2245 LVDS-TTL

Features

- 16 LVDS-TTL channels.
- Input- and output-capable
- No galvanic isolation
- High speed and low jitter
- RJ45 connectors

Applications

- Photon counting
- External equipment trigger
- Optical shutter control
- Serial communication with remote devices

General Description

The 2245 LVDS-TTL card is a 4hp EEM module. It adds general-purpose digital I/O capabilities to carrier cards such as 1124 Kasli and 1125 Kasli-SoC.

Each card provides sixteen total digital channels, with four RJ45 connectors in the front panel, controlled through 2 EEM connectors. Each RJ45 connector exposes four LVDS digital channels. Each individual EEM connector controls eight channels independently. Single EEM operation is possible. The direction (input or output) of each channel can be selected individually using DIP switches.

Outputs are intended to drive 100Ω loads and inputs are 100Ω terminated. This card can achieve higher speed and lower jitter than the isolated 2118/2128 BNC/SMA-TTL cards. Only shielded Ethernet Cat-6 cables should be connected.

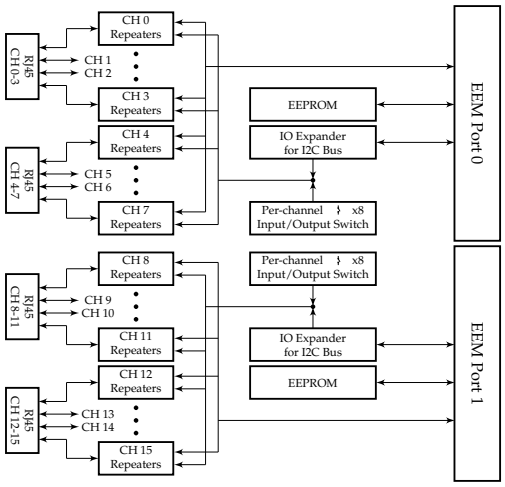


Figure 1: Simplified Block Diagram

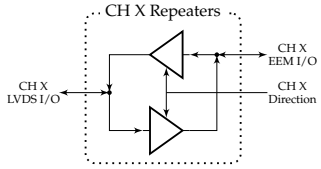


Figure 2: Detailed diagram for channel repeaters

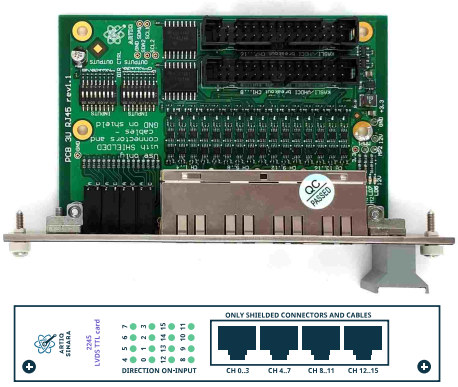


Figure 3: LVDS-TTL card and front panel

Source

2245 LVDS-TTL, like all the Sinara hardware family, is open-source hardware, and design files (schematics, PCB layouts, BOMs) can be found in detail at the repository https://github.com/sinara-hw/DIO_LVDS_RJ45/wiki.

Electrical Specifications

All specifications are in $-40°C \leq T_A \leq 85°C$ unless otherwise noted. Information in this section is based on the datasheet of the repeater IC (FIN1101K8X[1]).

Table 1: Recommended Input Voltage

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions
Magnitude of differential input	$\|V_{ID}\|$	0.1		3.3	V	
Common mode input	V_{IC}	$\|V_{ID}\|/2$		$3.3 - \|V_{ID}\|/2$	V	
Differential input threshold HIGH	V_{TH}			100	mV	
Differential input threshold LOW	V_{TL}	-100			mV	

All typical values of DC specifications are at $T_A = 25°C$.

