YSSC2P-A

SSCNET-II PCI Interface Adapter

Руководство пользователя

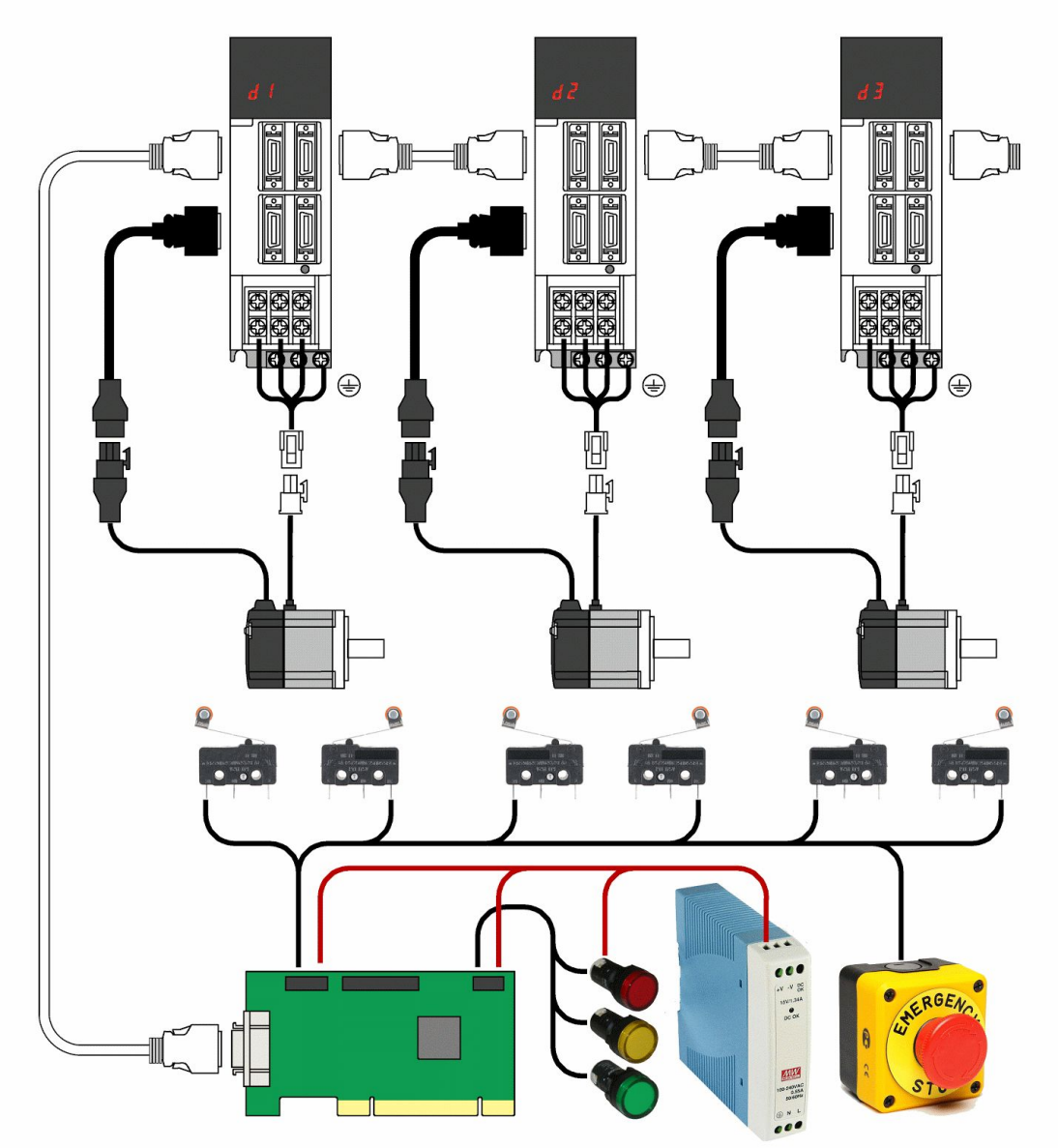
DRAFT

Содержание

Введение

YSSC2P-A – PCI контроллер, совместимый с интерфейсом управления сервоприводами SSCNET II фирмы Mitsubishi Electric. Контроллер позволяет подключать до 6 сервоусилителей типа MR-J2S-B и управлять ими в режиме положения или скорости. Дополнительно на плате контроллера расположены порты дискретного ввода-вывода для подключения к внешним цепям типа концевых выключателей и промежуточных реле. Контроллер работает под управлением LinuxCNC.

