

8/16-bit Microcontroller IP core (with multi-threading and pipeline support)

Unicore Systems Ltd

60-A Saksaganskogo St
Office 1
Kiev 01033
Ukraine
Phone: +38-044-289-87-44
Fax: : +38-044-289-87-44
E-mail: o.uzenkov@unicore.co.ua
URL: www.unicore.co.ua

Overview

B3 is a small, easy to use microcontroller core targeted for working as an embedded processor in different applications. It has a rich variety of commands, most of them require 1 CPU clock for execution. MCU operates with 8 and 16 bit data. Executable program and data share the same memory space. Maximal addressing range is 64 Kbytes. MCU has an additional 256-bytes IO space, which allows mapping different periphery. MCU is multi-thread. Thread request can be done by software and by hardware which provides flexibility to the designers. It has also a regular IRQ handling support, but IRQ execution is faster compared with other microcontrollers because user does not need to save register content at the beginning of the interrupt and does not need to restore it at the end.

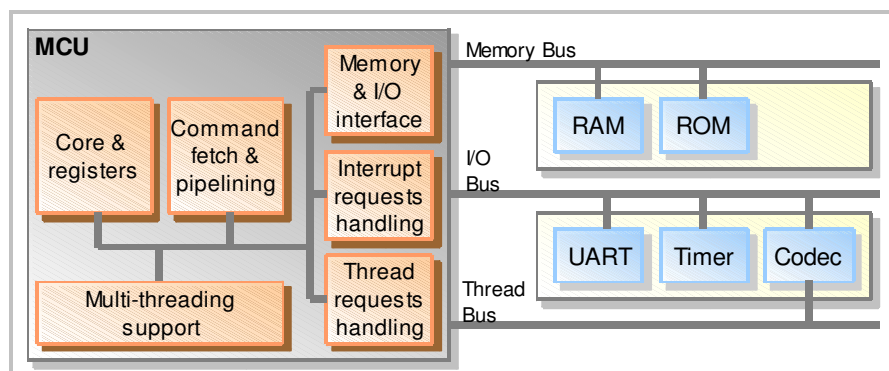


Figure 1. Typical System

Key Features

- 8 and 16 bit operations
- 64 Kbytes memory space and 256 bytes I/O space
- Command pipelining with 1 CPU clock of execution for most commands
- Native multi-threading support

Applications

- Communications
- Data codecs
- Real-time control
- Low power applications
- Smart DMA Controller.
- Multi-Thread applications.

- Effective Interrupt Requests handling
- Thread execution requests handling mechanism. External periphery can request the attention of the MCU when execution is needed. All other time MCU can stay in sleep mode, minimizing a consumption.
- IDE with integrated Linker and Assembler (instructions are included)

