h

pins.h

vars.h

Name: M03

Aliases:

portclr(OUTPUT_CCW_SPINDLE);

portset(OUTPUT_SPINDLE);

gvarset(7370,1);//Spindle State

timer=30;do{timer--;}while(timer>0); //

gvarset(7371,eparam);//Spindle Speed Mirror register

//gvarset(7372,0);//Mist State

//gvarset(7373,0);//Flood State

gvarset(8131, 100000); timer=30;do{timer--;}while(timer>0); //Задержка на 30мс

k=1234;

freq=eparam*k; //calculate the RAW frequency

gvarset(8130,freq); timer=30;do{timer--;}while(timer>0); //Задержка на 30мс

//delay after spindle started

timer=spindle_on_delay;

do{timer--;}while(timer>0); //delay for Spindle reach given speed

exit(99); //normal exit

};

M71: OK

M74: OK

M79: OK

M80: OK

M81: OK

M82: OK

M85: OK

M87: OK

M88: OK

M89: OK

OFF: OK

SPN: OK

ST001: OK

TRIG04: OK

SYS

PLC

Log

Stat

Info

Support

Cutchart

Config

CNC Settings

Axes/Motors

Inputs/Outputs/Sensors

Network

Motion

PLC

Hardware PLC

Hardware PLC: XML configs

PLC Configuration

Software PLC

G-codes settings

DXF import settings

Macro List

Macro Wizard

Probing Wizard

Preferences

Screen

Work Offsets

Parking Coordinates

Technology

Camera

5 axes RTCP

Panel/Pendant

Hardware

Common Hardware Settings

Encoders

Analogue Closed Loop

Pulse-Dir Closed Loop

ET2/ET4

Host Modbus

Advanced

Profile

Debug

UI Settings

PLC Sources

M167

M168

M169

M177

M180

M181

M182

M183

M210

M397

M398

M399

M62

M63

M71 (.)

M74 (.)

M79

M80

M81

M82

M85

M87

M88

M89

OFF

SPN

ST001

TRIG04

PLC Includes

func.h

mill-func.h

pins.h

vars.h

Name: SPN

Aliases:

#include vars.h

//set Spindle speed control via DAC

main()

{

dac01=val; //send the value to the DAC register

//Change the Spindle State

qvarset(7371,eparam); timer=30;do{timer--;}while(timer>0); //30ms delay

k=12345;

freq=eparam*k; //calculate the RAW frequency

gvarset(8130,freq); timer=30;do{timer--;}while(timer>0); //30ms delay

exit(99);//normal exit

};

