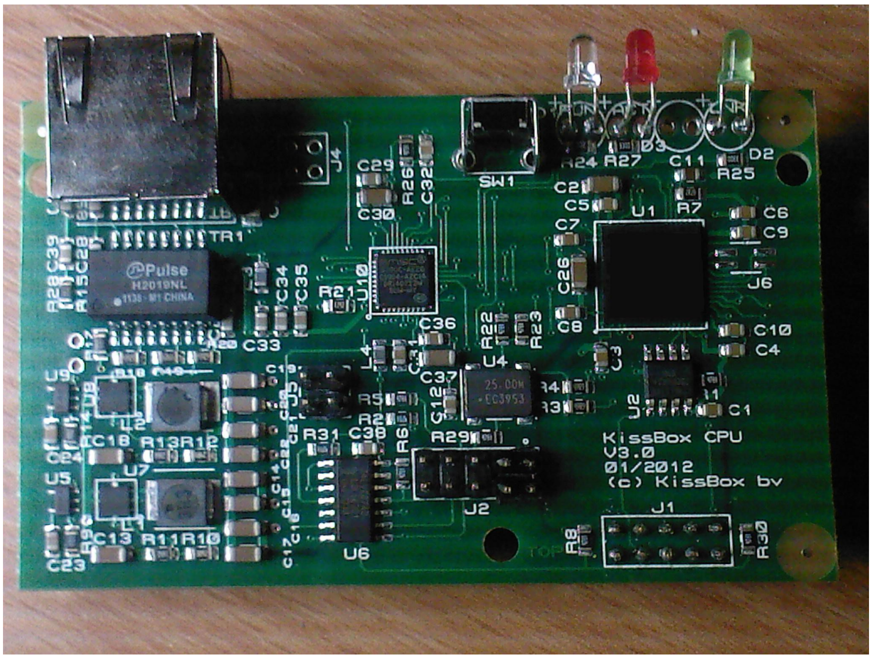
KISS-BOX

RTP-MIDI/HD NETWORK CPU

OEM HOST INTEGRATION MANUAL



Contents

1 - Introduction

The KissBox Network CPU is a compact embedded computer board, specially designed for audio / synthesizers / show-control networked applications, based on the KissBox V3 CPU.

The OEM RTP-MIDI firmware is designed to turn the V3 CPU into a network processor in charge of handling all lower communication layers and provide directly MIDI and HD data to an external application board.

The external processor can then be optimized for dedicated tasks. For example, associated with a DSP, the network CPU frees the DSP from all communication related tasks, providing the maximum computation power to signal processing tasks.

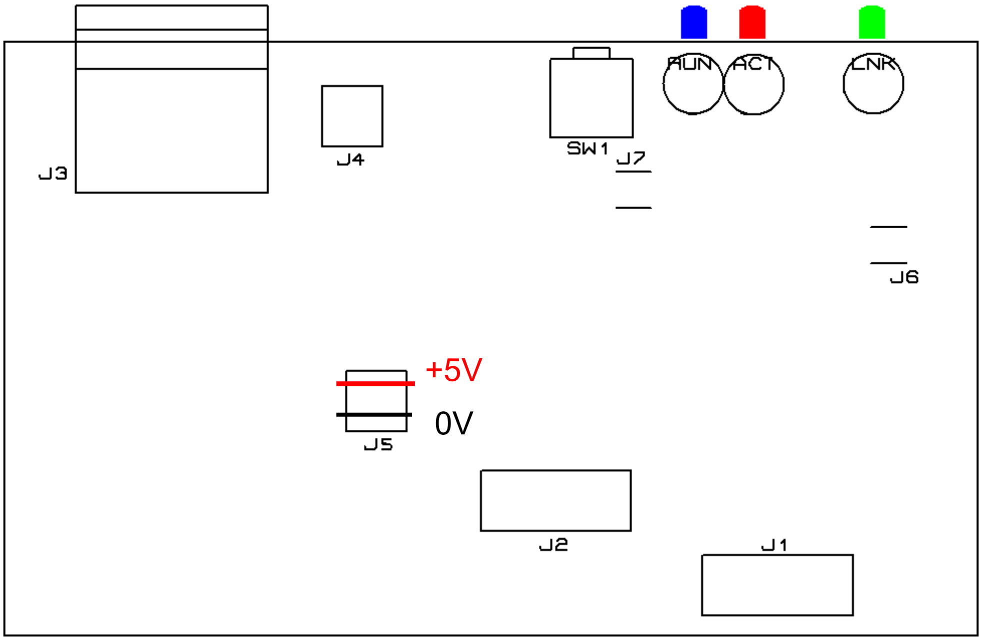
The network CPU can be used both in MIDI and HD setups. Change from MIDI to HD (and back) is simply done by updating the board firmware.