SSCNET (Servo System Controller Network) представляет собой синхронную высокоскоростную сеть для сервоприводов и систем управления движением. Для сети SSCNET требуется простейшая проводка, и при этом данная сеть чрезвычайно надежна.



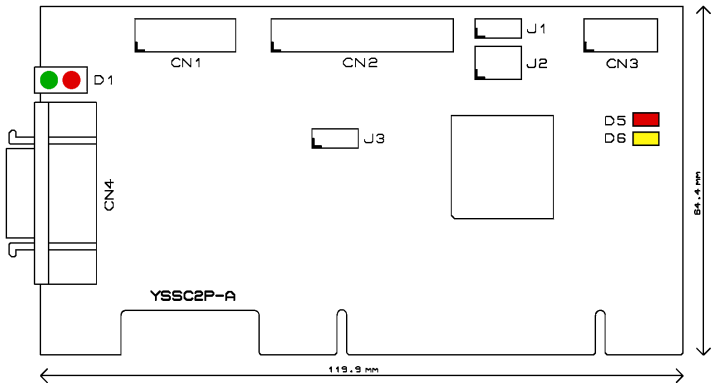
Сервоусилители соединяются с контроллером и между собой при помощи кабелей MR-J2HBUS?M. К последнему сервоусилителю в цепочке подключается терминатор MR-A-TM. На каждом сервоусилителе выставляется уникальный номер оси – от 1 до 6.

Спецификация

- PCI 32bit, 33MHz, 5В или 3.3В
- SSCNET - дуплекс, 5.6 мбит/с, RS-485, управление 1-6 севроусилителями
- Время цикла 0.88 мс, управление позиция/скорость
- 12 оптоизолированных входов 24VDC
- 8 выходов с открытым коллектором, до 30В, 100 мА каждый
- Разъем расширения – 17 двунаправленных линий
- Работа под управлением LinuxCNC
- Форм-фактор low profile PCI, 120 мм x 80 мм

Описание

Общий вид платы контроллера с размещением элементов коммутации и индикаторов:



Индикация

D1	состояние сервоусилителей
D5	ошибка загрузки конфигурации ПЛИС
D6	статус контроллера

Разъемы

CN1	дискретные входы
CN2	разъем расширения
CN3	дискретные выходы
CN4	подключение к сети SSCNET
J2	порт JTAG

Перемчки

J1	подключение питания +5V к CN2
J3	напряжение на подтягивающих резисторах сигналов CN2

D1 – состояние сервоусилителей

зеленый	постоянно	сервоусилители инициализированы контроллером
зеленый	~2 Гц	сервоусилители активны (SERVO-ON)
красный	~2 Гц	ошибка сервоусилителя
зеленый	>5 Гц	режим FLASH-монитора (nyxflash)

D5 – ошибка

красный	постоянно	ошибка начальной конфигурации ПЛИС

D6 – статус контроллера

желтый	постоянно	ошибка начальной конфигурации ПЛИС
желтый	~4 сек	контроллер в режиме ожидания
желтый	~2 Гц	контроллер синхронизирован с LinuxCNC

CN1 – входы

1	IN0	2	IN1
3	IN2	4	IN3
5	IN4	6	IN5
7	IN6	8	IN7
9	IN8	10	IN9
11	IN10	12	IN11
13	COM - вход +24V	14	COM - вход +24V

CN2 – расширение

1	IO0	2	IO1
3	IO2	4	IO3
5	IO4	6	IO5
7	IO6	8	IO7
9	IO8	10	GND
11	IO9	12	GND
13	IO10	14	GND
15	IO11 / A	16	GND
17	IO12 / B	18	GND или +5V
19	IO13 / Z	20	GND или +5V
21	IO14	22	GND или +5V
23	IO15 / RXD	24	GND или +5V
25	IO16 / TXD	26	GND или +5V*

