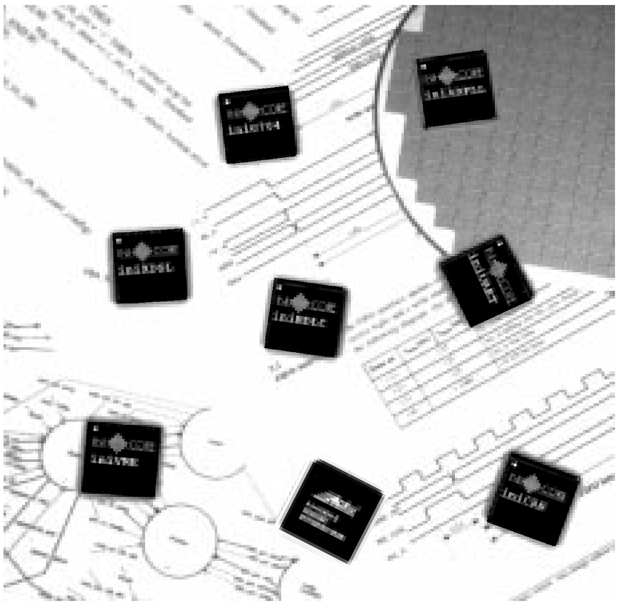
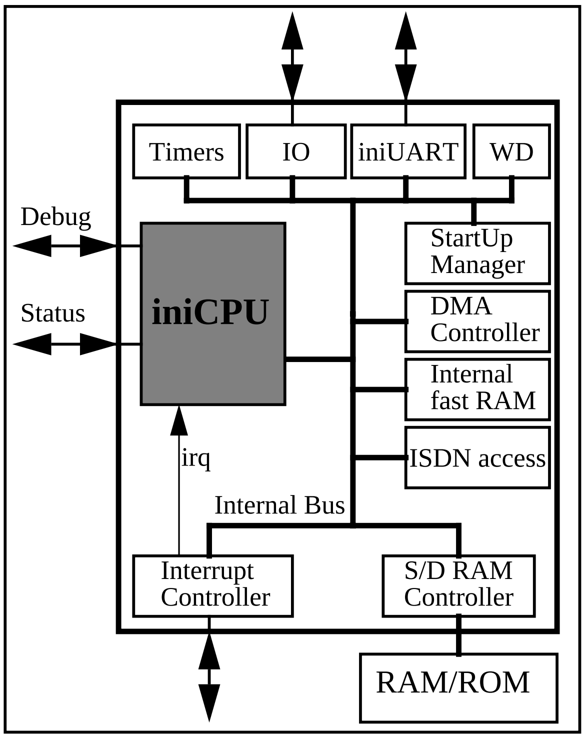
iniCPU Microprocessor data sheet



Features:

- Flexible 8bit Micro Controller Solution for Low to Mid Range Applications
- 6809 Software Compatible
- Save Operation, Illegal Opcode Recognition
- 3rd party C-Compiler, Assembler
- Address Expansion Circuit (Page Mode)
- External Interfaces for S/D RAM, IO etc.
- Hardware Debugging Circuits
- Fully Synchronous, 100% Technology Independent Design
- Performance at 40MHz with 25ns RAM:
  Peak: 10 Mips
  8 by 8 unsigned mul: 3.3 Mips
- Demoboard Available...

xample usage of iniCPU in a System:



iniCPU, a powerful 8bit micro processor, enables embedded computing and controlling on chip. It fulfils all needed requirements for being used in systems on chip and for high level programming.

Although existing standard products are cheap today, there are a lot of applications where an integration of the micro processor brings total system cost down. Less components, higher performance, glueless memory interfaces and customized peripherals allow you to build your system on chip.

The ‘System on Silicon’ approach with iniCPU ‘on board’ enables powerful features: Extended memory, software download at start-up, interfaces for serial EEPROM or cheap DRAM, DMA with any target and optimized communication with customized functions like filter algorithms and other cores like UART, CAN, HDLC, SBUS. For special requirements, like fast interrupt response time, we enhance the iniCPU to your demands.