Note : access to HD firmware is restricted to MMA HDWG members until official publication of HD protocol specification.

2 - CPU characteristics

- Dual core 32 bits CPU running at 400MHz (option : 500MHz) with DSP capabilities, giving an impressive 800 MIPS computing power (1000 MIPS for 500MHz option)
- 16 real-time tasks with 10ns task switching resolution (the number of real-time tasks available for end user depends on the firmware being run)
- Fully configurable I/O ports, with FPGA-like capabilities (support most communication protocols like SPI, UART, CAN, I^{2}S, etc...)
- Two high speed serial links (100Mbps) for multiprocessor communication
- Supports "hard real-time" network protocols like AVB
- 10/100 Mbps Ethernet with MDIX and auto-negotiation support
- 128 Kb on-chip fast SRAM for code and data
- 4 Mbits Flash for bootloader, file system and firmware storage (option : 8 Mbits Flash)
- PoE option (via daughterboard)
- Encrypted bootloader to protect intellectual property of custom firmware for network upgrade
- Eight 3.3V I/O ports with two 5V tolerant input lines
- Power supply : 5V (provided by application board or by PoE option board)
- Power consumption : typically 150mA under 5V (without application board)

3 - Interface signals



3.1 - J2 – Host port

The J2 connector is used to interconnect V3 CPU board to the host system. This connector can also be used to provide power supply to the CPU board when J5 is not installed.

All lines use 3.3V signaling levels.

WARNING : the lines on J2 are **not** 5V tolerant. Using 5V TTL voltage standard is not allowed to connect the V3 CPU to the host.

Connector is a 2x5 pins header, 0.1" spacing. Cable length between J2 and the host system boards shall be kept to the shortest possible length. If the host board is located more than 10 cms away from the V3 CPU board, a buffer module must be installed between J2 and the host board. For long cables between hosts and the V3 CPU, serial resistors of 33R may be needed to avoid signal ringing on the lines. Contact Kiss-Box for details about this particular setup.

Communication with host performed using a SPI synchronous link. The V3 CPU can be either SPI Master or SPI Slave, depending on the firmware loaded in the CPU.

The table hereafter list the different signals used for the SPI communication.

3.1.1 - Signals for SPI Master communication

Pin#	Function	Direction	Description
1	I/O 1	Input	Reserved for future use. A 4.7k resistor pull-down resistor is recommended on host board to hold this line in inactive state by default.
2	I/O 2	Input	Reserved for future use. A 4.7k resistor pull-down resistor is recommended on host board to hold this line in inactive state by default.
3	+5V		+5V power supply from host. Can be used to provide +5V to the CPU if user does not want to use J5. Voltage on this line **must** be kept under 5.25V under any circumstance. A voltage greater than 5.25V applied to this line even for a short time can destroy the V3 CPU board.
4	GND		0V connection for power-supply
5	I/O 3	Input	Reserved for future use. A 4.7k resistor pull-down resistor is recommended on host board to hold this line in inactive state by default
6	/SPICS	Output	SPI Chip Select – Active low
7	SCLK	Output	SPI Serial Clock
8	/HRA	Input	Host Requires Attention input – Active low When this input is set to logical "0", the V3 CPU will initiate a transfer with the host, even if there is no data to transmit to the host. If this line is not used, it must be pulled to logical "1" by a pull-up resistor.
9	SDO	Output	Serial Data Output. Data from V3 CPU to host.
10	SDI	Input	Serial Data Input. Data from host to V3 CPU.