* в зависимоти от положения J1, максимально - 500мА

CN3 – выходы

1	OUT0	2	OUT1
3	OUT2	4	OUT3
5	OUT4	6	OUT5
7	OUT6	8	OUT7
9	COM - вход +24V	10	GND

CN4 – SSCNET

1	LG	11	LG
2	RD	12	RD*
3	TD	13	TD*
4	LG	14	LG
5		15	
6		16	
7	EMG	17	EMG*
8		18	
9		19	
10		20	

J1 – питание на CN2

1-2	контакты CN2 18, 20, 22, 24, 26 подключены к земле
2-3	—//— к +5V*

* +5V с шины PCI, через самовосстанавливающийся предохранитель 0.5A

J3 – подтягивающие резисторы

1-2	контакты CN2 1–9, 11, 13, 15, 17, 19, 21, 23, 25 подтянуты к +5V*
2-3	—//— к земле*

* через резисторы 4.7 кОм

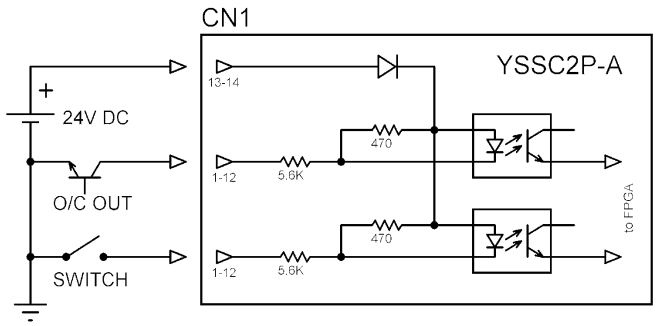
J2 – JTAG

1	TCK	2	GND
3	TDI	4	TDO
5	TMS	6	VCC 3.3V

Подключение

Входы

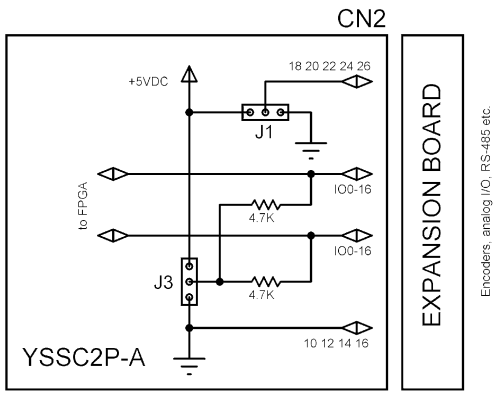
На разъем CN1 выведены 12 дискретных входов. Для работы входных оптоизоляторов необходим источник постоянного тока с напряжением 24В. Пример подключения в выходам с открытым коллектором и замыкающимся контактом:



Расширение

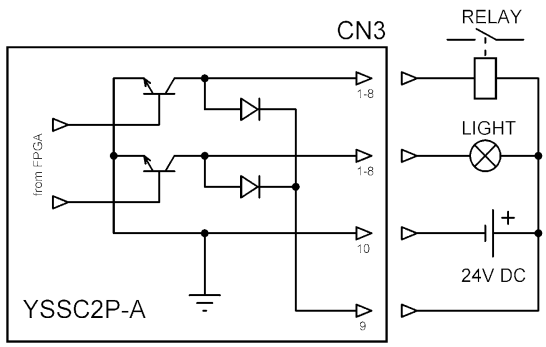
Разъем расширения CN2 предназначен для подключения дочерних плат с дополнительными интерфейсами внешних устройств. В версии контроллера v0.13 на CN2 выведены входы A, B, Z декодера квадратурного сигнала.

Все линии IO0..IO16 подключены к ПЛИС через преобразователи уровней, подтому допускают подключение к 5-вольтовой логике. При подключении 3.3В устройств перемычка J3 должна быть установлена в положение 2-3 - подтяжка к земле.