Functional Description

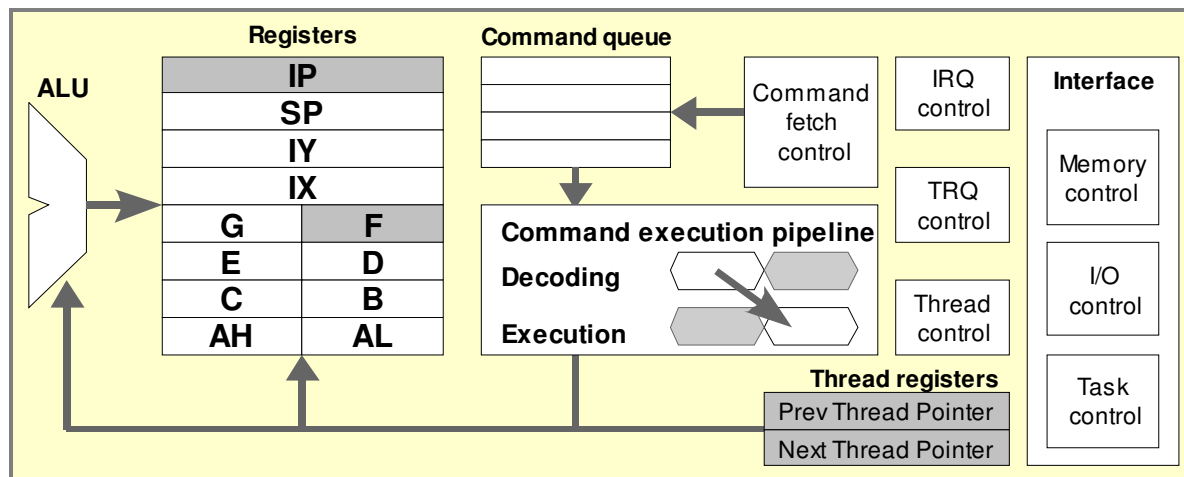


Figure 2. Simplified Block diagram of the microcontroller.

Register structure	Register structure is shown in Figure 2. All registers are identical when performing arithmetical operation (including IP (Instruction Pointer) and F (Flags) registers).
Memory addressing	Memory can be accessed: * directly, * indirectly, when address is placed into the MCU register, * directly or indirectly with offset, when 2 registers or register and a constant form an address, * with post-increment, post-decrement, pre-increment, pre-decrement, when address registers increments or decrements taking in account operands size. Memory access instructions require 1 or 2 CPU clocks. Device is optimized to work with synchronous RAM/ROM blocks.
IRQ	MCU has 8 IRQ inputs. Due to special multi threading mechanism IRQ handler does not have to store registers in the stack which makes interrupt requests extremely fast.
TRQ	There are 8 TRQ (Thread Execution Request) inputs. It requests execution of a thread from the place where it was previously stopped. There is a possibility to preserve values of all CPU registers which makes algorithm fast and effective. CPU clock can be stopped when no thread or interrupt requests are active. This can significantly improve the consumption.
I/Os	I/O address space is separated from memory address space. I/O instructions use

	different optimization criteria for better performance. All I/O commands require 1 CPU clock for execution. It is up to the designer whether to map external devices to Memory address space or to I/O address space.
Commands	MCU has a wide range of commands (data transfer, push and pop, arithmetic, shift, thread, IRQ and flow control). Command code has variable size (usually 1 byte) and often requires additional 1 or 2 bytes of data.
Multi-threading	Multi threading is natively supported by the core. There is a special set of commands. For example, there is a version of a JMP command which looks as a usual JMP for a calling process. But this command executes postponed threads from the list.

