

Welcome to [E-XFL.COM](https://www.e-xfl.com)

### What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

### Applications of "[Embedded - Microcontrollers](#)"

#### Details

Product Status	Obsolete
Core Processor	XCore
Core Size	32-Bit 24-Core
Speed	4000MIPS
Connectivity	USB
Peripherals	-
Number of I/O	176
Program Memory Size	2MB (2M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1M x 8
Voltage - Supply (Vcc/Vdd)	0.95V ~ 3.6V
Data Converters	-
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	374-LFBGA
Supplier Device Package	374-FBGA (18x18)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/xmos/xuf224-1024-fb374-c40">https://www.e-xfl.com/product-detail/xmos/xuf224-1024-fb374-c40</a>

# PRELIMINARY

- ▶ **Ports** The I/O pins are connected to the processing cores by Hardware Response ports. The port logic can drive its pins high and low, or it can sample the value on its pins optionally waiting for a particular condition. Section [6.3](#)
- ▶ **Clock blocks** xCORE devices include a set of programmable clock blocks that can be used to govern the rate at which ports execute. Section [6.4](#)
- ▶ **Memory** Each xCORE Tile integrates a bank of SRAM for instructions and data, and a block of one-time programmable (OTP) memory that can be configured for system wide security features. Section [9](#)
- ▶ **PLL** The PLL is used to create a high-speed processor clock given a low speed external oscillator. Section [7](#)
- ▶ **USB** The USB PHY provides High-Speed and Full-Speed, device, host, and on-the-go functionality. Data is communicated through ports on the digital node. A library is provided to implement USB device functionality. Section [10](#)
- ▶ **Flash** The device has a built-in 4MBflash. Section [8](#)
- ▶ **JTAG** The JTAG module can be used for loading programs, boundary scan testing, in-circuit source-level debugging and programming the OTP memory. Section [11](#)

## 1.1 Software

Devices are programmed using C, C++ or xC (C with multicore extensions). XMOS provides tested and proven software libraries, which allow you to quickly add interface and processor functionality such as USB, Ethernet, PWM, graphics driver, and audio EQ to your applications.

## 1.2 xTIMEcomposer Studio

The xTIMEcomposer Studio development environment provides all the tools you need to write and debug your programs, profile your application, and write images into flash memory or OTP memory on the device. Because xCORE devices operate deterministically, they can be simulated like hardware within xTIMEcomposer: uniquely in the embedded world, xTIMEcomposer Studio therefore includes a static timing analyzer, cycle-accurate simulator, and high-speed in-circuit instrumentation.

xTIMEcomposer can be driven from either a graphical development environment, or the command line. The tools are supported on Windows, Linux and MacOS X and available at no cost from [xmos.com/downloads](https://www.xmos.com/downloads). Information on using the tools is provided in the xTIMEcomposer User Guide, [X3766](#).

## 2 XUF224-1024-FB374 Features

### ► Multicore Microcontroller with Advanced Multi-Core RISC Architecture

- 24 real-time logical cores on 4 xCORE tiles
- Cores share up to 2000 MIPS
  - Up to 4000 MIPS in dual issue mode
- Each logical core has:
  - Guaranteed throughput of between  $\frac{1}{5}$  and  $\frac{1}{6}$  of tile MIPS
  - 16x32bit dedicated registers
- 167 high-density 16/32-bit instructions
  - All have single clock-cycle execution (except for divide)
  - 32x32→64-bit MAC instructions for DSP, arithmetic and user-definable cryptographic functions

### ► USB PHY, fully compliant with USB 2.0 specification

### ► Programmable I/O

- 176 general-purpose I/O pins, configurable as input or output
  - Up to 59 x 1bit port, 22 x 4bit port, 15 x 8bit port, 8 x 16bit port, 4 x 32bit port
  - 10 xCONNECT links
- Port sampling rates of up to 60 MHz with respect to an external clock
- 64 channel ends for communication with other cores, on or off-chip

### ► Memory

- 1024KB internal single-cycle SRAM (max 256KB per tile) for code and data storage
- 32KB internal OTP (max 8KB per tile) for application boot code
- 4MB internal flash for application code and overlays

### ► Hardware resources

- 24 clock blocks (6 per tile)
- 40 timers (10 per tile)
- 16 locks (4 per tile)

### ► JTAG Module for On-Chip Debug

### ► Security Features

- Programming lock disables debug and prevents read-back of memory contents
- AES bootloader ensures secrecy of IP held on external flash memory

### ► Ambient Temperature Range

- Commercial qualification: 0°C to 70°C
- Industrial qualification: -40°C to 85°C