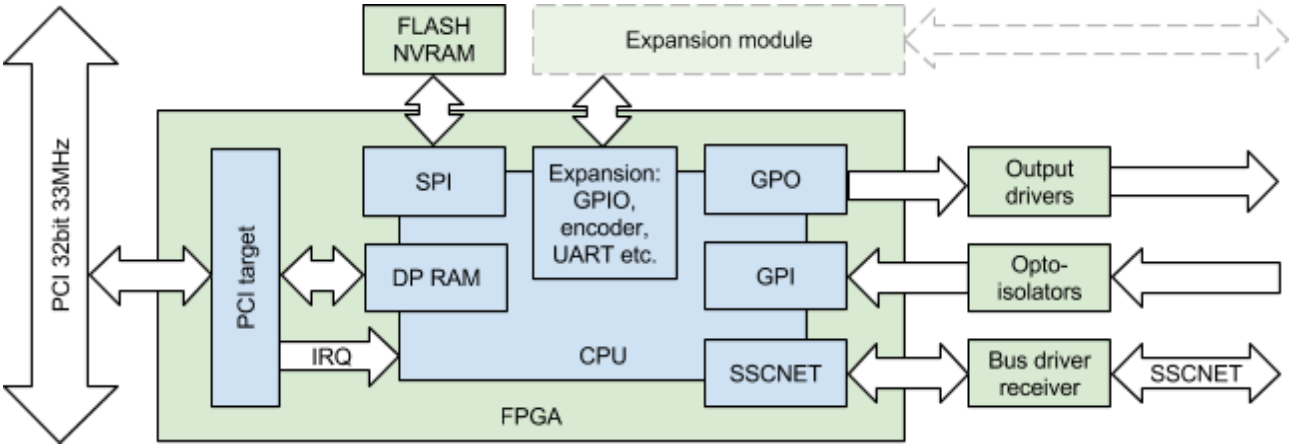
Выходы

На разъем CN3 выведены 8 выходов с открытым коллектором, выполненных на микросхеме ULN2803A. Выходы могут коммутировать до 100 мА постоянного тока с напряжением до 30В. Суммарный ток не должен предышать 500 мА. Вывод 9 подключается к положительному полюсу источника питания. С ним соединены катоды защитных дидов, гасящих обратные выбросы при использовании индуктивной нагруки (реле).

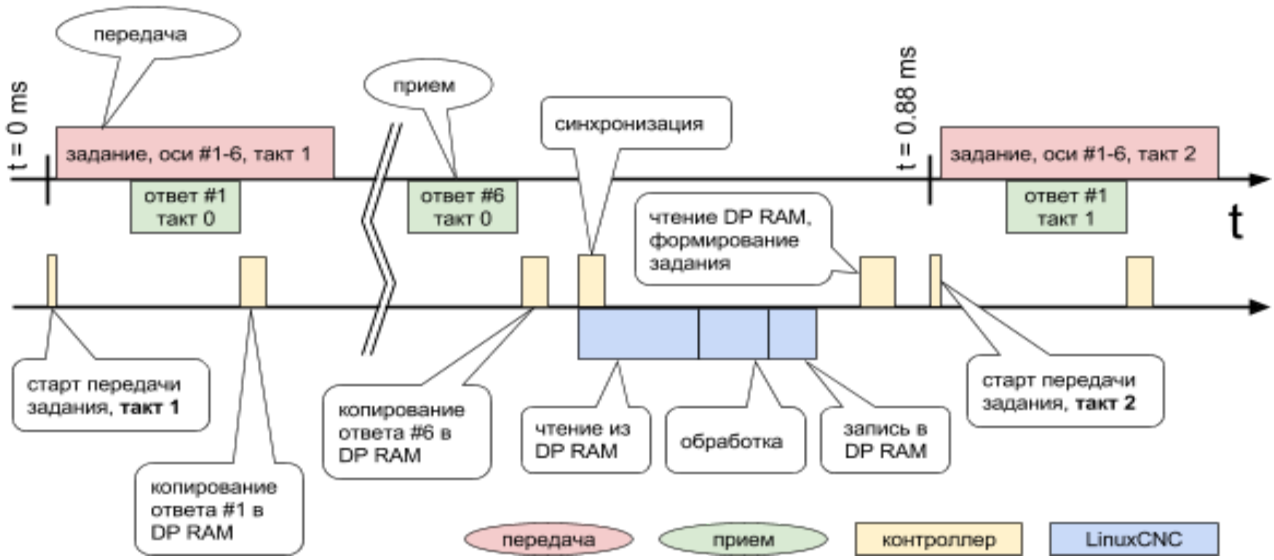


Описание работы контроллера

Контроллер выполнен на ПЛИС Xilinx Spartan 6, в которой реализованы контроллер PCI и система-на-кристалле с встроенным процесоором (soft-core processor). Процессор общается с сервоприводами по сети SSCNET, опрашивает и выдает сигналы на разъемах ввода-вывода и общается с хостом при помощи буфера обмена в двухпортовой памяти.