Table 2: DC Specifications

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions
Output differential voltage	V_{OD}	250	330	450	mV	With 100Ω load.
$\|V_{OD}\|$ change (LOW-to-HIGH)	ΔV_{OD}			25	mV	
Offset voltage	V_{OS}	1.125	1.23	1.375	V	
$\|V_{OS}\|$ change (LOW-to-HIGH)	ΔV_{OS}			25	mV	
Short circuit output current	I_{OS}		± 3.4	± 6	mA	
Input current	I_{IN}			± 20	µA	Recommended input voltage

All typical values of AC specifications are at $T_A = 25°C$, $V_{ID} = 300mV$, $V_{IC} = 1.3V$ unless otherwise given.

[1] https://www.onsemi.com/pdf/datasheet/fin1101-d.pdf

Table 3: AC Specifications

Parameter	Min.	Typ.	Max.	Unit	Conditions
Differential output rise time (20% to 80%)	0.29	0.40	0.58	ns	Duty cycle = 50%.
Differential output fall time (80% to 20%)	0.29	0.40	0.58	ns	
Pulse width distortion		0.01	0.2	ns	
LVDS data jitter, deterministic		85	125	ps	$PRBS = 2^{23} - 1$ 800 Mbps
LVDS clock jitter, random (RMS)		2.1	3.5	ps	400 MHz clock

Configuring IO Direction & Termination

The IO direction of each channel can be configured by DIP switches, which are found at the top of the card.

- IO direction switch closed (`ON`)
  Fixes the corresponding bank to output. The IO direction cannot be changed by I^2C.
- IO direction switch open (OFF)
  The corresponding bank is set to input by default. IO direction *can* be changed by I^2C.

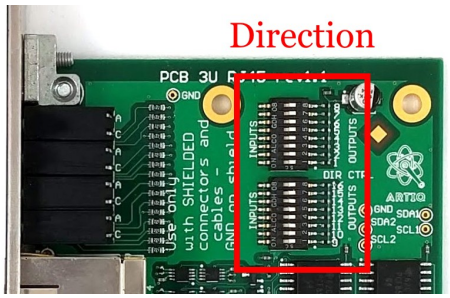


Figure 4: Position of switches

Example ARTIQ Code

The sections below demonstrate simple usage scenarios of extensions on the ARTIQ control system. These extensions make use of the resources of the 2245 LVDS-TTL card. They do not exhaustively demonstrate all the features of the ARTIQ system.

The full documentation for ARTIQ software and gateware, including guides for their use, is available at https://m-labs.hk/artiq/manual/. Please consult the manual for details and reference material of the functions and structures used here.

Timing accuracy in these examples is well under 1 nanosecond thanks to ARTIQ RTIO infrastructure.

One pulse per second

The channel should be configured as output in both gateware and hardware.

```
@kernel
def run(self):
    self.core.reset()
    while True:
        self.ttl0.pulse(500*ms)
        delay(500*ms)
```

Morse code

This example demonstrates some basic algorithmic features of the ARTIQ-Python language.

```
def prepare(self):
    # As of ARTIQ-6, the ARTIQ compiler has limited string handling
    # capabilities, so we pass a list of integers instead.
    message = ".- .-. - .. --.-"
    self.commands = [{".": 1, "-": 2, " ": 3}[c] for c in message]

@kernel
def run(self):
    self.core.reset()
    for cmd in self.commands:
        if cmd == 1:
            self.led.pulse(100*ms)
            delay(100*ms)
        if cmd == 2:
            self.led.pulse(300*ms)
            delay(100*ms)
        if cmd == 3:
            delay(700*ms)
```

Counting rising edges in a 1ms window

The channel should be configured as input in both gateware and hardware.

```
@kernel
def run(self):
    self.core.reset()
    gate_end_mu = self.ttl0.gate_rising(1*ms)
    counts = self.ttl0.count(gate_end_mu)
    print(counts)
```

This example code uses the software counter, which has a maximum count rate of approximately 1 million events per second. If the gateware counter is enabled on the TTL channel, it can typically count up to 125 million events per second:

```
@kernel
def run(self):
    self.core.reset()
    delay(6*ns)              # Coarse RTIO period: 0 - 7  ns
    self.ttl0.pulse(3*ns)    # Coarse RTIO period: 8 - 15 ns
```

