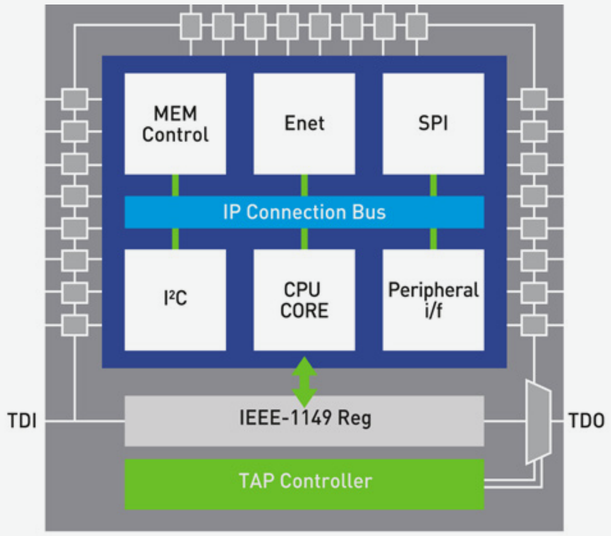
CORECOMMANDER
FOR MICROPROCESSORS AND MICROCONTROLLERS

Direct access to memory and peripheral devices (I/O)
for testing, debugging and in-system programming



- Direct access to memory and peripheral (I/O) devices of a microprocessor through its (JTAG) debug interface
- Read data from, write data to memory and peripherals without software programming
- At-speed execution of read and write cycles
- Testing and debugging of the connectivity of processor memory and peripherals with at-speed bus cycles without software programming
- Easy programming of processor flash memory without software programming

Corecommander provides high-level functions to write data to and read data from microprocessor memory and I/O addresses without software programming. CoreCommander functions are applied via the JTAG interface.

Applications

CoreCommander is used in design debug, manufacturing test and (field) service for many different applications such as:

- Diagnosing "dead-kernel" boards; no embedded code is required to perform memory reads and writes.
- Determining the right settings for the peripheral controller (DDR controller, flash memory controller, I/O controller) in combination with your particular memory or peripheral device. Write settings into the controller registers and verify whether proper access to memory is possible with those settings.
- Checking the proper connectivity of a memory or I/O device by writing data to it and reading data back from it.
- Programming board (serial nr) specific data such as calibration values, a mac address or a timestamp in flash memory, or program an entire flash.

Order information
CoreComm Micro (core)
(core) = ARM 7, ARM 9, ARM 11, Cortex-A, Cortex-R, Cortex-M, Blackfin, PXA2xx, PXA3xx, IXP4xx, PowerPC-MPC500 family, PowerPC-MPC5500 family, PowerPC-MPC5600 family, C28x, XC166, Tricore, PIC32

[1] if the uProcessor also contains a boundary-scan register then teh tests and in-system programming operations can also be done using the boundary-scan register instead of the CoreCommander. Whether in that case the CoreCommander or the boundary-scan register is used depends on preference or performance.

Background

A uP performs read and write operations on its bus to access memory and I/O locations. The read and write cycles normally result when the uP executes a program that is stored in memory. This requires a good functioning interface between uP and its memory and a program stored in memory.

Execution of a stored program is less logical when one wants to test or debug the connections between a uP and its (external) memory. Another method as provided by CoreCommander is then needed. With CoreCommander the uP is not executing a program. Instead it is externally instructed to write data to and read data from memory and I/O addresses specified with the CoreCommander commands.

CORECOMMANDER
FOR MICROPROCESSORS AND MICROCONTROLLERS

Direct access to memory and peripheral devices (I/O)
for testing, debugging and in-system programming

Using CoreCommander the coding to test memory connections or to program a flash memory becomes a simple and straightforward sequence of write and read commands.

Usage

CoreCommander can be used as interactive **hardware debug tool** via its high-level GUI. In this interface register access commands or full memory reads and writes can be selected for inspection and control and executed with a direct view of the results. Sequences of commands can be re-played within the interactive window or exported into a Python editor. The interactive usage is particularly valuable during hardware bring up and debugging in design and (field) service. For **automated scripts** the functions from CoreCommander can be called directly from programming (scripting) environments such as Python, LabView, LabWindows, Visual basic, C, C++, .NET and TestStand. This is highly valuable to create re-usable tests for specific devices or clusters and for in-system flash programming.