### ► Speed Grade

- 40: 2000 MIPS

### ► Power Consumption

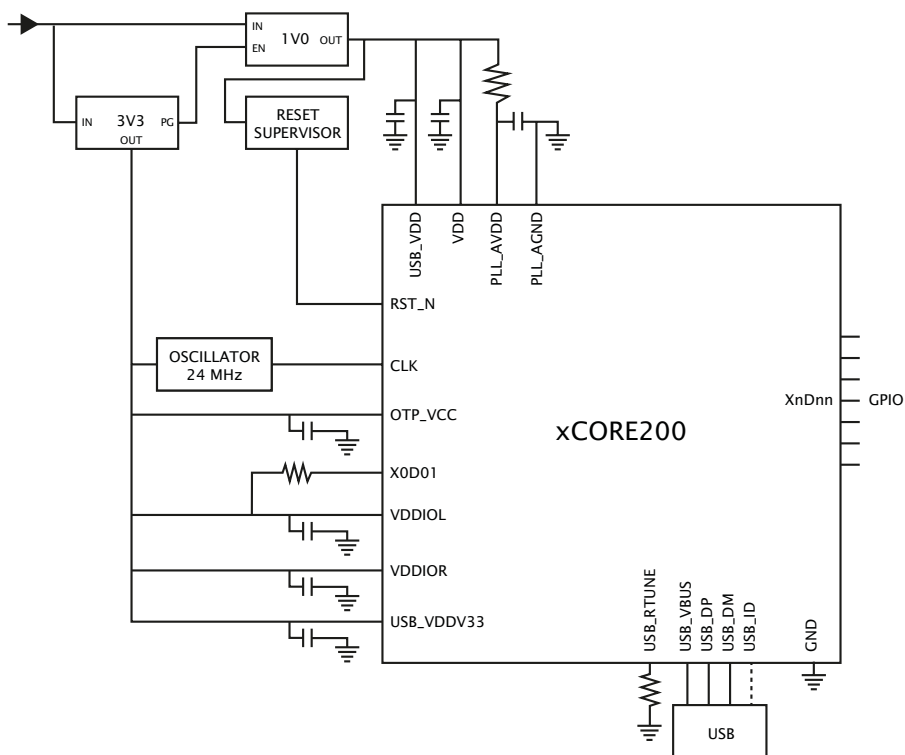
- 1140 mA (typical)