INICORE created iniCPU, the structured VHDL model for simulation and synthesis on any target technology. Standard tools like debugger, assembler and C compiler are available from third parties.

INICORE - the reliable Core and System Provider. We provide high quality IP, design expertise and leading edge silicon to the industry.

US Sales Office:
INICORE INC.
5600 Mowry School Road, Suite 180,
Newark, CA 94560
Tel: 510 445 1529 Fax: 510 656 0995
E-mail: ask_us@inicore.com
Web: www.inicore.com

INICORE AG
Mattenstrasse 6a,CH-2555 Brügg, Switzerland
Tel: ++41 32 374 32 00, Fax: ++41 32 374 32 01
E-mail: ask_us@inicore.ch
Web: www.inicore.ch

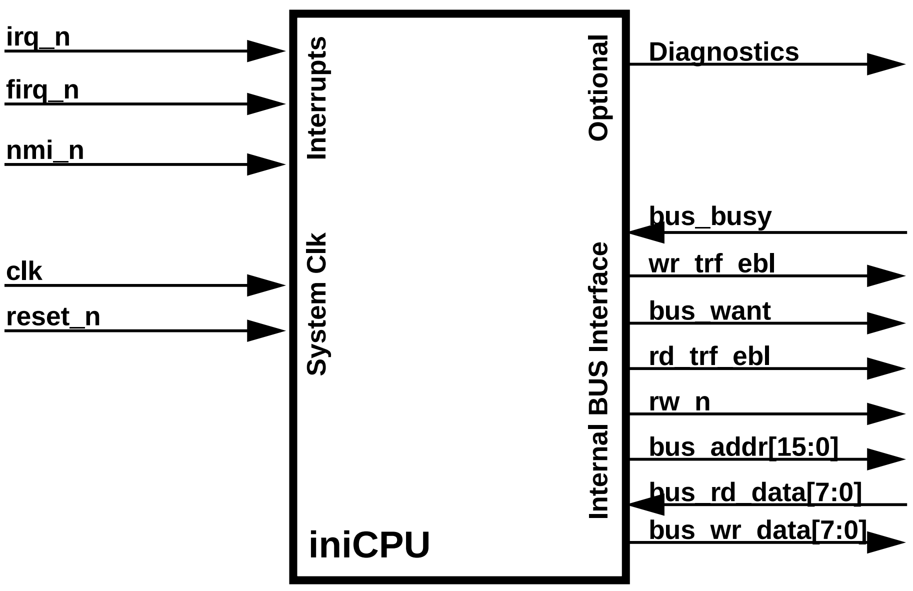
1 Overview

The iniCPU is a software compatible 6809 microprocessor implemented in VHDL using a stuctured and synchronous design methodology.
This flexible 8-bit microprocessor can be used for low to mid range applications and is easily integrated into any ASIC or FPGA technologies. The iniCPU is designed that application specific performance enhancements (e.g., high-speed multiplier) can be integrated too.

