YSSC2P-A

SSCNET-II PCI Interface Adapter

User manual

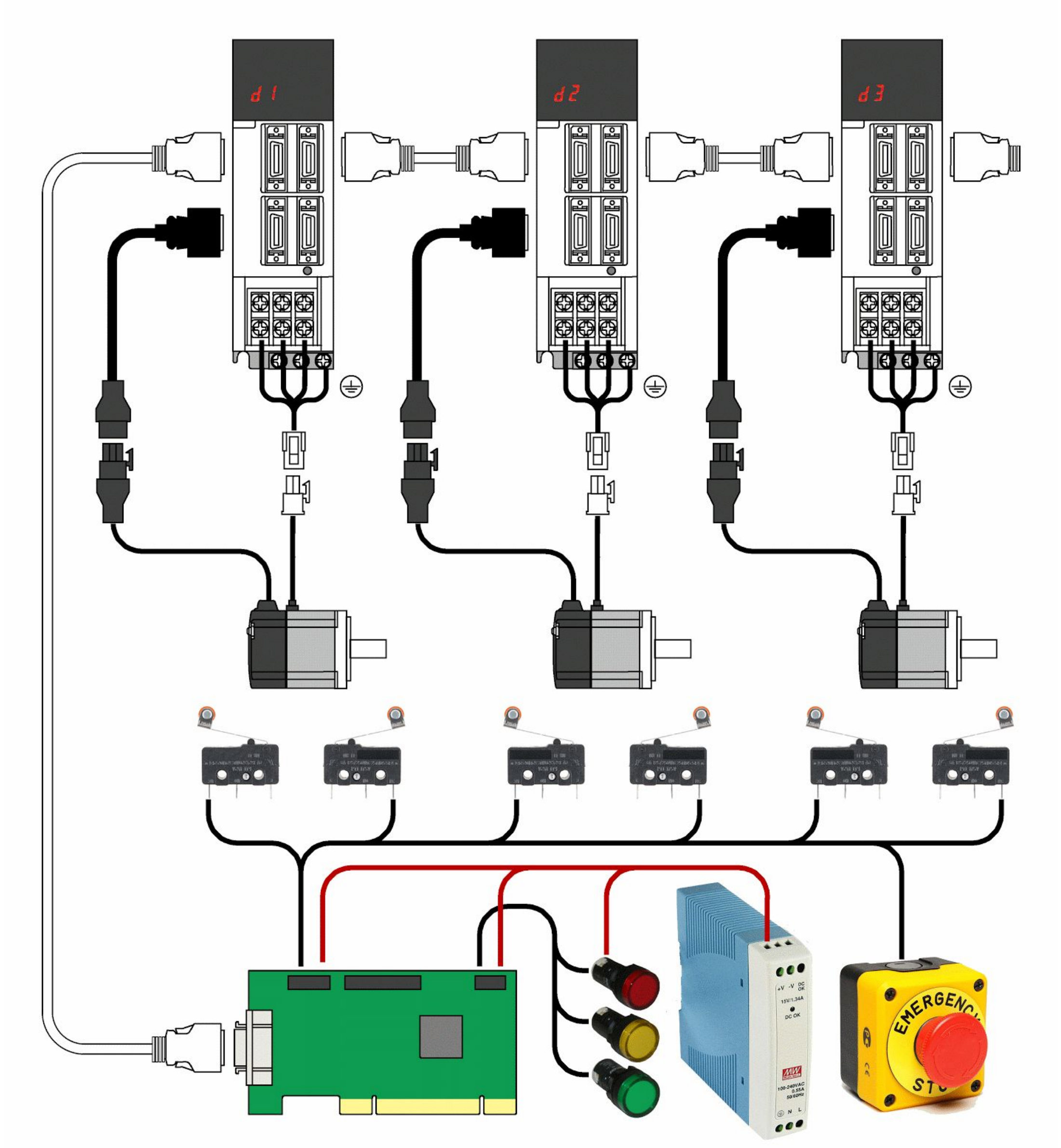
Contents

Introduction

YSSC2P-A is a PCI interface controller compatible with Mitsubishi MR-J2S-B SSCNET II network servo amplifiers. It support up up to six MR-J2S-B drives in position and velocity control mode. The controller also features general-purpose I/O headers for connecting to limit switches and relays. The software includes LinuxCNC 2.7.4 driver module and an optional patch for absolute encoder support.