Core I/O signals

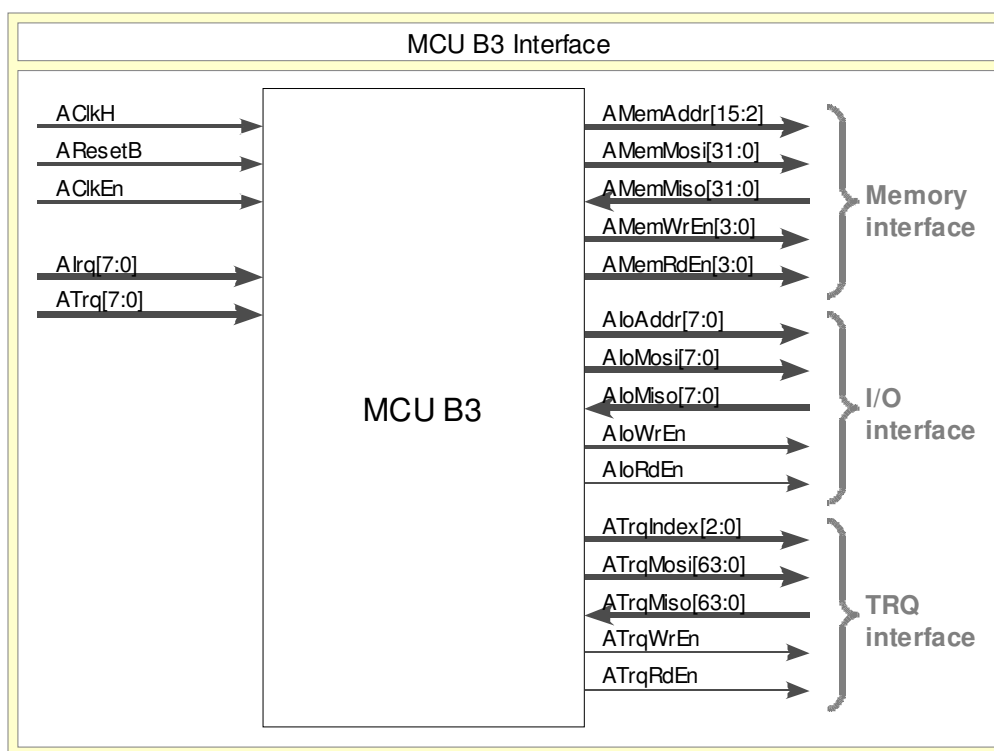


Figure 3. The MCU Symbol

Group	Signal	Direction	Description
Control	ACIkh	Input	CPU CLK signal
	AResetB	Input	Asynchronous reset (active low)
Memory interface	AMemAddr[15:2]	Output	Memory address
	AMemMosi[31:0]	Input	Memory data input (from memory to MCU)
	AMemMoso[31:0]	Output	Memory data output (from MCU to memory)
	AMemWrEn[3:0]	Output	Memory write enable
	AMemRdEn[3:0]	Output	Memory read enable
I/O	AIoAddr[7:0]	Output	I/O address

interface	AIoDataIn[7:0]	Input	I/O data input (from periphery to MCU)
	AIoDataOut[7:0]	Output	I/O data output (from MCU to periphery)
	AIoWrEn	Output	I/O write enable
	AIoRdEn	Output	I/O read enable
IRQ	AIrq[7:0]	Input	IRQ request
TRQ	ATrq[7:0]	Input	TRQ (Thread execution) request
	ATrqIndex[2:0]	Output	Thread index
	ATrqMiso[63:0]	Input	Thread context input (optional)
	ATrqMosi[63:0]	Output	Thread context output (optional)
	ATrqWrEn	Output	Thread context write enable (optional)
	ATrqRdEn	Output	Thread context read enable (optional)

Code and Data memory access

Code memory and Data memory share the same address space of the MCU. To access code and data there are 4 buses:

- Address bus AMemAddr[15:2]
- Data IN bus AMemMiso[31:0]
- Data OUT bus AMemMosi[31:0]
- Control bus AMemWrEn[3:0], AMemRdEn[3:0]

MCU can access 64Kbytes of memory. Address bus has 14 lines: AMemAddr[15:2]. 2 lowest address lines 1 and 0 are absent. Memory is organized as an array of 32-bit words. Control signals AMemWrEn[3:0] show which part of 32-bit word to write. The same way signals AMemRdEn[3:0] show which part of 32-bit word to read.

Executable code is always read in 4 bytes. Since commands have different size and are not aligned in the memory, MCU aligns them itself. When memory is not occupied, MCU continuously reads executable program and stores it in the internal cache (queue). This ensures that execution is not terminated when memory cannot be accessed to fetch next command (for example when application accesses memory to write or read data).

MCU is optimized to work with synchronous memory.