Программа организована в виде цикла с периодом SSCNET 0.88 мс. Ниже приведена временная диаграмма одного цикла:



Цикл начинается с прерывания таймера, по которому контроллер запускает передачу кадра задания SSCNET. Затем от сервоусилителей, в соответствием с их порядковыми номерами, поступают ответы с feedback данными за прошлй цикл. Контроллер их принимает, обрабатывает и записывает в буфер обмена (DPRAM). После получения ответа сервоусилителя #6 контроллер ожидает получение прерывания от хоста, при помощи которого осуществляется синхронизация циклов контроллера и LinuxCNC. В зависимости от времени его получения контроллер корректирует длительность следующего цикла.
LinuxCNC каждый цикл сервопотока вызывает подпрограмму в драйвере. В начале выполнения генерируется прерывание для синхронизации цикла контроллера. Затем копируются данные из буфера обмена в оперативную память. Полученные данные обрабатываются и присваиваются выходным пинам. В зависимости от состояния сигналов HAL на входных пинах в буфере обмена формируется задание на следующий цикл. Время выполнения подпрограммы в драйвере должно уложиться в отведенное протоколом окно XX мкс.
После этого контроллер получает второе прерывание от таймера, после которого читает задание из буфера обмена и формирует кадр задания. Затем цикл повторяется с передачи задания.

Драйвер HAL

Модуль драйвера контроллера называется sscii. Так как контроллер дожен быть синхронизирован с LinuxCNC, в конфигурации HAL необходимо задать время цикла сервопотока 888888 наносекунд. Загрузить модуль и добавить его вызов в сервопоток

```
loadrt motmod servo_period_nsec=888888 ...
loadrt sscii
addf sscii.0 servo-thread
```

NB! Хотя драйвер поддерживает несколько контроллеров, но из-за real-time ограничений и текущей реализации PCI интерфейса такая конфигурация работать не будет. Также скоростные устройства на той же PCI шине (сетевые карты, Wi-Fi адаптеры, FireWire и т.п) будут мешать фунционированию LinuxCNC с контроллером.

SSCII

Состояние контроллера. Пины:

- **sscii.0.ready** (bit, out) – контроллер синхронизирован с 0.88 мс сервоциклом LinuxCNC. До этого момента другие пины не активны
- **sscii.0.error-cnt** (u32, out) – счетчик ошибок. Количество тактов, во время которых отсутствовала синзронизация с контроллером. Если значение счетчика увеличивается, вероятно jitter сервопотока слишком высок. Допуснимое значение порядка 100 мкс.
- **sscii.0.phase** (float, out) – задержка сервопотока относительно сервоцикла контроллера, микросекунд. Не должна превышать ±50 мкс.

Servo

Параметры:

- **sscii.0.servo.<axis>.pos-scale** (float, rw) – масштаб позиции, единиц длины на оборот. По-умолчанию равен 5, что соответствует конфигурации с прямым соединением сервомотора с ШВП с шагом 5 мм
- **sscii.0.servo.<axis>.vel-scale** (bit, rw) – масштаб скорости. По-умолчанию равен 10, что соответсвует заданию скорости **vel-cmd** в оборотах в минуту с разрешением 0.1 RPM

Пины:

- **sscii.0.servo.<axis>.online** (bit, out) – контроллер инициализировал сервоусилитель оси <axis> и он готов к работе
- **sscii.0.servo.<axis>.offline** (bit, out) – инвертированный пин online
- **sscii.0.servo.<axis>.ready** (bit, out) – сервоусилитель в состоянии READY-ON
- **sscii.0.servo.<axis>.enabled** (bit, out) – сервоусилитель активен – SERVO-ON
- **sscii.0.servo.<axis>.warning** (bit, out) – предупреждение
- **sscii.0.servo.<axis>.alarm** (bit, out) – сервоусилитель в состоянии ошибки
- **sscii.0.servo.<axis>.alarm-code** (u32, out) – код ошибки
- **sscii.0.servo.<axis>.zero-speed** (bit, out) – нулевая скорость вращения
- **sscii.0.servo.<axis>.in-position** (bit, out) – позиционирование выполнено
- **sscii.0.servo.<axis>.power** (bit, in) – включить силовое реле (READY-ON).
- **sscii.0.servo.<axis>.enable** (bit, in) – включить серво (SERVO-ON)
- **sscii.0.servo.<axis>.pos-cmd** (float, in) – задание позции от ЧПУ
- **sscii.0.servo.<axis>.pos-fb** (float, out) – текущая позиция мотора.
- **sscii.0.servo.<axis>.velocity-mode** (bit, rw) – режим управления скоростью. Изменение режима возможно при нулевой скорости
- **sscii.0.servo.<axis>.vel-cmd** (float, in) – задание скорости от ЧПУ
- **sscii.0.servo.<axis>.vel-fb** (float, out) – текущая скорость