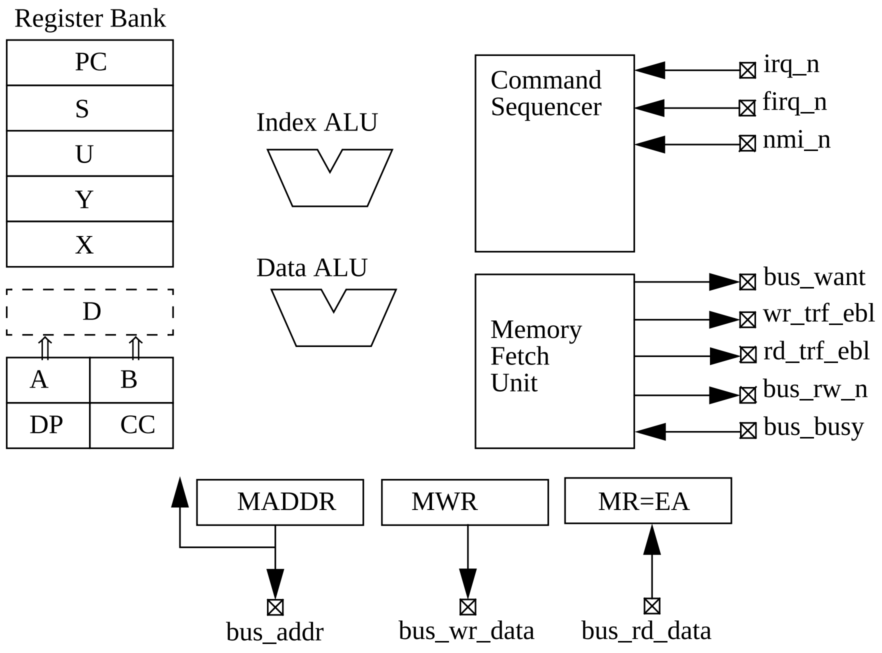
Inicore built a simulation and a hardware verification environment. The simulation environment is VHDL based and simulates the synthesizable VHDL code of the iniCPU. This functional VHDL testbench reading Motorola S-Records from a file can be used for performance check, software verification or to get used to the functionality.
A demonstration board is available which shows the functionality of the iniCPU in a hard ware testbed. For this purpose, the microprocessor is implemented into an Actel FPGA.

2 Structure of iniCPU

Pinout of iniCPU:



Structure of iniCPU:



The main parts of the iniCPU are:
- Command Sequencer Unit: Controls the main functionality of the processor.
- Memory Fetch Unit: This unit handels the Internal uP Bus.
- ALU: Arithmetic Logic Unit.
- Internal Registers: See chapter ‘4. Programming Model’ for more information.
- Bus Registers: MADDR: Memory Address Register
  MWR: Memory Write Register
  MR: Memory Read Register = Effective Address

The modular structure of the iniCPU allows flexible handling of the Arithmetic Logic Unit (ALU).

3 IO description

The following part lists the input and output ports of INICORE’s iniCPU core and explains their functionality.

3.1 General inputs

These pins (clk and reset_n) are use to clock and initialize the whole iniCPU core. There are no other clocks in this core.

Pin Name	Type	Description
clk	in	System clock
reset_n	in	Asynchronous system reset, active low

3.2 iniCPU IO’s

3.2.1 Interrupts of iniCPU

Please see the functional description of the Interrupts in chapter ‘5.2 Interrupts’

Pin Name	Type	Description
nmi_n	in	Non maskable interrupt, active low
firq_n	in	Fast interrupt request, active low
irq_n	in	Interrupt request, active low

3.2.2 Internal BUS Interface of iniCPU

The interface is synchronous to the system clock. See also chapter ‘5.1 Internal Bus Interface Timing’ for more information.

Pin Name	Type	Description
bus_busy	in	Bus busy, If ‘1’ the uP can not access the bus and waits until bus is released.
wr_trf_ebl	out	Write transfer enable
bus_want	out	Bus want. The uP requests the bus.
rd_trf_ebl	out	Read transfer enable
rw_n	out	Read / write-not. If ‘1’ then read cycle
bus_addr[15:0]	out	Bus address
bus_rd_data[7:0]	in	Read data
bus_wr_data[7:0]	out	Write data

3.2.3 Diagnostic Signals of iniCPU

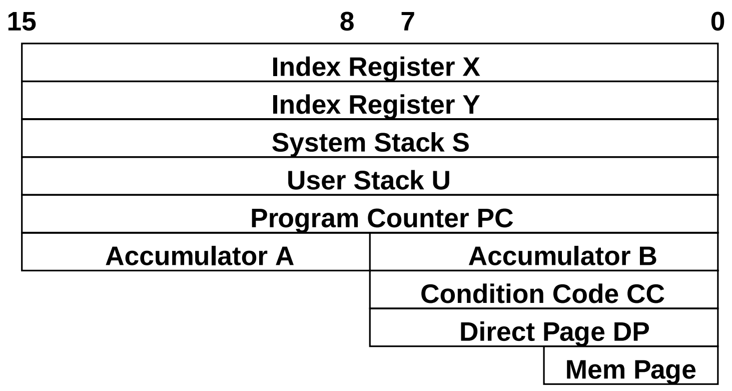
These signals are only used for diagnostic purposes and are not necessary for standard operation.

