1. General Information

EUROCOM/USB converter board allows connecting any equipment with Eurocom interface to any equipment (computer) with USB host interface.

EUROCOM/USB converter board specification:

1. Binary throughput: 256..2048 kbps

2. Eurocom interface D1/86
   - B type, as per EUROCOM D/1 1986 IB6 (pages IB6-1...IB6-5)
   - Physical layer only
   - Binary throughput 256, 512, 1024, or 2048kbps programmable), Eurocom D/1 and Eurocom-EES
   - Source clock mode:
     - Internal source
     - Loop (received clock)
   - Internal source clock stability: ±1 ppm
   - Connector: D-SUB 9 female-type

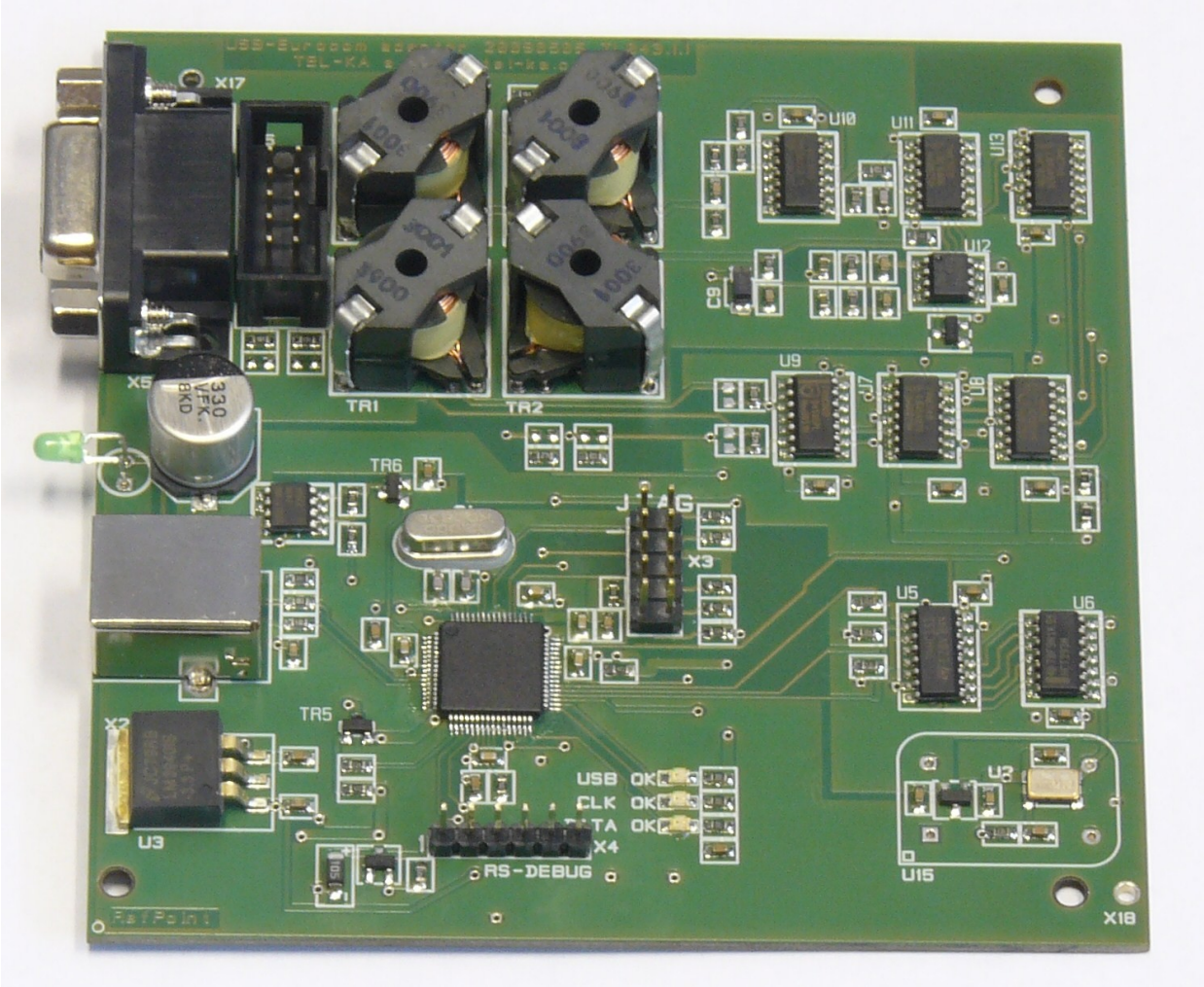
3. USB:
   - USB 1.1 full-speed (12Mbit/s) operation
   - Isochronous transfer mode
   - High Powered function, power consumption < 250 mA