There’s an early experimental firmware version with can be used with MACH3 software without any additional software plugins or drivers via bidirectional parallel port emulation.

SSCNET is a synchronous high-speed network for servo drives and motion controllers. SSCNET only requires a simple daisy chain wiring between servo amplifiers. See Mitsubishi documentation for details.

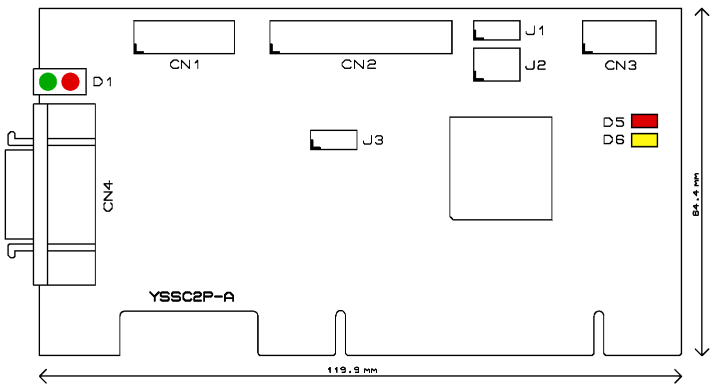


Specifications

- PCI 32bit, 33MHz, 5V or 3.3V
- Xilinx Spartan-6 FPGA
- Compatible servo amplifiers: MR-J2S-*B
- SSCNET II - 5.6 Mbit/s, RS-485 signalling, controls 1 to 6 servo amplifiers
- 0.88ms cycle, position or velocity control
- 12 optoisolated digital inputs, 24VDC
- 8 open collector outputs, up to 30VDC, 100mA each
- Expansion connector – 17 bidirectional 5V tolerant I/O lines to the FPGA
- LinuxCNC driver
- Low profile PCI board, 120 mm x 80 mm

Board layout

The controller board layout showing connectors and indication LEDs:



LEDs

D1	servo amplifier status
D5	FPGA boot error
D6	controller status

Connectors

CN1	digital inputs
CN2	expansion connector
CN3	digital outputs
CN4	SSCNET connector
J2	JTAG connector

Jumpers

J1	+5V power on CN2
J3	pull-up/pull-down resistor at CN2

D1 – servo amplifier status

green	continuous	servo amplifiers initialized
green	~2 Hz	SERVO-ON
red	~2 Hz	servo amplifier alarm
green	>5 Hz	firmware FLASH monitor (nyxflash)

D5 – error

red	continuous	FPGA boot error

D6 – controller status

amber	continuous	FPGA boot error
amber	~4 s	controller stand-by
amber	~2 Hz	controller is synchronized with LinuxCNC

CN1 – digital inputs

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 - +24V input	14	COM - +24V input

CN2 – expansion

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 or +5V
19	IO13 / Z	20	GND or +5V
21	IO14	22	GND or +5V
23	IO15 / RXD	24	GND or +5V
25	IO16 / TXD	26	GND or +5V*

* depends on J1 position, up to 500mA (fused)

CN3 – digital outputs

1	OUT0	2	OUT1
3	OUT2	4	OUT3
5	OUT4	6	OUT5
7	OUT6	8	OUT7
9	COM - +24V input	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 – CN2 power

1-2	CN2 pins 18, 20, 22, 24, 26 connected to ground
2-3	—//— to +5V*

* +5V is sourced from PCI connector via 500mA self-resettable fuse

J3 – weak pull ups/downs

1-2	CN2 pins 1–9, 11, 13, 15, 17, 19, 21, 23, 25 pulled up to +5V*
2-3	—//— pulled down to GND*