3.1.2 - Signals for SPI Slave communication

Pin#	Function	Direction	Description
1	I/O 1	Input	Reserved for future use. A 4.7k resistor pull-down resistor is recommended on host board to hold this line in inactive state by default.
2	I/O 2	Input	Reserved for future use. A 4.7k resistor pull-down resistor is recommended on host board to hold this line in inactive state by default.
3	+5V		+5V power supply from host. Can be used to provide +5V to the CPU if user does not want to use J5. Voltage on this line **must** be kept under 5.25V under any circumstance. A voltage greater than 5.25V applied to this line even for a short time can destroy the V3 CPU board.
4	GND		0V connection for power-supply
5	I/O 3	Input	Reserved for future use. A 4.7k resistor pull-down resistor is recommended on host board to hold this line in inactive state by default
6	/SPICS	Input	SPI Chip Select – Active low
7	SCLK	Input	SPI Serial Clock.
8	I/O 4	Input	Reserved for future use. A 4.7k resistor pull-down resistor is recommended on host board to hold this line in inactive state by default.
9	SDO	Output	Serial Data Output. Data from V3 CPU to host.
10	SDI	Input	Serial Data Input. Data from host to V3 CPU.

3.2 - J5 – 5V power supply

Used to provide 5V power supply to the CPU board, from PoE daughterboard or from external power supply if PoE option is not installed.

WARNING : the power supply voltage shall ALWAYS be kept between 4.75V and 5.25V

The board is not guaranteed to work properly if voltage drops under 4.75V

The board and/or the application board connected to J2 can be damaged/destroyed if voltage applied to J5 exceeds 5.25V, even for a short time.

Connector is a 2x2 pins header, 0.1" spacing.

Pin#	Function	Description
1	+5V	+5V power supply
2	+5V	+5V power supply
3	GND	0V
4	GND	0V

3.3 - Reserved connectors

Connectors J1, J4, J6 and J7 are reserved for special uses and future expansions. They shall never be connected to host board.

4 - SPI Protocol Configuration

SPI interfaces have multiple configuration parameters:

- SPI Master / SPI Slave
- SPI Clock Speed
- SPI Direction (MSB transmitted first or LSB transmitted first)
- SPI Mode (CPOL and CPHA active states)

These parameters must be configured at firmware level and can not be configured over the SPI interface itself. They are statically configured at compilation time for now.

SPI Master firmware uses SPI Mode 0 (CPOL=0 / CPHA=0), 32 bits, MSB transmitted first. Speed defined in configuration profile. Contact KissBox for details about clock frequency selection.

SPI Slave firmware uses SPI Mode 3 (CPOL=1 / CPHA=1), 32 bits, MSB transmitted first. Maximum clock speed=25MHz. Recommended clock speed when using standard flat-cable between V3 CPU and host is between 10 to 15MHz to avoid communication errors due to signal ringing (cable inductance)

Note that these two configurations are ideal for communication with Analog Device's SHARC DSP family. If you intend to use the V3 CPU in association with a SHARC based host, KissBox can provide details about the DSP configuration to use.

Contact KissBox if you need specific communication profile (LSB first, SPI Modes 1 and 2, etc...)

5 - MIDI 1.0 host communication protocol

MIDI 1.0 host communication protocol is based on MMA Specification for USB communication.

MIDI data is carried in the packet in 32 bit MIDI Event. Most common MIDI messages are 2 or 3 bytes packed into one MIDI Event. Longer messages, generally System Exclusive messages are carried in multiple MIDI Events. These MIDI Event provide a method to transfer MIDI messages with 32 bit fixed length messages to help memory allocation. This also makes parsing MIDI events easier by packetizing the separate bytes of a MIDI message into one parsed MIDI Event.

The first byte in each 32-bit MIDI Event is a sub-header containing a Session Index Number (4 bits) followed by a Code Index Number (4 bits). The remaining three bytes contain the actual MIDI event. Most typical parsed MIDI events are two or three bytes in length. Unused bytes are reserved and must be padded with zeros (in the case of a one- or two-byte MIDI event) to preserve the 32-bit fixed length of the MIDI Event.

5.1 - Session Index Number

Session Index Number is used to identify the RTP-MIDI session handler providing MIDI data. Multiple MIDI streams can then use the same SPI link.