4. Power supply from USB Interface

5. Board size: 100mm • 100mm • 18mm

6. Software:
   1. Linux device driver included
   2. Windows device driver available on request

2. Board layout



Board contains:

EUROCOM (DSUB-9) connector
EUROCOM (flat cable FC-10) connector
USB “B” interface
'Power' status indicator (LED)
'USB OK' status indicators (SMD-LED)
'CLK OK' status indicator (SMD-LED)
'DATA OK' status indicator (SMD-LED)
JTAG connectors
Serial debug connector (RS-DEBUG)

3. Eurocom interface B, DSUB-9 female

FC 10	Pin DB9	Signal Name Eurocom B	Direction	RL-xxx TDM connector	AN/GRC-xxx Eurocom In/Out
1	1	Transmit Clock A	Output from converter	L	F
2	6	Transmit Clock B		M	E
3	2	Transmit Data A		H	H
4	7	Transmit Data B		G	G
5	3	Receive Clock A	Input to converter	K	S
6	8	Receive Clock B		J	R
7	4	Receive Data A		F	T
8	9	Receive Data B		E	J
9 10	5	GND	Chassis	V	V

1. RL-xxx : Ericsson/Konsberg
2. AN/GRC-xxx : Marconi/Ultra

4. Signaling on board

1. 'Power' LED (between Eurocom and USB interfaces) indicates USB interface power state. Power is enabled if the host accepts device descriptors and sends 'set_configuration' command. In practice this happens if device driver is installed.
2. 'USB OK.' LED indicates correct transmission between the converter and host via USB 0 interface
3. 'CLK OK' LED indicates correct frequency of clock received on Eurocom interface (256 kHz or 512 kHz or 1024 kHz or 2048 kHz).
4. 'DATA OK' LED indicates data receipt on Eurocom interface. It is enabled when received data rate is greater than 1/256 of clock frequency.

5. Configuration

Converter board software allows bandwidth reservation on USB interface, and selection of transmission frequency. If the transmitted or received clock frequency is too high in comparison to reserved bandwidth, the transmission via USB interface is stopped.

1. USB interface bandwidth reservation
   USB interface bandwidth reservation for transmission (with any of following throughputs: 256, 512, 1024, 2048kbps) is executed by issuing standard SET_INTERFACE command (USB Specification 2.0 ch. 9.4 Standard Device Request), bmRequestType = 0x01. 'Alternate setting' value set to 0 means no reservation, values from 1 to 4 mean respectively throughputs from 256 to 2048 kbps.

2. Setting transmitter clock
   Transmitter clock is set by executing SET_INTERFACE command with following parameters bmRequest=0x21 (class, interface), bRequest=0x2F. 'Value' field selects frequency:
   0 – 256 kHz, 1 – 512 kHz, 2 – 1024 kHz, 3 – 2048kHz,
   4 – OFF, 5 – receiver clock, 7 – OFF

3. Setting transmitter clock correction
   The device allows to correct the transmitter clock (as per CFPT-9301 generator documentation ±5ppm). Generators voltage is controlled by SET_INTERFACE command with following parameters: bmRequest=0x21 (class, interface), bRequest=0x2E. 16 bit 'Value' field holds voltage value. 0x00 means correction disabled, 0x01 is correction enabled and 0V set, 0xffff means 3.3V).

6. Linux Driver

Linux device driver (for kernel 2.6.xx) is distributed along with converter board. The driver allows transmission between Eurocom interface and host memory using standard IO operations (open/close/read/write) and to configure converter board using ioctl function. After connection, driver creates a device file /dev/eurocom***x*** where ***x*** is the device number. To start sending / receiving data, program should open the driver using *open* function and set initial board configuration using *ioctl,* and start data exchange using *write/read* functions. Data is transmitted/received LSB first. We recommend using separate threads for read / write operations. A test program is attached to driver, it is BER meter for measurement text 1+x^14+x^15.

ioctl commands selection (see EurocomUsbIoctl.h):

1. EURUSBA_SET_CONFIG
   Command sets the device work mode, depending on argument value (uint32_t)

0x0000 – device turned off
0x0001 – 256 kbit/s, internal TxClock
0x0002 – 512 kbit/s, internal TxClock
0x0003 – 1024 kbit/s, internal TxClock
0x0004 – 2048 kbit/s, internal TxClock
0x0011 – 256 kbit/s, TxClock from RxClock
0x0012 – 512 kbit/s, TxClock from RxClock
0x0013 – 1024 kbit/s, TxClock from RxClock
0x0014 – 2048 kbit/s, TxClock from RxClock
other values are not allowed and will not cause state change