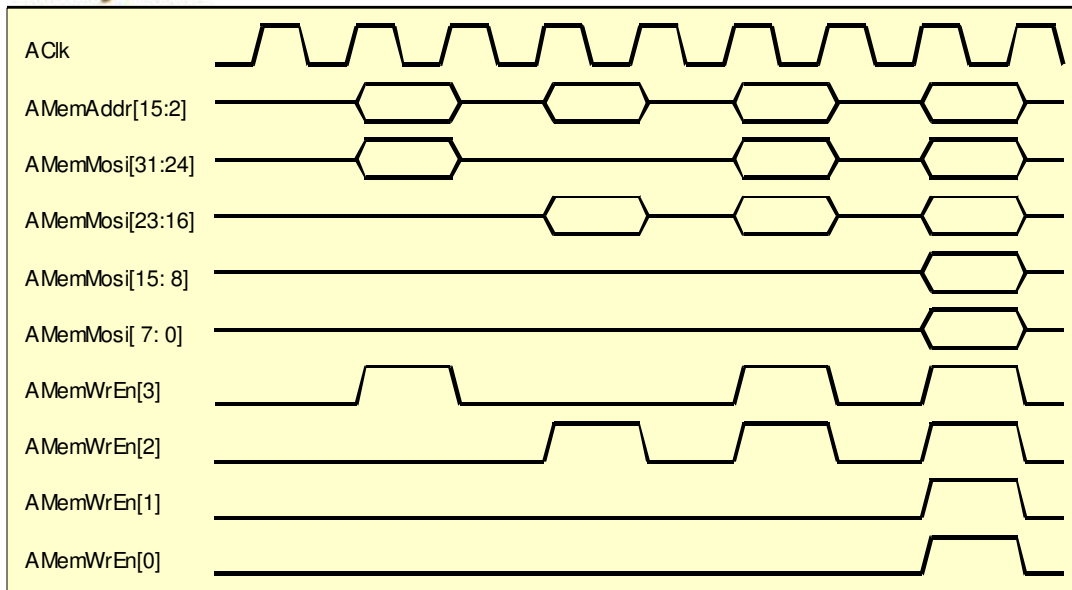


Figure 4: Memory write operations

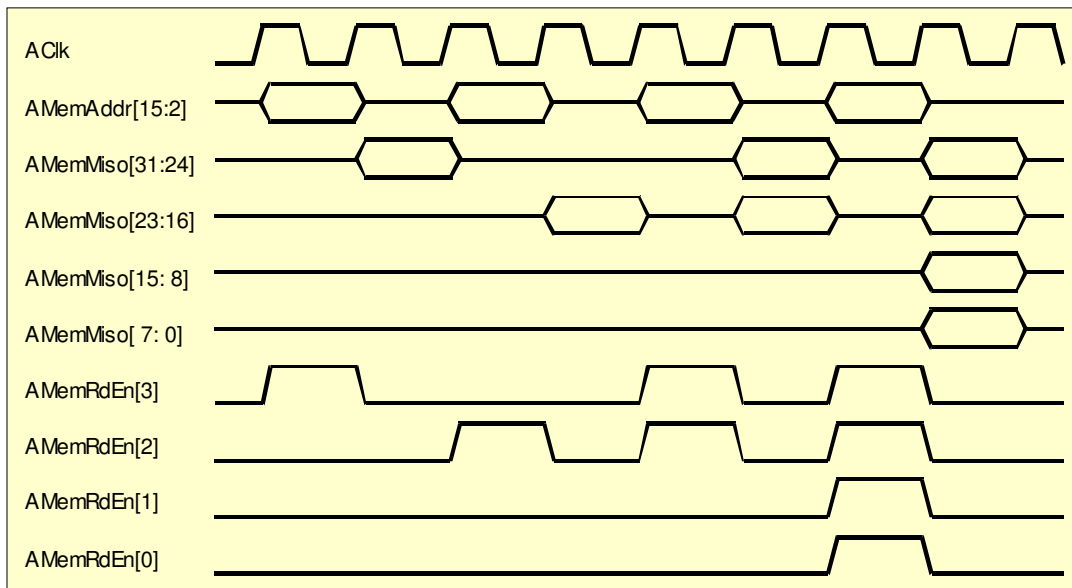


Figure 5: Memory read operations

I/O access

Input/Output space is used to access periphery. It has separate address bus $AIoAddr[7:0]$ and a separate data bus $AIoDataIn[7:0]$, $AIoDataOut[7:0]$.