5.2 - Code Index Number

The Code Index Number (CIN) indicates the classification of the bytes in the DATA_X fields and the number of bytes in the message. The following table summarizes these classifications.

CIN	DATA_X size	Description
0x0	1, 2 or 3	Miscellaneous function codes. Reserved for future extensions.
0x1	1, 2 or 3	Cable events. Reserved for future expansion.
0x2	2	Two-byte System Common messages like MTC, SongSelect, etc.
0x3	3	Three-byte System Common messages like SPP, etc.
0x4	3	SysEx starts or continues
0x5	1	Single-byte System Common Message or SysEx ends with following single byte
0x6	2	SysEx ends with following two bytes.
0x7	3	SysEx ends with following three bytes.
0x8	3	Note-off
0x9	3	Note-on
0xA	3	Poly-KeyPressure
0xB	3	Control Change
0xC	2	Program Change
0xD	2	Channel Pressure
0xE	3	PitchBend Change
0xF	1	Single-byte message

5.3 - DATA_1, DATA_2, DATA_3: MIDI Message Data

MIDI data bytes are transmitted in DATA_1, DATA_2 and DATA_3 exactly in the same format as the MIDI 1.0 specification requires.

Running status is never used, so all the messages are formed of all bytes.

5.4 - 32 bits messages examples

All examples hereafter are given with Session Index Number = 0

5.4.1 - Timing clock

MIDI message = 0xF8

SPI message = 0x0FF80000

5.4.2 - Note On

MIDI message = 0x90 0x2A 0x40

SPI message = 0x09902A40

5.4.3 - System Exclusive

MIDI message = 0xF0 0x01 0x02 0x03 0x04 0x05 0xF7

SPI messages =	0x04F00102	(SYSEX start)
	0x04030405	(SYSEX continues)
	0x05F70000	(SYSEX ends with one byte)

5.5 - "Stuffing" messages

Since the SPI link exchanges input and output data at the same time, MIDI data will not be available from both side most of the time.

For example, if the HRA line is activated, it means that the host has MIDI data to send. Most of the time, there will be no data in the network buffer of the KissBox CPU (otherwise, a SPI transfer would have been initiated if the CPU is Master)

In the other direction, when a MIDI message is being received from the network, it's highly probable that the host has no data to send.

In order for the receiver (on each side) to detect that there was no MIDI data transmitted in one direction, two reserved values are used to report the lack of data:

- 0xFFFFFFFF
- 0x00000000

The KissBox CPU will send one of these values when a SPI transfer is initiated and no data are available in its buffers. The host shall reject the 32 bits message received from the KissBox CPU if it contains one of these values.

In the other direction, when the KissBox network CPU receives this value from the host, it will recognize the reserved value as "No data available". The data is then ignored and not transmitted over the network.

6 - HD host communication protocol

HD communication protocol is based on 32 bits words, making a complete HD message a multiple of 32 bits.

Compared to MIDI 1.0, HD does not have a specific marker to identify the first word of a message. In order to allow the host CPU to synchronize properly on any new incoming HD message, HD firmware for V3 CPU uses a dedicated synchronization protocol.

Each complete HD message transmitted to or from the V3 CPU is preceded by a synchronization word containing value 0xFFFFFFFF. This word shall not be interpreted by the host as a part of HD message.

Details for HD communication protocol are restricted for now to MMA HD Working Group members until official release of specification.

If you want to implement HD communication protocol in your host, please contact Kiss-Box to provide your HDWG identity. We will then provide the details of the HD communication protocol under NDA.

7 - Document revisions

Date	Auteur	Version	Description
10/12/2011	B.Bouchez	1.0	First draft
20/02/2014	B.Bouchez	2.0	Updated for public release

Legal notice: this document and all information contained within are the property of KissBox. It is not allowed to share or copy this document and the informations it contains without a previous written agreement from KissBox. No liability or responsibility is accepted by KissBox for any direct or indirect damage, loss, accident, expense, or any other claim arising from the use of the informations provided by the document and/or KissBox related product.

Prepared with OpenOffice Writer software.

For more information about OpenOffice, go to http://www.openoffice.org.