M71: OK

M74: OK

M79: OK

M80: OK

M81: OK

M82: OK

M85: OK

M87: OK

M88: OK

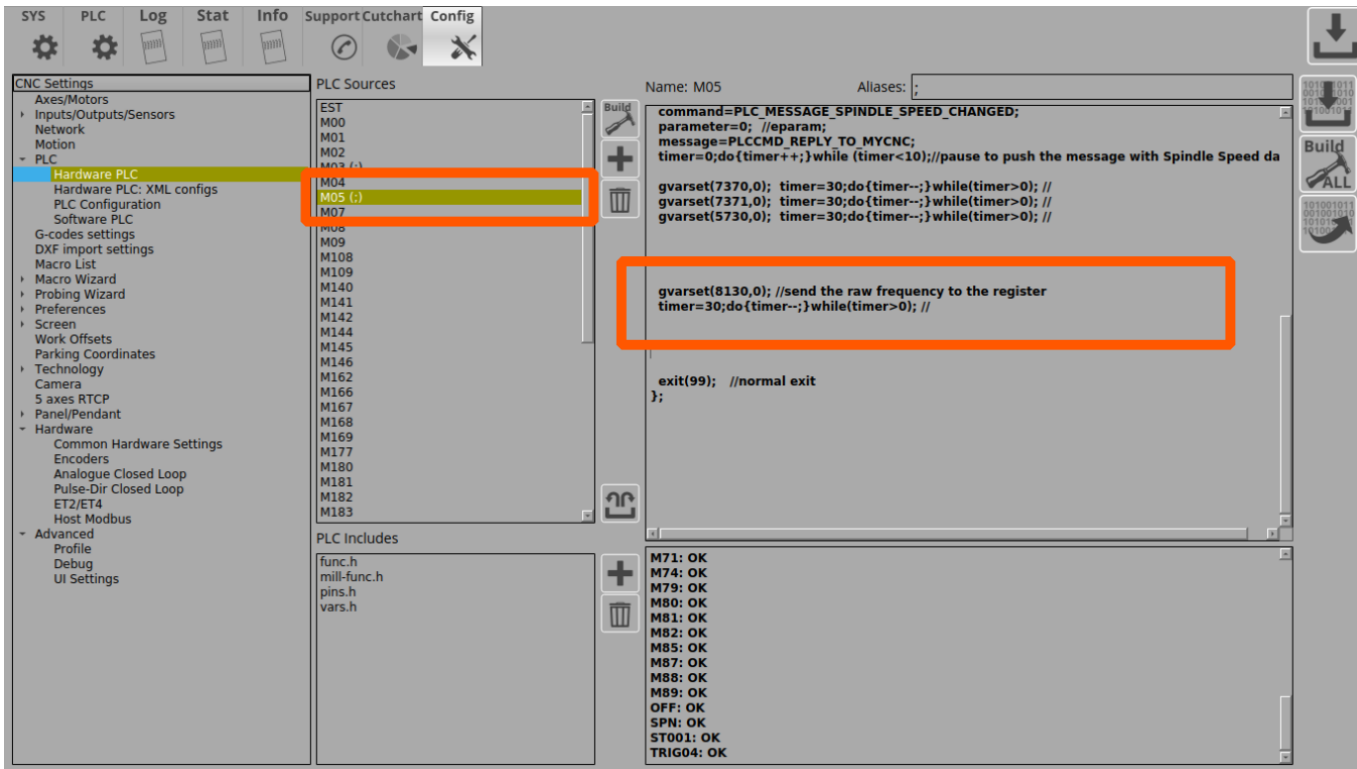
M89: OK

OFF: OK

SPN: OK

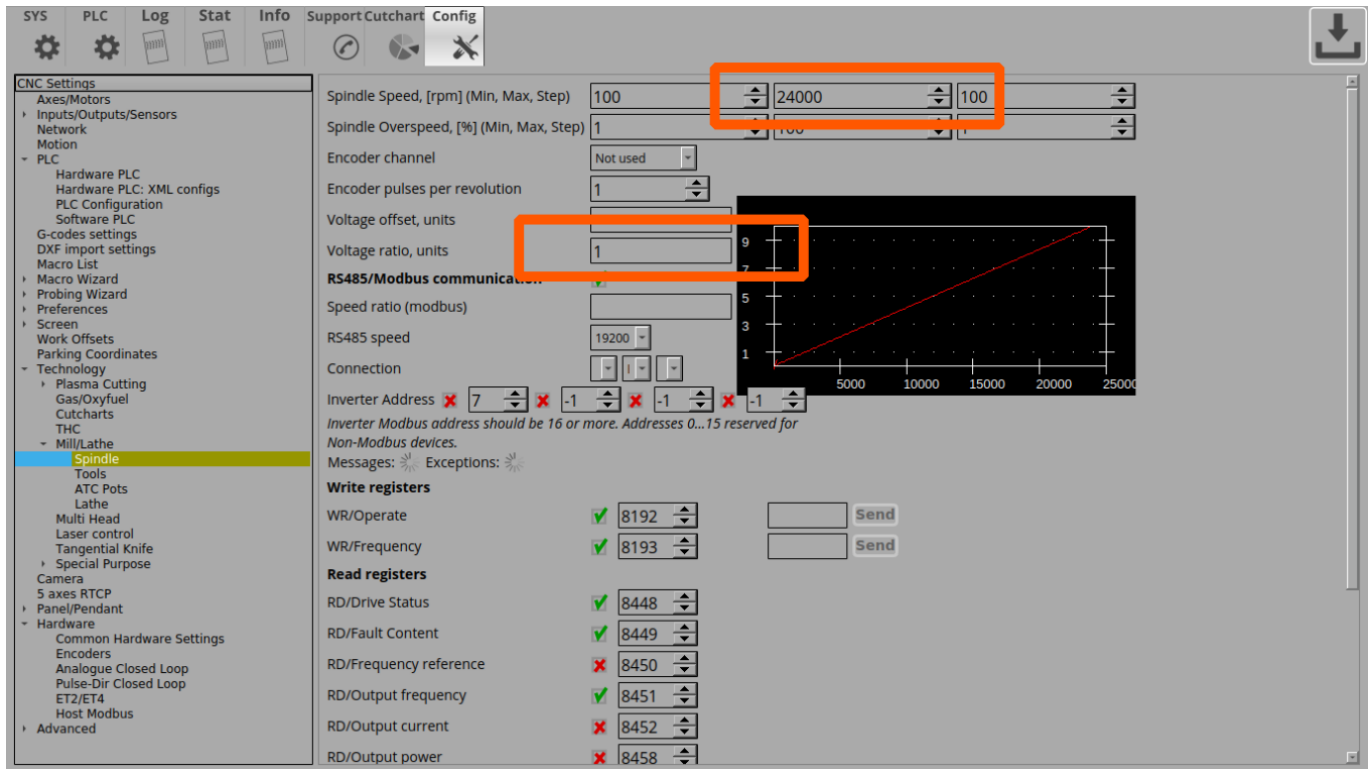
ST001: OK

TRIG04: OK



Если стоит задача подключить одновременно оба шпинделя и переключать их в процессе работы (например по номеру инструмента), то надо организовывать более сложную PLC процедуру, в которой осуществлять проверку номера инструмента, значения глобальной переменной или входа контроллера и по этому условию включать только один из шпинделей.

соответственно, прописана как максимальная скорость шпинделя в настройках



При этой скорости шпинделя на аналоговый выход необходимо выдать полный сигнал 10В, поэтому коэффициент “voltage ratio” выставлен в “1” (в случае, например, шпинделя с диапазоном входного сигнала 0-5В, этот коэффициент был бы 0.5, чтобы на максимальной скорости получить сигнал 5В).

При вызове процедур PLC включения шпинделя (M03.plc) и изменения скорости шпинделя (SPN.plc) значение скорости шпинделя приходит в переменную **eparam**.

myCNC контроллеры имеют 12-бит регистры для значений ШИМ и ЦАП 0-10В. Это означает, что при максимальной скорости шпинделя в “24000 об/мин” и коэффициенте “1” в переменной **eparam** будет максимальное значение в 4095.

Допустим, что максимальная скорость серво-шпинделя составляет 4500 об/мин. Тогда значение **eparam** на скорости в 4500 об/мин составит

$$4500 * (4095/24000) = 768$$

Pulse-Dir вход сервошпинделя настроен на 10000 импульсов, то есть вал двигателя сделает полный оборот при 10000 импульсах. Тогда для достижения полной скорости 4500 об/мин необходима частота пульсов

$$10000 * (4500/60) = 750\ 000$$

Значение “RAW” регистра для частоты 750кГц (750000Гц) должно быть

$$750000/0.0014549=515499347$$

Если максимальная скорость соответствует величине **eparam** “768”, то значение коэффициента для получения “515499347” будет