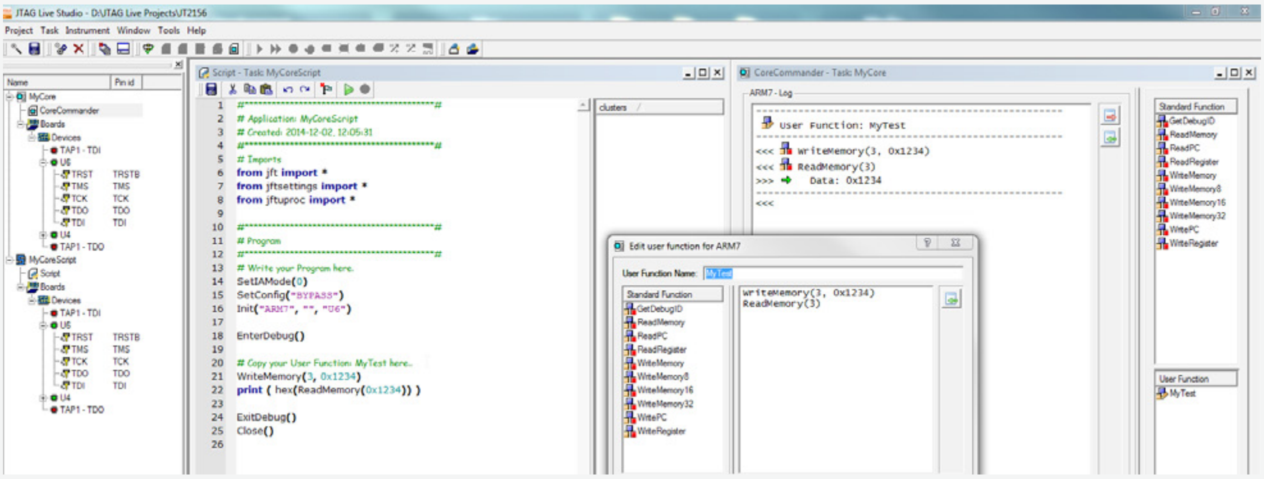
- Available for µProcessors, µControllers and DSP's using one or more of the following cores:

Analog Devices:	Blackfin
Arm:	ARM7, ARM9, ARM11, Cortex-A, Cortex-R, Cortex-M
Freescale:	PowerPC MPC500, MPC5500, and MPC 5600 families
Infineon:	C166, Tricore
Microchip:	PIC32
Texas Instruments:	C28x (TMS320)
Xscale:	IXP4xx, PXA2xx, PXA3xx

- Ask your JTAG supplier for the support for your specific µProcessor or µController.

Note: a CoreCommander is core specific, not device specific. Hence any µController that uses one of the above cores is supported through these CoreCommanders.

Figure 2



```
#****************************************#
# Application: MyCoreScript
# Created: 2014-12-02, 12:05:31
#****************************************#
# Imports
from jft import *
from jftsettings import *
from jftuproc import *

#****************************************#
# Program
#****************************************#
# Write your Program here.
SetIAMode(0)
SetConfig("BYPASS")
Init("ARM7", "", "U6")

EnterDebug()

# Copy your User Function: MyTest here..
WriteMemory(3, 0x1234)
print ( hex(ReadMemory(0x1234)) )

ExitDebug()
Close()
```

```
------------------------------------------------------------
 User Function: MyTest
------------------------------------------------------------
<<<  WriteMemory(3, 0x1234)
<<<  ReadMemory(3)
>>>    Data: 0x1234
------------------------------------------------------------
<<<
```

User Function Name: MyTest

```
WriteMemory(3, 0x1234)
ReadMemory(3)
```

CoreCommander GUI and Python code

Region or Country	Telephone	E-mail
• Europe or Rest of World	+31 (0)40 295 0870	info@jtag.nl
• United Kingdom & Ireland	+44 (0)1234 831212	sales@jtag.co.uk
• North America	Toll free - 877 FOR JTAG Western US - 949 454 9040	info@jtag.com
• China, Malaysia, Singapore, Thailand, Taiwan	+86 (021) 5831 1577	info@jtag.com.cn
• Germany	+49 (0)971 6991064	germany@jtag.com
• Finland	+358 (0)9 4730 2670	finland@jtag.com
• Sweden	+46 (0)8 754 6200	sweden@jtag.com

JTAG TECHNOLOGIES®

We *are* boundary-scan.®