Application developers can choose either map periphery on external address space or external IO space. Instruction code for IO access is usually smaller and faster, but does not allow many indexing modes and somewhat limited in registers use. Address space and IO space do not overlap.

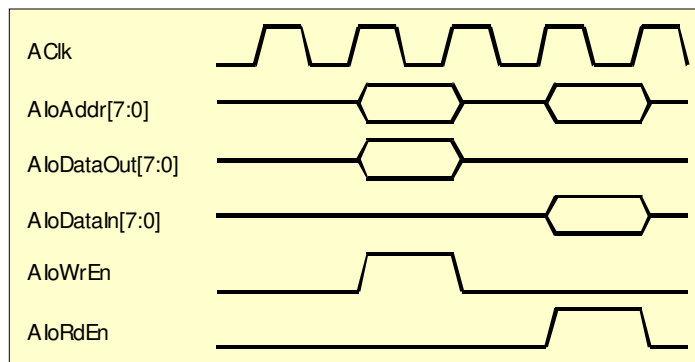


Figure 6: I/O Write and Read operations

IRQ and TRQ

In order to speed-up the reaction of MCU core to an external event, IRQ and TRQ mechanism is used. External periphery must set a request then special subroutine starts and executes the request. There is a special flag IF (interrupts enable) which can allow (when set) or disable (when cleared) IRQ and TRQ. Inside interrupt routine interrupts must be disabled, i.e. neither nested IRQ no nested TRQ allowed.

It takes 4.5 CLK to enter IRQ routine and 4.5 CLK to exit. IRQ routine does not need to save any register, all registers are saved and restored automatically.

There are some differences between IRQ and TRQ routines. See following table..

Differences between IRQ and TRQ		
Parameter	IRQ	TRQ
Start/return point	Starts at the beginning of interrupt routine handler	Continues from the place where it was previously stopped
Final instruction of interrupt/thread routine	IRET. Restores registers and resumes execution of interrupted program	TRET. Saves all registers. Some register values can be passed to hardware block (which issued TRQ request) as parameters. Restores registers of interrupted program and resumes its execution
Interrupt routine start	Registers are undefined, flag IF=0	Registers contain same value as they had before TRET instruction. Hardware block which issues TRQ request can modify some registers (including Flags) if it has to pass some parameters into TRQ handler
Performance	Requires 4-5 CLK to enter and 4-5 CLK to exit	Requires 4-5 CLK to enter and 4-5 CLK to exit. If TRQ request is active at the moment of TRET, physical exit/enter is skipped and TRET is executed as NOP. Hardware block which issues TRQ request can still modify register values. Only 2 CLK is required in this case

Software development tools

Software development includes IDE with integrated assembler and linker. System requirements:
OS Windows XP or Windows Vista