Метод оценки необходимого ускорения генератора.

Если, в нашем случае максимальная частота составляет 750 000, то ускорение должно быть равно этой же величине, чтобы “разогнаться” до этой частоты за 1 сек.

Тестовые коды для процедур включения шпинделя и регулировки скорости шпинделя

M03.plc

```
//Turn on Spindle clockwise
#include pins.h
#include vars.h
main()
{
    command=PLC_MESSAGE_SPINDLE_SPEED_CHANGED;
    parameter=eparam;
    message=PLCCMD_REPLY_TO_MYCNC;
    timer=0;do{timer++;}while (timer<10);//pause to push the message with
Spindle Speed data

    timer=0;
    proc=plc_proc_spindle;

    val=eparam;
    if (val>0xffff) {val=0xffff;};
    if (val<0) {val=0;};

    dac01=val;

    portclr(OUTPUT_CCW_SPINDLE);
    portset(OUTPUT_SPINDLE);

    gvarset(7370,1);//Spindle State
    timer=30;do{timer--;}while (timer>0); //
    gvarset(7371,eparam);//Spindle Speed Mirror register

    //gvarset(7372,0);//Mist State
    //gvarset(7373,0);//Flood State

    gvarset(8131, 500000); timer=30;do{timer--;}while(timer>0); //Задержка
на 30мс
```

```
k=671223;
freq=val*k; //calculate the RAW frequency
if (freq>515499348) {freq=515499348;};
gvarset(8130,freq); timer=30;do{timer--;}while(timer>0); //Задержка на
30мс

//delay after spindle started
timer=spindle_on_delay;
do{timer--;}while (timer>0); //delay for Spindle reach given speed

exit(99);    //normal exit
};
```

SPN.plc

```
#include vars.h
//set Spindle speed control via DAC
main()
{
    val=eparam;
    dac01=val; //send the value to the DAC register

    //Change the Spindle State
    gvarset(7371,eparam); timer=30;do{timer--;}while (timer>0); //30ms
    delay

    s=gvarget(7370);
    if (s!=0) //if spindle should be ON
    {
        k=671223;
        freq=val*k; //calculate the RAW frequency
        if (freq>515499348) {freq=515499348;};
        gvarset(8130,freq); timer=30;do{timer--;}while(timer>0); //30ms
        delay
    };
    exit(99); //normal exit
};
```