### ► 374-pin FBGA package 0.8 mm pitch

Signal	Function	Type	Properties
X2D00	1A <sup>0</sup>	I/O	IOL, PD
X2D02	4A <sup>0</sup> 8A <sup>0</sup> 16A <sup>0</sup> 32A <sup>20</sup>	I/O	IOL, PD
X2D03	4A <sup>1</sup> 8A <sup>1</sup> 16A <sup>1</sup> 32A <sup>21</sup>	I/O	IOL, PD
X2D04	4B <sup>0</sup> 8A <sup>2</sup> 16A <sup>2</sup> 32A <sup>22</sup>	I/O	IOL, PD
X2D05	4B <sup>1</sup> 8A <sup>3</sup> 16A <sup>3</sup> 32A <sup>23</sup>	I/O	IOL, PD
X2D06	4B <sup>2</sup> 8A <sup>4</sup> 16A <sup>4</sup> 32A <sup>24</sup>	I/O	IOL, PD
X2D07	4B <sup>3</sup> 8A <sup>5</sup> 16A <sup>5</sup> 32A <sup>25</sup>	I/O	IOL, PD
X2D08	4A <sup>2</sup> 8A <sup>6</sup> 16A <sup>6</sup> 32A <sup>26</sup>	I/O	IOL, PD
X2D09	4A <sup>3</sup> 8A <sup>7</sup> 16A <sup>7</sup> 32A <sup>27</sup>	I/O	IOL, PD
X2D11	1D <sup>0</sup>	I/O	IOL, PD
X2D12	1E <sup>0</sup>	I/O	IOR, PD
X2D13	1F <sup>0</sup>	I/O	IOR, PD
X2D14	4C <sup>0</sup> 8B <sup>0</sup> 16A <sup>8</sup> 32A <sup>28</sup>	I/O	IOR, PD
X2D15	4C <sup>1</sup> 8B <sup>1</sup> 16A <sup>9</sup> 32A <sup>29</sup>	I/O	IOR, PD
X2D16	XL4 <sup>4</sup> <sub>in</sub> 4D <sup>0</sup> 8B <sup>2</sup> 16A <sup>10</sup>	I/O	IOR, PD
X2D17	XL4 <sup>3</sup> <sub>in</sub> 4D <sup>1</sup> 8B <sup>3</sup> 16A <sup>11</sup>	I/O	IOR, PD
X2D18	XL4 <sup>2</sup> <sub>in</sub> 4D <sup>2</sup> 8B <sup>4</sup> 16A <sup>12</sup>	I/O	IOR, PD
X2D19	XL4 <sup>1</sup> <sub>in</sub> 4D <sup>3</sup> 8B <sup>5</sup> 16A <sup>13</sup>	I/O	IOR, PD
X2D20	4C <sup>2</sup> 8B <sup>6</sup> 16A <sup>14</sup> 32A <sup>30</sup>	I/O	IOR, PD
X2D21	4C <sup>3</sup> 8B <sup>7</sup> 16A <sup>15</sup> 32A <sup>31</sup>	I/O	IOR, PD
X2D22	1G <sup>0</sup>	I/O	IOR, PD
X2D23	1H <sup>0</sup>	I/O	IOR, PD
X2D24	XL7 <sup>0</sup> <sub>in</sub> 1I <sup>0</sup>	I/O	IOR, PD
X2D25	XL7 <sup>0</sup> <sub>out</sub> 1J <sup>0</sup>	I/O	IOR, PD
X2D26	XL7 <sup>3</sup> <sub>out</sub> 4E <sup>0</sup> 8C <sup>0</sup> 16B <sup>0</sup>	I/O	IOR, PD
X2D27	XL7 <sup>4</sup> <sub>out</sub> 4E <sup>1</sup> 8C <sup>1</sup> 16B <sup>1</sup>	I/O	IOR, PD
X2D28	4F <sup>0</sup> 8C <sup>2</sup> 16B <sup>2</sup>	I/O	IOR, PD
X2D29	4F <sup>1</sup> 8C <sup>3</sup> 16B <sup>3</sup>	I/O	IOR, PD
X2D30	4F <sup>2</sup> 8C <sup>4</sup> 16B <sup>4</sup>	I/O	IOR, PD
X2D31	4F <sup>3</sup> 8C <sup>5</sup> 16B <sup>5</sup>	I/O	IOR, PD
X2D32	4E <sup>2</sup> 8C <sup>6</sup> 16B <sup>6</sup>	I/O	IOR, PD
X2D33	4E <sup>3</sup> 8C <sup>7</sup> 16B <sup>7</sup>	I/O	IOR, PD
X2D34	XL7 <sup>1</sup> <sub>out</sub> 1K <sup>0</sup>	I/O	IOR, PD
X2D35	XL7 <sup>2</sup> <sub>out</sub> 1L <sup>0</sup>	I/O	IOR, PD
X2D36	1M <sup>0</sup> 8D <sup>0</sup> 16B <sup>8</sup>	I/O	IOL, PD
X2D49	XL5 <sup>4</sup> <sub>in</sub> 32A <sup>0</sup>	I/O	IOR, PD
X2D50	XL5 <sup>3</sup> <sub>in</sub> 32A <sup>1</sup>	I/O	IOR, PD
X2D51	XL5 <sup>2</sup> <sub>in</sub> 32A <sup>2</sup>	I/O	IOR, PD
X2D52	XL5 <sup>1</sup> <sub>in</sub> 32A <sup>3</sup>	I/O	IOR, PD
X2D53	XL5 <sup>0</sup> <sub>in</sub> 32A <sup>4</sup>	I/O	IOR, PD
X2D54	XL5 <sup>0</sup> <sub>out</sub> 32A <sup>5</sup>	I/O	IOR, PD
X2D55	XL5 <sup>1</sup> <sub>out</sub> 32A <sup>6</sup>	I/O	IOR, PD
X2D56	XL5 <sup>2</sup> <sub>out</sub> 32A <sup>7</sup>	I/O	IOR, PD

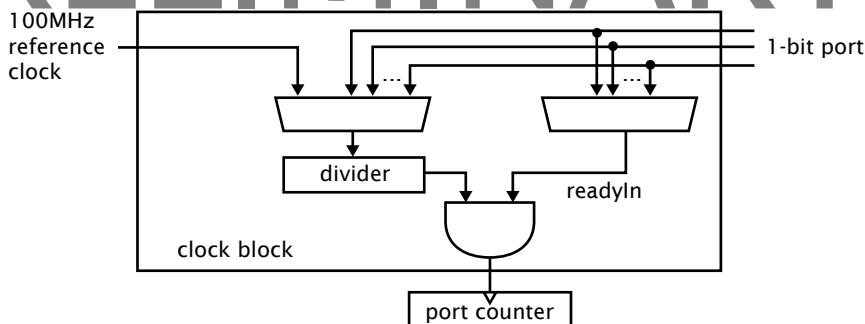
(continued)

## 5 Example Application Diagram



**Figure 2:**  
Simplified  
Reference  
Schematic

PRELIMINARY



**Figure 5:**  
Clock block  
diagram

A clock block can use a 1-bit port as its clock source allowing external application clocks to be used to drive the input and output interfaces. xCORE-200 clock blocks optionally divide the clock input from a 1-bit port.