2. EURUSBA_SET_IF_CONFIG
   Command selects reserved bandwidth depending on argument value (uint8_t)
   0x00 – no bandwidth reserved
   0x01 –256 kbps
   0x02 –512 kbps
   0x03 –1024 kbps
   0x04 –2048 kbps

3. EURUSBA_SET_HW_CONFIG
   Command sets transmitter clock source depending on argument (uint8_t):
   0x00 – 256 kHz
   0x01 – 512 kHz
   0x02 – 1024 kHz
   0x03 – 2048 kHz
   0x04 - OFF
   0x05 – RxClk
   0x06 – 4096kHz (only for testing, should not be used)
   0x07 - OFF

7. **USB descriptors**:

```
Device Descriptor:
bcdUSB:               0x0101
bDeviceClass:         0x02
bDeviceSubClass:      0x00
bDeviceProtocol:      0x00
bMaxPacketSize0:      0x40 (64)
idVendor:             0xFFFF
idProduct:            0xFFF8
bcdDevice:            0x0100
iManufacturer:        0x01
iProduct:             0x02
iSerialNumber:        0x03
bNumConfigurations:   0x01

Configuration Descriptor:
bNumInterfaces        0x02
bConfigurationValue   0x01
bmAttributes          0xc0
bMaxPower             125      //250mA

Interface 0 Descriptor
bInterfaceNumber      0x00
bAlternateSetting     0x00
bNumEndpoints         0x01
bInterfaceClass       0x02
bInterfaceSubClass    0x82
bInterfaceProtocol    0xff

Endpoint Descriptor
bEndpointAddress      0x81
bmAttributes          0x03
wMaxPacketSize        0x10
bInterval             0x10     //16ms



Interface 1 Descriptor (0)
bInterfaceNumber      0x01
bAlternateSetting     0x00
bNumEndpoints         0x00
bInterfaceClass       0x0A
bInterfaceSubClass    0x00
bInterfaceProtocol    0x00



Interface 1 Descriptor (1)
bInterfaceNumber      0x01
bAlternateSetting     0x01
bNumEndpoints         0x02
bInterfaceClass       0x0A
bInterfaceSubClass    0x00
bInterfaceProtocol    0x00
```

Endpoint Descriptor

bEndpointAddress	0x03	
bmAttributes	0x05	
wMaxPacketSize	0x24	
bInterval	0x01	//1ms

Endpoint Descriptor

bEndpointAddress	0x86	
bmAttributes	0x25	
wMaxPacketSize	0x24	
bInterval	0x01	//1ms

Interface 1 Descriptor (2)

bInterfaceNumber	0x01
bAlternateSetting	0x02
bNumEndpoints	0x02
bInterfaceClass	0x0A
bInterfaceSubClass	0x00
bInterfaceProtocol	0x00

Endpoint Descriptor

bEndpointAddress	0x03	
bmAttributes	0x05	
wMaxPacketSize	0x44	
bInterval	0x01	//1ms

Endpoint Descriptor

bEndpointAddress	0x86	
bmAttributes	0x25	
wMaxPacketSize	0x44	
bInterval	0x01	//1ms

Interface 1 Descriptor (3)

bInterfaceNumber	0x01
bAlternateSetting	0x03
bNumEndpoints	0x02
bInterfaceClass	0x0A
bInterfaceSubClass	0x00
bInterfaceProtocol	0x00

Endpoint Descriptor

bEndpointAddress	0x03	
bmAttributes	0x05	
wMaxPacketSize	0x84	
bInterval	0x01	//1ms

Endpoint Descriptor

bEndpointAddress	0x86	
bmAttributes	0x25	
wMaxPacketSize	0x84	
bInterval	0x01	//1ms

Interface 1 Descriptor (4)

```
bInterfaceNumber       0x01
bAlternateSetting      0x04
bNumEndpoints          0x02
bInterfaceClass        0x0A
bInterfaceSubClass     0x00
bInterfaceProtocol     0x00
```

Endpoint Descriptor

```
bEndpointAddress       0x03
bmAttributes           0x05
wMaxPacketSize         0x104
bInterval              0x01      //1ms
```

Endpoint Descriptor

```
bEndpointAddress       0x86
bmAttributes           0x25
wMaxPacketSize         0x104
bInterval              0x01      //1ms
```