* via 4.7k resistors

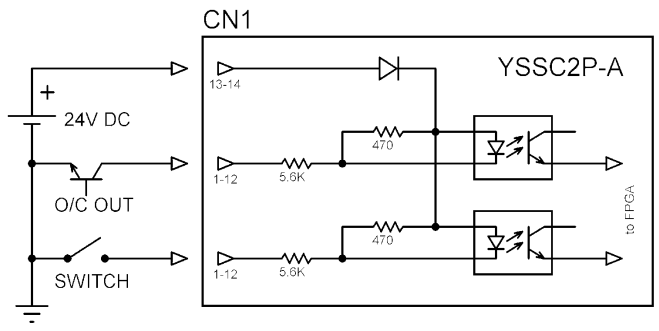
J2 – JTAG

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

Connections

Digital inputs

CN1 contains 12 optoisolated digital inputs. External 24VDC field power supply is required for operation. Example connection of open-collector outputs or mechanical switches is shown on schematics. When using non-contact proximity sensors choose NPN output type.

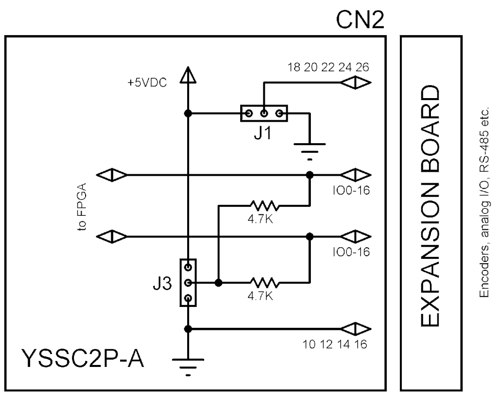


Expansion

CN2 connector is intended for daughter boards with future firmware upgrade. Version 1.0 supports TTL quadrature encoder inputs A, B, Z.

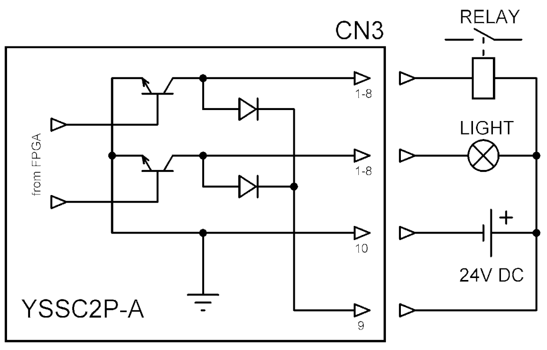
All IO0..IO16 lines are 3.3V LVTTL and are 5V tolerant.

Pins 18, 20, 22, 24, 26 optionally provide 5V power to external devices. The power is taken from the PCI bus and fused to 500mA.



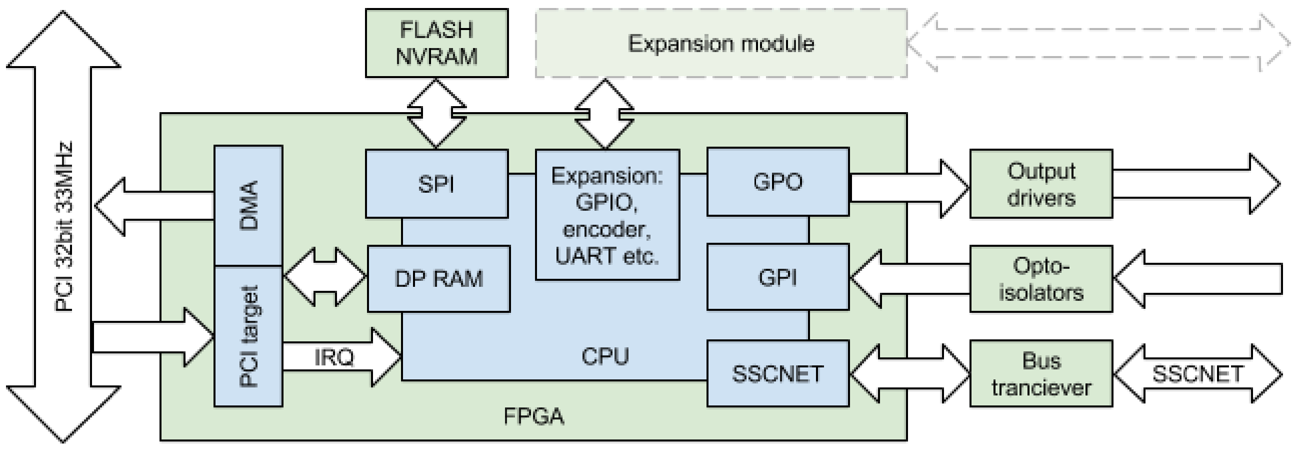
Digital outputs

CN3 has 8 open-collector outputs from ULN2803A IC. It is capable of 100 mA/30V. Total current should not exceed 500mA. Pin 9 connects to positive terminal of the field power supply for flyback protection diodes when using inductive load such as magnetic relays.