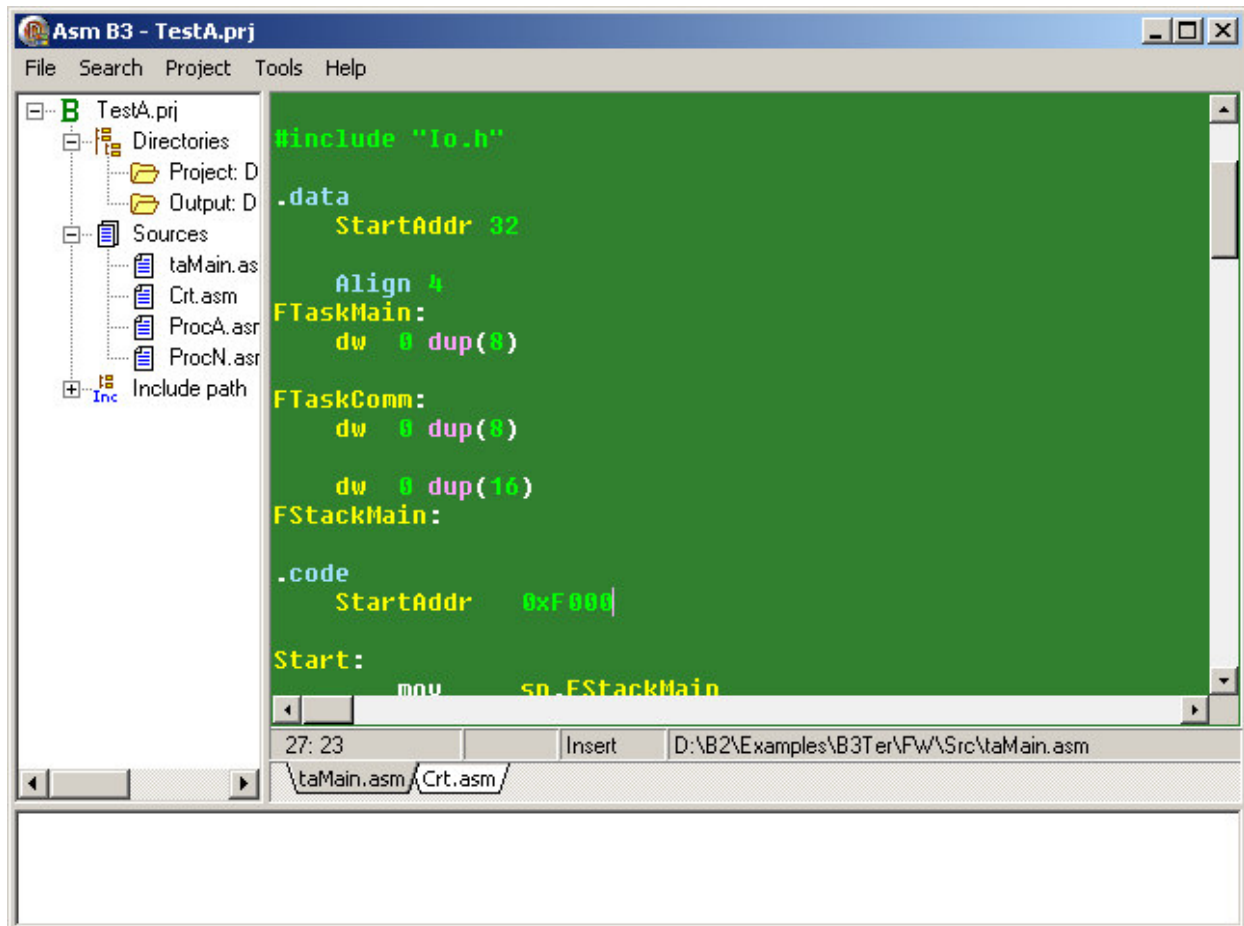


Figure 7. IDE with integrated assembler and linker

Verification description

The MCU core has been hardware verified on a custom made development board.

Design Features

- Technology Independent
- Layered architecture
- Fully Synchronous Design with no Latches
- Highly Modular Design with clearly defined interfaces
- Scan friendly RTL
- Consistent coding procedures

Implementation details

Target device	5vlx30ff676	4vsx25ff668
Slices (Area)	2123 (11%)	(16%)
Maximum system clock	204 MHz	146 MHz

These synthesis results are provided for reference only. Please contact us for estimates for your application

Deliverables

- RTL Verilog source code or synthesized netlist;
- Verilog Test environment;
- User guide, test specification and scripts.
- Example Synthesis scripts
- 3 months free email support to ensure successful integration into the customer's system
- Changes to the internal design to meet customer requirements are possible
- IDE with integrated assembler and linker

Ordering Information

This product is available directly from Unicore Systems Ltd under the IP License. To purchase or make further inquiries about this IP core or any other Unicore Systems products and services please contact us at the address specified on the front page.

All trademarks mentioned in the document are trademarks of their respective owners.