Responding to an external trigger

One channel needs to be configured as input, and the other as output.

```
    self.core.reset()
    self.ttl0.set(62.5*MHz)
```

SPI Master Device

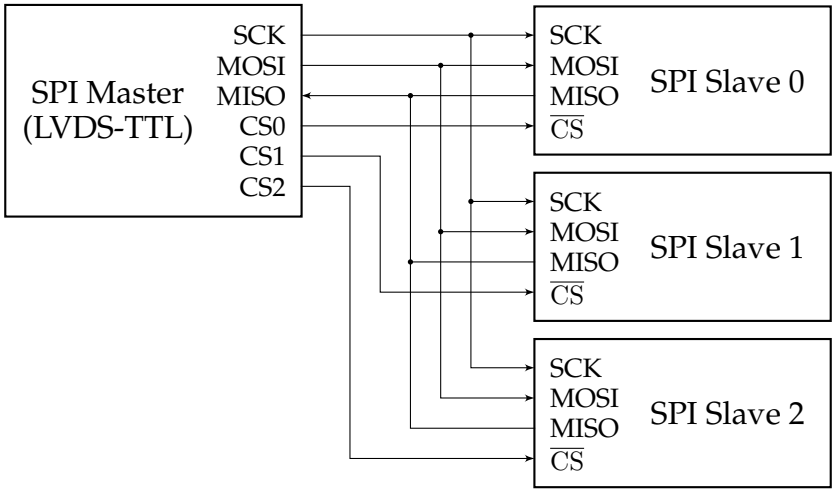
If one of the two card EEM ports is configured as `dio_spi` instead of `dio`, its associated TTL channels can be configured as SPI master devices. Invocation of an SPI transfer follows this pattern:

1. Set the `config` register (e.g. using `set_config_mu()`).
2. Start the SPI transfer by writing the `data` register using `write()`.
3. If the data from the SPI slave is to be read (i.e. `SPI_INPUT` was set in `config`), invoke `read()` to read.

The list of configurations supported in the gateware are listed as below:

Flag	Description
`SPI_OFFLINE`	Switch all pins to high impedance mode.
`SPI_END`	Next SPI transfer is the last of the transcation.
`SPI_INPUT`	Submit SPI read data as RTIO input event when the transfer is complete.
`SPI_CS_POLARITY`	Active level of chip select (CS)
`SPI_CLK_POLARITY`	Idle level of serial clock (SCK)
`SPI_CLK_PHASE`	Data is sampled on falling edge & shifted out on rising edge.
`SPI_LSB_FIRST`	LSB is the first on bit on the wire.
`SPI_HALF_DUPLEX`	Use 3-wire SPI, where MOSI is both input & output capable.

The following ARTIQ example demonstrates the flow of an SPI transaction on a typical SPI setup with 3 homogeneous slaves. The direction switches on the LVDS-TTL card should be set to the correct IO direction for all relevant channels before powering on.



SPI Configuration

The following examples will assume the SPI communication has the following properties:

- Chip select (CS) is active low
- Serial clock (SCK) idle level is low
- Data is sampled on rising edge of SCK & shifted out on falling edge of SCK
- Most significant bit (MSB) first
- Full duplex

The baseline configuration for an `SPIMaster` instance can be defined as such:

```
from artiq.coredevice import spi2 as spi


SPI_CONFIG = (0 * spi.SPI_OFFLINE | 0 * spi.SPI_END |
              0 * spi.SPI_INPUT | 0 * spi.SPI_CS_POLARITY |
              0 * spi.SPI_CLK_POLARITY | 0 * spi.SPI_CLK_PHASE |
              0 * spi.SPI_LSB_FIRST | 0 * spi.SPI_HALF_DUPLEX)
```

The `SPI_END` & `SPI_INPUT` flags will be modified during runtime in the following example.