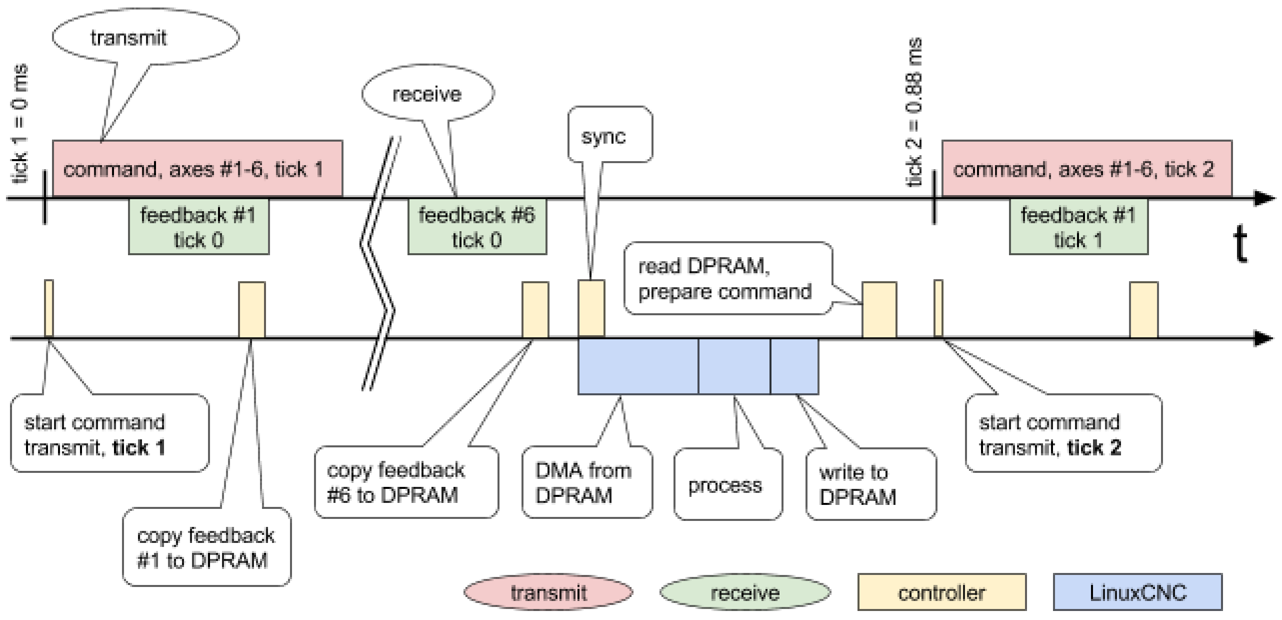


Theory of operation

The controller is implemented in a Xilinx Spartan-6 FPGA. It includes a PCI controller core and a system-on-a-chip with a soft-core CPU. The CPU executes a firmware which handles communication with servo drives and I/O processing. The communication with the host computer is handled via a dual-ported RAM and interrupts.



The software executes an endless loop with a 0.88ms period. Below is a timing diagram of a cycle:


transmit
tick 1 = 0 ms
command, axes #1-6, tick 1
feedback #1 tick 0
receive
feedback #6 tick 0
sync
tick 2 = 0.88 ms
command, axes #1-6, tick 2
feedback #1 tick 1
t
read DPRAM, prepare command
start command transmit, tick 1
copy feedback #1 to DPRAM
copy feedback #6 to DPRAM
DMA from DPRAM
process
write to DPRAM
start command transmit, tick 2
transmit
receive
controller
LinuxCNC