In many cases I/O signals are accompanied by strobing signals. The xCORE ports can input and interpret strobe (known as readyIn and readyOut) signals generated by external sources, and ports can generate strobe signals to accompany output data.

On reset, each port is connected to clock block 0, which runs from the xCORE Tile reference clock.

## 6.5 Channels and Channel Ends

Logical cores communicate using point-to-point connections, formed between two channel ends. A channel-end is a resource on an xCORE tile, that is allocated by the program. Each channel-end has a unique system-wide identifier that comprises a unique number and their tile identifier. Data is transmitted to a channel-end by an output-instruction; and the other side executes an input-instruction. Data can be passed synchronously or asynchronously between the channel ends.

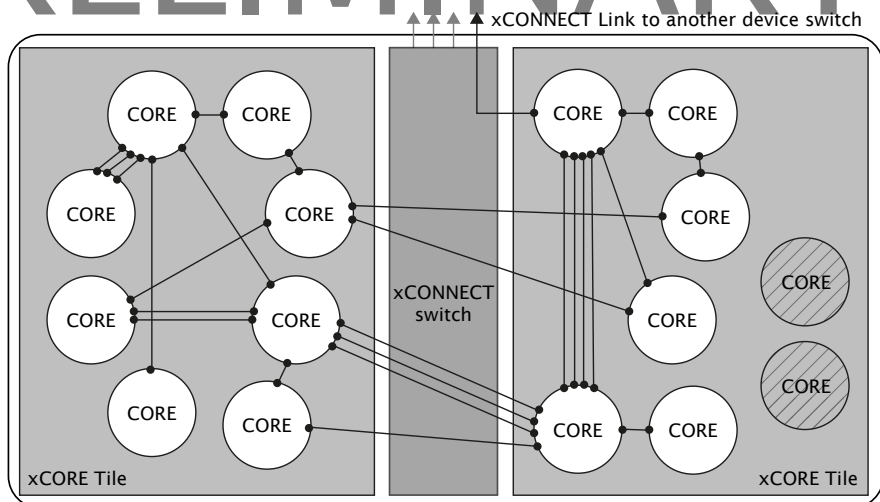
## 6.6 xCONNECT Switch and Links

XMOS devices provide a scalable architecture, where multiple xCORE devices can be connected together to form one system. Each xCORE device has an xCONNECT interconnect that provides a communication infrastructure for all tasks that run on the various xCORE tiles on the system.

The interconnect relies on a collection of switches and XMOS links. Each xCORE device has an on-chip switch that can set up circuits or route data. The switches are connected by xConnect Links. An XMOS link provides a physical connection between two switches. The switch has a routing algorithm that supports many different topologies, including lines, meshes, trees, and hypercubes.

The links operate in either 2 wires per direction or 5 wires per direction mode, depending on the amount of bandwidth required. Circuit switched, streaming

PRELIMINARY



**Figure 6:**  
Switch, links  
and channel  
ends

and packet switched data can both be supported efficiently. Streams provide the fastest possible data rates between xCORE Tiles (up to 250 MBit/s), but each stream requires a single link to be reserved between switches on two tiles. All packet communications can be multiplexed onto a single link.

Information on the supported routing topologies that can be used to connect multiple devices together can be found in the XS1-UF Link Performance and Design Guide, [X2999](#).

## 7 PLL

The PLL creates a high-speed clock that is used for the switch, tile, and reference clock. The PLL multiplication value is selected through the two MODE pins, and can be changed by software to speed up the tile or use less power. The MODE pins are set as shown in Figure 7:

Oscillator Frequency	MODE		Tile Frequency	PLL Ratio	PLL settings		
	1	0			OD	F	R
3.25-10 MHz	0	0	130-400 MHz	40	1	159	0
9-25 MHz	1	1	144-400 MHz	16	1	63	0
25-50 MHz	1	0	167-400 MHz	8	1	31	0
50-100 MHz	0	1	196-400 MHz	4	1	15	0