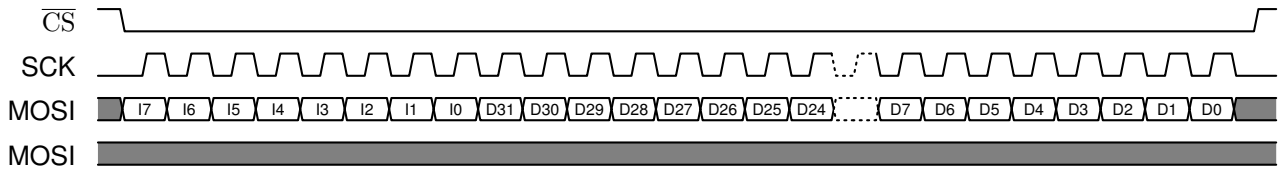
SPI frequency

Frequency of the SPI clock must be the result of RTIO clock frequency divided by an integer factor in [2, 257]. In the following examples, the SPI frequency will be set to 1 MHz by dividing the RTIO frequency (125 MHz) by 125.

```
CLK_DIV = 125
```

SPI write

Typically, an SPI write operation involves sending an instruction and data to the SPI slaves. Suppose the instruction and data are 8 bits and 32 bits respectively. The timing diagram of such a write operation is shown in the following:

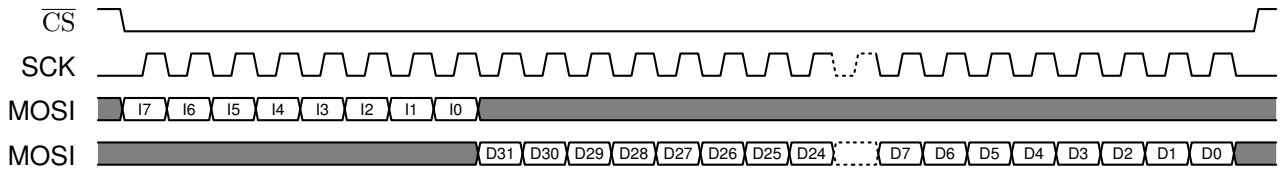


Suppose the instruction is `0x13`, while the data is `0xDEADBEEF`. In addition, both slave 1 & 2 are selected. This SPI transaction can be performed with the following code:

```
@kernel
def run(self):
    self.core.reset()
    self.spi.set_config_mu(SPI_CONFIG, 8, CLK_DIV, 0b110)
    self.spi.write(0x13 << 24)      # Shift the bits to the MSBs.
                                    # Since SPI_LSB_FIRST is NOT set,
                                    # SPI Machine will shift out bits from
                                    # the MSB of the `data` register.`
    self.spi.set_config_mu(SPI_CONFIG | spi.SPI_END, 32, CLK_DIV, 0b110)
    self.spi.write(0xDEADBEEF)
```

SPI read

A 32-bit read is represented by the following timing diagram:



Suppose the instruction is `0x81`, where only slave 0 is selected. This SPI transcation can be performed by the following code.

```
@kernel
def run(self):
    self.core.reset()
    self.spi.set_config_mu(SPI_CONFIG, 8, CLK_DIV, 0b001)
    self.spi.write(0x81 << 24)      # Shift the bits to the MSBs.
                                    # Since SPI_LSB_FIRST is NOT set,
                                    # SPI Machine will shift out bits from
                                    # the MSB of the `data` register.`
    self.spi.set_config_mu(SPI_CONFIG | spi.SPI_END | spi.SPI_INPUT,
                           32, CLK_DIV, 0b001)
    self.spi.write(0)               # write() performs the SPI transfer.
                                    # As suggested by the timing diagram,
                                    # the exact value of this argument
                                    # does not matter.
    print(self.spi.read())
```

Ordering Information

To order, please visit https://m-labs.hk and choose 2245 LVDS-TTL in the ARTIQ/Sinara hardware selection tool. Cards can be ordered as part of a fully-featured ARTIQ/Sinara crate or standalone through the 'Spare cards' option. Otherwise, orders can also be made by writing directly to mailto:sales@m-labs.hk.