The controller cycle begins with a timer interrupt which starts a transmission of a SSCNET control frame. In response servo amplifiers transmit a feedback frames in sequence defined by their IDs. The controller receives, processes and writes it to the exchange buffer in the DPRAM. After all feedback received the controller waits for a sync interrupt from the host driver. Measuring the timing of the interrupt the controller slightly adjusts its 0.88 timer so that to be in sync with the servo thread of the LinuxCNC.
On each servo thread cycle LinuxCNC calls driver function. At the beginning of execution the drivers generates a sync interrupt to the controller processor. Following that the driver starts the DMA engine to copy feedback data from the dual-port buffer RAM to system RAM with a PCI burst transfer. Received data is used to update HAL output pins of the driver. Then driver HAL input pins are read and values used to compose a command packet for the next cycle which is written to the dual-port RAM buffer.
Then the controller processor receives second timer interrupt on which it reads command packet from DPRAM buffer, and prepares a SSCNET control frame using the data received from the host driver. The cycle repeats.

LinuxCNC driver module

LinuxCNC HAL driver module name is sscii. During operation the controller and SSCNET network are synchronized to servo thread of the LinuxCNC. Therefore servo thread period should be set to 888888 ns - the period of SSCNET-II network. The following HAL commands load the driver and add its callback function to the servo thread:

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

SSCII

Controller status pins:

- **sscii.0.ready** (bit, out) – the controller is in sync with LinuxCNC 0.88ms servo thread. Before it come true other pins are inactive
- **sscii.0.error-cnt** (u32, out) – communication error count. Number of clocks when servo thread sync was absent. If the value increases then the jitter is probably too large. Jitter tolerance is about 100 microseconds
- **sscii.0.phase** (float, out) – offset of the servo thread call relative to the timing slot of controller’s 0.88ms cycle, in microseconds. Should not exceed ±50us

Servo

THe driver support up to 6 connected servo amplifiers numbered <axis> 0 thru 5. Parameters:

- **sscii.0.servo.<axis>.pos-scale** (float, rw) – position scale factor, in length units per revolution. Default is 5, which corresponds to 5, which is suitable for direct motor coupling to 5mm ball screw
- **sscii.0.servo.<axis>.vel-scale** (bit, rw) – velocity scable factor. Default is 10, then **vel-cmd** is in rotation per minute with 0.1 rpm resolution

Pins:

- **sscii.0.servo.<axis>.online** (bit, out) – the controller has detected and initialized <axis> servo amplifier
- **sscii.0.servo.<axis>.offline** (bit, out) – inverted “online” pin output
- **sscii.0.servo.<axis>.ready** (bit, out) – READY-ON (power relay is on)
- **sscii.0.servo.<axis>.enabled** (bit, out) – SERVO-ON
- **sscii.0.servo.<axis>.warning** (bit, out) – servo amplifier warning
- **sscii.0.servo.<axis>.alarm** (bit, out) – servo amplifier alarm
- **sscii.0.servo.<axis>.alarm-code** (u32, out) – servo amplifier alarm code (hex)
- **sscii.0.servo.<axis>.zero-speed** (bit, out) – motor velocity is below zero speed threshold (param No.30)
- **sscii.0.servo.<axis>.in-position** (bit, out) – position complete (threshold set in param No.20)

- **sscii.0.servo.<axis>.power** (bit, in) – turn on amplifier power relay (READY-ON).
- **sscii.0.servo.<axis>.enable** (bit, in) – enable servo (SERVO-ON)
- **sscii.0.servo.<axis>.pos-cmd** (float, in) – commanded position
- **sscii.0.servo.<axis>.pos-fb** (float, out) – motor feedback position
- **sscii.0.servo.<axis>.velocity-mode** (bit, rw) – velocity control mode. Change is possible at zero speed only
- **sscii.0.servo.<axis>.vel-cmd** (float, in) – commanded velocity
- **sscii.0.servo.<axis>.vel-fb** (float, out) – motor velocity feedback
- **sscii.0.servo.<axis>.trq-fb** (float, out) – motor torque feedback, percents from nominal motor torque
- **sscii.0.servo.<axis>.droop** (s32, out) – droop pulse count. The difference between commanded and actual position of servo motor