**Figure 7:**  
PLL multiplier  
values and  
MODE pins

PRELIMINARY

Feature	Bit	Description
Disable JTAG	0	The JTAG interface is disabled, making it impossible for the tile state or memory content to be accessed via the JTAG interface.
Disable Link access	1	Other tiles are forbidden access to the processor state via the system switch. Disabling both JTAG and Link access transforms an xCORE Tile into a “secure island” with other tiles free for non-secure user application code.
Secure Boot	5	The xCORE Tile is forced to boot from address 0 of the OTP, allowing the xCORE Tile boot ROM to be bypassed (see §8).
Redundant rows	7	Enables redundant rows in OTP.
Sector Lock 0	8	Disable programming of OTP sector 0.
Sector Lock 1	9	Disable programming of OTP sector 1.
Sector Lock 2	10	Disable programming of OTP sector 2.
Sector Lock 3	11	Disable programming of OTP sector 3.
OTP Master Lock	12	Disable OTP programming completely: disables updates to all sectors and security register.
Disable JTAG-OTP	13	Disable all (read & write) access from the JTAG interface to this OTP.
	21..15	General purpose software accessible security register available to end-users.
	31..22	General purpose user programmable JTAG UserID code extension.

**Figure 10:**  
Security  
register  
features

The OTP memory is programmed using three special I/O ports: the OTP address port is a 16-bit port with resource ID 0x100200, the OTP data is written via a 32-bit port with resource ID 0x200100, and the OTP control is on a 16-bit port with ID 0x100300. Programming is performed through `libotp` and `xburn`.

## 9.2 SRAM

Each xCORE Tile integrates a single 256KBSRAM bank for both instructions and data. All internal memory is 32 bits wide, and instructions are either 16-bit or 32-bit. Byte (8-bit), half-word (16-bit) or word (32-bit) accesses are supported and are executed within one tile clock cycle. There is no dedicated external memory interface, although data memory can be expanded through appropriate use of the ports.

## 10 USB PHY

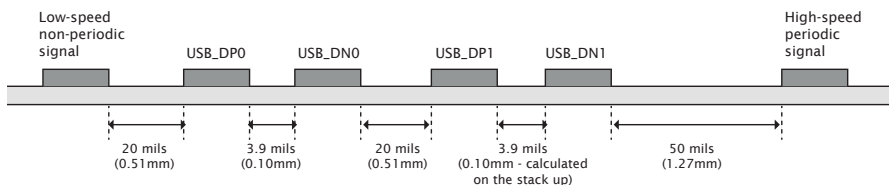
The USB PHY provides High-Speed and Full-Speed, device, host, and on-the-go functionality. The PHY is configured through a set of peripheral registers (Appendix F),



# PRELIMINARY

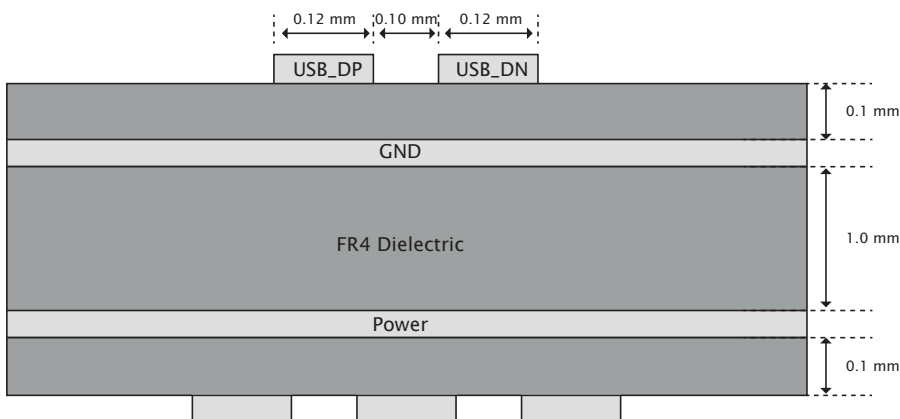
board traces for USB\_DP and USB\_DN are tightly matched. In addition, according to the USB 2.0 specification, the USB\_DP and USB\_DN differential impedance must be  $90\ \Omega$ .

**Figure 15:**  
USB trace separation showing a low speed signal, two differential pairs and a high-speed clock



## 12.2.1 General routing and placement guidelines

The following guidelines will help to avoid signal quality and EMI problems on high speed USB designs. They relate to a four-layer (Signal, GND, Power, Signal) PCB.



**Figure 16:**  
Example USB board stack

For best results, most of the routing should be done on the top layer (assuming the USB connector and XS2-UF24A-1024-FB374 are on the top layer) closest to GND. Reference planes should be below the transmission lines in order to maintain control of the trace impedance.

We recommend that the high-speed clock and high-speed USB differential pairs are routed first before any other routing. When routing high speed USB signals, the following guidelines should be followed:

- High speed differential pairs should be routed together.
- High-speed USB signal pair traces should be trace-length matched. Maximum trace-length mismatch should be no greater than 4mm.