- **sscii.0.servo.<axis>.trq-fb** (float, out) – текущий момент, развиваемый двигателем, в процентах от номинала
- **sscii.0.servo.<axis>.droop** (s32, out) – рассогласование, отсчетов энкодера
- **sscii.0.servo.<axis>.error-cnt** (s32, out) – счетчик ошибок связи с сервоусилителем

- **sscii.0.servo.<axis>.limit-torque** (bit, in) – включить ограничение крутящего момента
- **sscii.0.servo.<axis>.forward-torque** (float, in) – задание ограничения крутящего момента при прямом вращении
- **sscii.0.servo.<axis>.reverse-torque** (float, in) – задание ограничения крутящего момента при обратном вращении
- **sscii.0.servo.<axis>.torque-clamped** (bit, out) – индикация ограничения крутящего момента
- **sscii.0.servo.<axis>.absolute-pos-valid** (bit, out) – абсолютная позиция действительна

Encoder

Декодер квадратурных сигналов. Параметры:

- **sscii.0.encoder.0.cpr** (float, rw) – разрешение энкодера, отсчетов на оборот. Знак определяет направление счета. По-умолчанию равен “–10000”

Пины:

- **sscii.0.encoder.<enc>.index-enable** (bit, io) – обнулить счетчик энкодера при прохождении индексной метки Z
- **sscii.0.encoder.<enc>.pos** (float, out) – значение счетчика энкодера деленное на параметр cpr. Дробное число оборотов энкодера со знаком.

GPIO

Ввод-вывод, пины:

- **sscii.0.gpio.<n>.in** (bit, out) – состояние дискретного входа
- **sscii.0.gpio.<n>.in-not** (bit, out) – инвертированое состояние дискретного входа
- **sscii.0.gpio.<n>.out** (bit, in) – установаить состояние выхода
- **sscii.0.gpio.<n>.enable** (bit, in) – направление сигнала - выход