- **sscii.0.servo.<axis>.error-cnt** (s32, out) – controller-amplifier feedback transfer error count. Errors may be due to faulty connection of the SSCNET cables

- **sscii.0.servo.<axis>.limit-torque** (bit, in) – turn on torque limiting
- **sscii.0.servo.<axis>.forward-torque** (float, in) – forward rotation torque limit, percents
- **sscii.0.servo.<axis>.reverse-torque** (float, in) – reverse rotation torque limit
- **sscii.0.servo.<axis>.torque-clamped** (bit, out) – torque clamp indication

- **sscii.0.servo.<axis>.absolute** (bit, out) – absolute position feedback is valid (param No.1)
- **sscii.0.servo.<axis>.abs-pos-lost** (bit, out) – absolute position is not valid

Encoder

An encoder input channel converts quadrature signals on Phase A, Phase B, Index Z into 32-bit up/down counter values. The maximum count frequency is 20 MHz. Parameter:

- **sscii.0.encoder.0.cpr** (float, rw) – encoder resolution, counts per revolution. Sign defines rotation direction. Default is minus 10000

Pins:

- **sscii.0.encoder.<enc>.index-enable** (bit, io) – reset counter on next index mark Z
- **sscii.0.encoder.<enc>.pos** (float, out) – encoder counter divided by cpr parameter value. Signed fractional number of revolutions.

The interface is intended for use with motion.spindle-index-enable and **motion.spindle-revs.**

GPIO

Digital input signals IN0..IN11 connected to CN1. Pins:

- **sscii.0.gpio.<n>.in** (bit, out) – a value of digital input <n> 0 thru 11
- **sscii.0.gpio.<n>.in-not** (bit, out) – inverted

Digital output signals OUT0..OUT7 going to CN3:

- **sscii.0.gpio.<n>.out** (bit, in) – set output number <n> 0 thru 7

Absolute encoder support

All Mitsubishi J2 Super series motors feature absolute encoders. This allows to eliminate LinuxCNC homing operation on each startup. As LinuxCNC 2.7.4 lacks support of absolute encoders a patch is provided. It adds pins to halui module:

- **halui.joint.<n>.set-homed** (bit, in) – set axis “homed” state to true for current offsets without actually performing a homing operation

Upon startup a script is executed which inspects if servo amplifiers are in absolute mode and position data is valid. In that case it sets corresponding axes to “homed”. Also the patch changes values saved into **position.txt** file from current position **pos_fb** to **motor_offset**. TODO: change to preserve compatibility.

Added INI file parameters MAX_JOG_VELOCITY, MAX_AJOG_VELOCITY to differentiate maximum jogging velocity from absolute maximum velocity that can be used in programs. AXIS UI sliders “Jog Speed” and “Max Velocity” behave consistent.

Firmware upgrade

FPGA configuration and embedded processor firmware are stored in a 16 mbit FLASH memory chip. Memory is dividen to 64 KBytes sectors. Sector 0 contains a bootloader, sectors 1-6 – backup configuration, 8-13 - primary configuration. FLASH writing is done by running the **nyxflash** utility given a firmware image file name as an argument. Should be run as 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
```

Writing bootloader is done with BOOT name:

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

Optional second argument specifies starting offset, in hex bytes. For example, writing a backup configuration is done with:

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

Giving ERASE as an argument erases a sector at given address:

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

Caution! If the utility reports verification errors do not turn off the computer. Repeat the writing command instead. If the bootloader or both primary and backup configurations are damaged then the board is rendered unusable without FPGA JTAG programmer.

Disclaimer

This protocol implementation is in no way associated with Mitsubishi Electric and based solely on reverse engineering. Because of that and the variety of uses for this equipment, the user of and those responsible for applying this equipment must satisfy themselves as to the acceptability of each application and the use of the equipment. The illustrations in this manual are intended solely to illustrate the text of this manual. Because of the many variables and requirements associated with any particular installation, the manufacturer cannot assume responsibility or liability for actual use based upon the illustrative uses and applications. In no event will the manufacturer be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment. THE MANUFACTURER DISCLAIMS ANY IMPLIED WARRANTY OR FITNESS FOR A PARTICULAR PURPOSE.