### 13.9 JTAG Timing

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
f(TCK_D)	TCK frequency (debug)			18	MHz	
f(TCK_B)	TCK frequency (boundary scan)			10	MHz	
T(SETUP)	TDO to TCK setup time	5			ns	A
T(HOLD)	TDO to TCK hold time	5			ns	A
T(DELAY)	TCK to output delay			15	ns	B

**Figure 25:**  
JTAG timing

A Timing applies to TMS and TDI inputs.

B Timing applies to TDO output from negative edge of TCK.

All JTAG operations are synchronous to TCK.

control-token 192	24-bit response channel-end identifier	16-bit register number	32-bit data	control-token 1
----------------------	---	---------------------------	----------------	--------------------

The response to a write message comprises either control tokens 3 and 1 (for success), or control tokens 4 and 1 (for failure).

A read message comprises the following:

control-token 193	24-bit response channel-end identifier	16-bit register number	control-token 1
----------------------	---	---------------------------	--------------------

The response to the read message comprises either control token 3, 32-bit of data, and control-token 1 (for success), or control tokens 4 and 1 (for failure).

### A.3 Accessing node configuration

Node configuration registers can be accessed through the interconnect using the functions `write_node_config_reg(device, ...)` and `read_node_config_reg(device, ...)`, where `device` is the name of the node. These functions implement the protocols described below.

Instead of using the functions above, a channel-end can be allocated to communicate with the node configuration registers. The destination of the channel-end should be set to `0xnnnnC30C` where `nnnn` is the node-identifier.

A write message comprises the following:

control-token 192	24-bit response channel-end identifier	16-bit register number	32-bit data	control-token 1
----------------------	---	---------------------------	----------------	--------------------

The response to a write message comprises either control tokens 3 and 1 (for success), or control tokens 4 and 1 (for failure).

A read message comprises the following:

control-token 193	24-bit response channel-end identifier	16-bit register number	control-token 1
----------------------	---	---------------------------	--------------------

The response to a read message comprises either control token 3, 32-bit of data, and control-token 1 (for success), or control tokens 4 and 1 (for failure).

### A.4 Accessing a register of an analogue peripheral

Peripheral registers can be accessed through the interconnect using the functions `write_periph_32(device, peripheral, ...)`, `read_periph_32(device, peripheral, ...)`, `write_periph_8(device, peripheral, ...)`, and `read_periph_8(device, peripheral, ...)`; where `device` is the name of the analogue device, and `peripheral` is the number of the peripheral. These functions implement the protocols described below.

A channel-end should be allocated to communicate with the configuration registers. The destination of the channel-end should be set to `0xnnnnpp02` where `nnnn` is the node-identifier and `pp` is the peripheral identifier.

# PRELIMINARY

A write message comprises the following:

control-token 36	24-bit response channel-end identifier	8-bit register number	8-bit size	data	control-token 1
---------------------	---	--------------------------	---------------	------	--------------------

The response to a write message comprises either control tokens 3 and 1 (for success), or control tokens 4 and 1 (for failure).

A read message comprises the following:

control-token 37	24-bit response channel-end identifier	8-bit register number	8-bit size	control-token 1
---------------------	---	--------------------------	---------------	--------------------

The response to the read message comprises either control token 3, data, and control-token 1 (for success), or control tokens 4 and 1 (for failure).

**B.18 Debug interrupt data: 0x16**

On a data watchpoint, this register contains the effective address of the memory operation that triggered the debugger. On a resource watchpoint, it contains the resource identifier.

**0x16:**  
Debug  
interrupt data

Bits	Perm	Init	Description
31:0	DRW		Value.

**B.19 Debug core control: 0x18**

This register enables the debugger to temporarily disable logical cores. When returning from the debug interrupts, the cores set in this register will not execute. This enables single stepping to be implemented.

**0x18:**  
Debug core  
control

Bits	Perm	Init	Description
31:8	RO	-	Reserved
7:0	DRW		1-hot vector defining which threads are stopped when not in debug mode. Every bit which is set prevents the respective thread from running.

**B.20 Debug scratch: 0x20 .. 0x27**

A set of registers used by the debug ROM to communicate with an external debugger, for example over JTAG. This is the same set of registers as the [Debug Scratch registers in the xCORE tile configuration](#).

**0x20 .. 0x27:**  
Debug  
scratch

Bits	Perm	Init	Description
31:0	DRW		Value.

**B.21 Instruction breakpoint address: 0x30 .. 0x33**

This register contains the address of the instruction breakpoint. If the PC matches this address, then a debug interrupt will be taken. There are four instruction breakpoints that are controlled individually.

## C Tile Configuration

The xCORE Tile control registers can be accessed using configuration reads and writes (use `write_tile_config_reg(tileref, ...)` and `read_tile_config_reg(tileref, ...)` for reads and writes).