Пример конфигурации HAL

```
#
# 3 axen + spindle SSCNET control using YSSC2P-A board
#

loadrt trivkins
loadrt motmod base_period_nsec=888888 servo_period_nsec=888888 num_joints=3
addf motion-command-handler servo-thread
addf motion-controller servo-thread
loadrt sscii
addf sscii.0 servo-thread

# limit switches at CN1.1 .. 5
net x-pos-limit axis.0.pos-lim-sw-in <= sscii.0.gpio.0.in-not
net x-neg-limit axis.0.neg-lim-sw-in <= sscii.0.gpio.1.in-not
net y-pos-limit axis.1.pos-lim-sw-in <= sscii.0.gpio.2.in-not
net y-neg-limit axis.1.neg-lim-sw-in <= sscii.0.gpio.3.in-not
net z-pos-limit axis.2.pos-lim-sw-in <= sscii.0.gpio.5.in-not
net z-neg-limit axis.2.neg-lim-sw-in <= sscii.0.gpio.4.in-not
net x-neg-limit => axis.0.home-sw-in # shared home/limits
net y-neg-limit => axis.1.home-sw-in
net z-pos-limit => axis.2.home-sw-in

# motion
setp sscii.0.servo.0.pos-scale 5.0
setp sscii.0.servo.1.pos-scale 5.0
setp sscii.0.servo.2.pos-scale 2.5 # 1:2 reduction

net x-pos-fb axis.0.motor-pos-fb <= sscii.0.servo.0.pos-fb
net y-pos-fb axis.1.motor-pos-fb <= sscii.0.servo.1.pos-fb
net z-pos-fb axis.2.motor-pos-fb <= sscii.0.servo.2.pos-fb

net x-cmd axis.0.motor-pos-cmd => sscii.0.servo.0.pos-cmd
net y-cmd axis.1.motor-pos-cmd => sscii.0.servo.1.pos-cmd
net z-cmd axis.2.motor-pos-cmd => sscii.0.servo.2.pos-cmd

net x-enable axis.0.amp-enable-out => sscii.0.servo.0.enable
net y-enable axis.1.amp-enable-out => sscii.0.servo.1.enable
net z-enable axis.2.amp-enable-out => sscii.0.servo.2.enable

net x-fault axis.0.amp-fault-in <= sscii.0.servo.0.alarm
net y-fault axis.1.amp-fault-in <= sscii.0.servo.1.alarm
net z-fault axis.2.amp-fault-in <= sscii.0.servo.2.alarm

# e-stop cuts amplifier power
net estop-out => sscii.0.servo.0.power
net estop-out => sscii.0.servo.1.power
net estop-out => sscii.0.servo.2.power
net estop-out => sscii.0.servo.5.power

# sscnet spindle
setp sscii.0.servo.5.vel-scale 10
setp sscii.0.servo.5.vel-ctl true       # velocity control

loadrt limit2 names=spindle-ramp
loadrt near names=spindle-at-speed

addf spindle-ramp servo-thread
setp spindle-ramp.maxv 3000     # у с к о р е н и е, о б/м и н*с е к
addf spindle-at-speed servo-thread

net spindle-cmd motion.spindle-speed-out => spindle-ramp.in
net spindle-ramped spindle-ramp.out => sscii.0.servo.5.vel-cmd
net spindle-fb motion.spindle-speed-in <= sscii.0.servo.5.vel-fb
net spindle-cmd => spindle-at-speed.in1
net spindle-ramped => spindle-at-speed.in2
net spindle-ready spindle-at-speed.out => motion.spindle-at-speed
net spindle-enable motion.spindle-on => sscii.0.servo.5.enable
net spindle-idx-en sscii.0.encoder.0.index-enable <=> \
                     motion.spindle-index-enable
```

```
net spindle-position sscii.0.encoder.0.pos => motion.spindle-revs

# enable spindle reverse
net trick1 motion.spindle-forward
net trick2 motion.spindle-reverse

# estop from the AXis UI
net estop-out <= iocontrol.0.user-enable-out
net estop-out => iocontrol.0.emc-enable-in
```

Обновление микропрограммы

Конфигурация ПЛИС и ПО встроенного процессора хранится в микросхеме энергонезависимой памяти FLASH. Память разделена на секторы по 64 Кбайт. Сектор 0 содержит начальный загрузчик, секторы 1-6 – резервную копию прошивки, 8-13 - оновную копию. Запись выполняется при помощи утилиты nyxflash с именем файла прошивки в качестве аргумента. Запускать от root.

```
# nyxflash nyx_fw_v0.12.bin
writing nyx_fw_v0.12.bin 340604 bytes to 0x80000
erasing sector #8 #9 #10 #11 #12 #13
writing sector #8 #9 #10 #11 #12 #13
programming successful
exiting monitor
```

Опция BOOT записывает начальный загрузчик:

```
# nyxflash BOOT
writing BOOT 64 bytes to 0x0
erasing sector #0
writing sector #0
programming successful
exiting monitor
```

Вторым аргументом можно указать шестнадцатеричный адрес, по которму будет производиться запись. Например, записать резервную копию прошивки:

```
# nyxflash nyx_fw_v0.14.bin 10000
writing nyx_fw_v0.12.bin 340604 bytes to 0x10000
rasing sector #1 #2 #3 #4 #5 #6
writing sector #1 #2 #3 #4 #5 #6
programming successful
exiting monitor
```

Ключевое слово ERASE стирает сектор по адресу, указанному третьим аргументом при запуске:

```
# nyxflash ERASE 0
erasing sector #0
exiting monitor
```

С осторожностью! При ошибках записи, не выключая компьютер, повторить команду. Если поврежден начальный загрузчик или обе версии прошивки, то для восстановления необходимо подключение программатора.

LinuxCNC installation: debian/wheezy

1. install
2. passwd root
3. vi /etc/lightdm/lightdm.conf
   autologin-user=<user>
   autologin-user-timeout=0
4. unzip nyx_v0.14b.zip
   cd nyx_v0.14
   halcompile --install sscii.comp
5. linuxcnc