Number	Perm	Description
0x00	CRO	Device identification
0x01	CRO	xCORE Tile description 1
0x02	CRO	xCORE Tile description 2
0x04	CRW	Control PSwitch permissions to debug registers
0x05	CRW	Cause debug interrupts
0x06	CRW	xCORE Tile clock divider
0x07	CRO	Security configuration
0x20 .. 0x27	CRW	Debug scratch
0x40	CRO	PC of logical core 0
0x41	CRO	PC of logical core 1
0x42	CRO	PC of logical core 2
0x43	CRO	PC of logical core 3
0x44	CRO	PC of logical core 4
0x45	CRO	PC of logical core 5
0x46	CRO	PC of logical core 6
0x47	CRO	PC of logical core 7
0x60	CRO	SR of logical core 0
0x61	CRO	SR of logical core 1
0x62	CRO	SR of logical core 2
0x63	CRO	SR of logical core 3
0x64	CRO	SR of logical core 4
0x65	CRO	SR of logical core 5
0x66	CRO	SR of logical core 6
0x67	CRO	SR of logical core 7

**Figure 30:**  
Summary

### C.1 Device identification: 0x00

This register identifies the xCORE Tile

**0x41:**  
PC of logical  
core 1

Bits	Perm	Init	Description
31:0	CRO		Value.

### C.11 PC of logical core 2: 0x42

Value of the PC of logical core 2.

**0x42:**  
PC of logical  
core 2

Bits	Perm	Init	Description
31:0	CRO		Value.

### C.12 PC of logical core 3: 0x43

Value of the PC of logical core 3.

**0x43:**  
PC of logical  
core 3

Bits	Perm	Init	Description
31:0	CRO		Value.

### C.13 PC of logical core 4: 0x44

Value of the PC of logical core 4.

**0x44:**  
PC of logical  
core 4

Bits	Perm	Init	Description
31:0	CRO		Value.

### C.14 PC of logical core 5: 0x45

Value of the PC of logical core 5.

**0x45:**  
PC of logical  
core 5

Bits	Perm	Init	Description
31:0	CRO		Value.

## C.24 SR of logical core 7: 0x67

Value of the SR of logical core 7

**0x67:**  
SR of logical  
core 7

Bits	Perm	Init	Description
31:0	CRO		Value.



PRELIMINARY

**D.8 System JTAG device ID register: 0x09**

<b>0x09:</b> System JTAG device ID register	Bits	Perm	Init	Description
	31:28	RO		
	27:12	RO		
	11:1	RO		
	0	RO		

**D.9 System USERCODE register: 0x0A**

<b>0x0A:</b> System USERCODE register	Bits	Perm	Init	Description
	31:18	RO		JTAG USERCODE value programmed into OTP SR
	17:0	RO		metal fixable ID code

**D.10 Directions 0-7: 0x0C**

This register contains eight directions, for packets with a mismatch in bits 7..0 of the node-identifier. The direction in which a packet will be routed is governed by the most significant mismatching bit.

<b>0x0C:</b> Directions 0-7	Bits	Perm	Init	Description
	31:28	RW	0	The direction for packets whose dimension is 7.
	27:24	RW	0	The direction for packets whose dimension is 6.
	23:20	RW	0	The direction for packets whose dimension is 5.
	19:16	RW	0	The direction for packets whose dimension is 4.
	15:12	RW	0	The direction for packets whose dimension is 3.
	11:8	RW	0	The direction for packets whose dimension is 2.
	7:4	RW	0	The direction for packets whose dimension is 1.
	3:0	RW	0	The direction for packets whose dimension is 0.

**D.11 Directions 8-15: 0x0D**

This register contains eight directions, for packets with a mismatch in bits 15..8 of the node-identifier. The direction in which a packet will be routed is governed by the most significant mismatching bit.

PRELIMINARY

**0x1F:**  
Debug source

Bits	Perm	Init	Description
31:5	RO	-	Reserved
4	RW		Reserved.
3:2	RO	-	Reserved
1	RW		If set, XCore1 is the source of last GlobalDebug event.
0	RW		If set, XCore0 is the source of last GlobalDebug event.

### D.15 Link status, direction, and network: 0x20 .. 0x28

These registers contain status information for low level debugging (read-only), the network number that each link belongs to, and the direction that each link is part of. The registers control links 0..7.

**0x20 .. 0x28:**  
Link status,  
direction, and  
network

Bits	Perm	Init	Description
31:26	RO	-	Reserved
25:24	RO		Identify the SRC_TARGET type 0 - SLink, 1 - PLink, 2 - SSCTL, 3 - Undefined.
23:16	RO		When the link is in use, this is the destination link number to which all packets are sent.
15:12	RO	-	Reserved
11:8	RW	0	The direction that this link operates in.
7:6	RO	-	Reserved
5:4	RW	0	Determines the network to which this link belongs, reset as 0.
3	RO	-	Reserved
2	RO		1 when the current packet is considered junk and will be thrown away.
1	RO		1 when the dest side of the link is in use.
0	RO		1 when the source side of the link is in use.

### D.16 PLink status and network: 0x40 .. 0x47

These registers contain status information and the network number that each processor-link belongs to.

## F USB PHY Configuration

The USB PHY is connected to the ports shown in section 10.

The *USB PHY* is peripheral 1. The control registers are accessed using 32-bit reads and writes (use `write_periph_32(device, 1, ...)` and `read_periph_32(device, ↵ 1, ...)` for reads and writes).

Number	Perm	Description
0x00	WO	UIFM reset
0x04	RW	UIFM IFM control
0x08	RW	UIFM Device Address
0x0C	RW	UIFM functional control
0x10	RW	UIFM on-the-go control
0x14	RO	UIFM on-the-go flags
0x18	RW	UIFM Serial Control
0x1C	RW	UIFM signal flags
0x20	RW	UIFM Sticky flags
0x24	RW	UIFM port masks
0x28	RW	UIFM SOF value
0x2C	RO	UIFM PID
0x30	RO	UIFM Endpoint
0x34	RW	UIFM Endpoint match
0x38	RW	OTG Flags mask
0x3C	RW	UIFM power signalling
0x40	RW	UIFM PHY control

**Figure 33:**  
Summary

### F.1 UIFM reset: 0x00

A write to this register with any data resets all UIFM state, but does not otherwise affect the phy.

**0x00:**  
UIFM reset

Bits	Perm	Init	Description
31:0	WO		Value.

### F.2 UIFM IFM control: 0x04

General settings of the UIFM IFM state machine.

**0x04:**  
UIFM IFM  
control

Bits	Perm	Init	Description
31:8	RO	-	Reserved
7	RW	0	Set to 1 to enable XEVACKMODE mode.
6	RW	0	Set to 1 to enable SOFISTOKEN mode.
5	RW	0	Set to 1 to enable UIFM power signalling mode.
4	RW	0	Set to 1 to enable IF timing mode.
3	RO	-	Reserved
2	RW	0	Set to 1 to enable UIFM linestate decoder.
1	RW	0	Set to 1 to enable UIFM CHECKTOKENS mode.
0	RW	0	Set to 1 to enable UIFM DOTOKENS mode.

### F.3 UIFM Device Address: 0x08

The device address whose packets should be received. 0 until enumeration, it should be set to the assigned value after enumeration.

**0x08:**  
UIFM Device  
Address

Bits	Perm	Init	Description
31:7	RO	-	Reserved
6:0	RW	0	The enumerated USB device address must be stored here. Only packets to this address are passed on.

### F.4 UIFM functional control: 0x0C

**0x0C:**  
UIFM  
functional  
control

Bits	Perm	Init	Description
31:5	RO	-	Reserved
4:2	RW	1	Set to 0 to disable UIFM to UTMI+ OPMODE mode.
1	RW	1	Set to 1 to switch UIFM to UTMI+ TERMSELECT mode.
0	RW	1	Set to 1 to switch UIFM to UTMI+ XCVRSELECT mode.

### F.5 UIFM on-the-go control: 0x10

This register is used to negotiate an on-the-go connection.

PRELIMINARY

---

**0x2C:**  
 UIFM PID
 

---

Bits	Perm	Init	Description
31:4	RO	-	Reserved
3:0	RO	0	Value of the last received PID.

### F.13 UIFM Endpoint: 0x30

The last endpoint seen

---

**0x30:**  
 UIFM  
 Endpoint
 

---

Bits	Perm	Init	Description
31:5	RO	-	Reserved
4	RO	0	1 if endpoint contains a valid value.
3:0	RO	0	A copy of the last received endpoint.

### F.14 UIFM Endpoint match: 0x34

This register can be used to mark UIFM endpoints as special.

---

**0x34:**  
 UIFM  
 Endpoint  
 match
 

---

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RW	0	This register contains a bit for each endpoint. If its bit is set, the endpoint will be supplied on the RX port when ORed with 0x10.

### F.15 OTG Flags mask: 0x38

---

**0x38:**  
 OTG Flags  
 mask
 

---

Bits	Perm	Init	Description
31:0	RW	0	Data

### F.16 UIFM power signalling: 0x3C

---

**0x3C:**  
 UIFM power  
 signalling
 

---

Bits	Perm	Init	Description
31:9	RO	-	Reserved
8	RW	0	Valid
7:0	RW	0	Data