Pin Name	Type	Description
mem_rdy	out	Memory ready, see chapter 5.1 ‘Internal Bus Interface Timing’ for timing diagram
seq_state[5:0]	out	Command sequencer state register
cc[7:0]	out	Condition code register
pc[15:0]	out	Program counter register
stack[15:0]	out	Stack pointer
stack_operation	out	Stack operation, if ‘1’ a stack operation cycle is running
opcode[5:0]	out	Operation code

4 Programming model

4.1 Internal Registers

The following table shows the existing registers in the iniCPU:



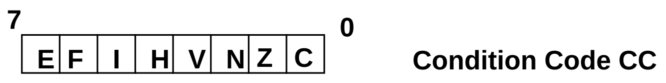
Index Registers (X, Y)
The index registers are used in indexed mode of addressing and are used for calculating the effective addresses. During some indexed modes, the contents of the index register are incremented or decremented to point to the next item of tabular type data. All 4 pointer register (U, S, X, Y) can be used as index register.

Stack Pointer Registers (U, S)
The system stack pointer S is automatically used by the processor during subroutine calls and interrupts! The user stack pointer is controlled by the programmer. Both stack pointers point to the top of the stack.

Program Counter Register
The program counter (PC) points to the address of the next instruction to be executed.

Accumulators (A, B, D)
The A and B accumulators are general purpose registers. Some instructions concatenate the registers A and B to achieve 16 bit values. This register is then referred as the D register with the A register as most significant byte.

Condition Code (CC) Register



Beginning with bit(0) the condition code register contains following flags:

C:	Carry from ALU
Z:	Zero from ALU
N:	Negative (2 complement) from ALU
V:	Overflow (2 complement) from ALU
H:	Half Carry (low nibble) from ALU
I:	Interrupt mask register for irq_n
F:	Interrupt mask for firq_n
E:	Entire Flag for irq_n/firq_n stack operations

Direct Page (DP) Register:

These registers enhance the direct addressing mode. During direct addressing instruction execution, the contents of DP appears at the higher address outputs (A15- A8).

4.2 Instructions/ Address modes

All standard instructions (load, store, and, or, add, sub, etc.) are implemented. Further, there are transfer/exchange instructions for all registers, 8 by 8 unsigned multiply, decimal adjust and signed/unsigned conditional branches. A powerful stack command is also available. Software interrupts are supported.

All instructions[1] may be used with different addressing modes:

- Inherent: Register source and destination, no post bytes, e.g. INCA
- Immediate: Postbyte is parameter, e.g. LDA #100
- Extended: Postbytes are address, e.g. LDA 1000
- Direct: Postbyte is low address, DP is high address, e.g. LDA <50
- Indexed: Postbyte(s) define indexed mode:
  (Index registers are X,Y,U,S)
  - Post increment by 1 and 2, e.g. LDA ,S++
  - Pre decrement by 1 and 2, e.g. LDA ,-X
  - Zero offset, e.g. LDA 0,X
  - 5/8/16bit signed offset, e.g. LDA -7,Y
  - A/B/D signed offset, e.g. LDB A,U
  - 8/16bit signed offset PC relative, e.g. LDA 665,PCR
- Indexed indirect: The same types as indexed may be used. The indirect mode uses the argument of the index mode as address for an indirect access.

The iniCPU core contains already special instructions to set and to read the memory page register. This way, memory paging is not time wasting and enables the use of more than the limited 64k bytes address space.

To insure save operation, illegal opcodes are recognized and will lead to a exception, where a vector is fetched to enter the exception handler routine.

1. Restrictions only for special commands

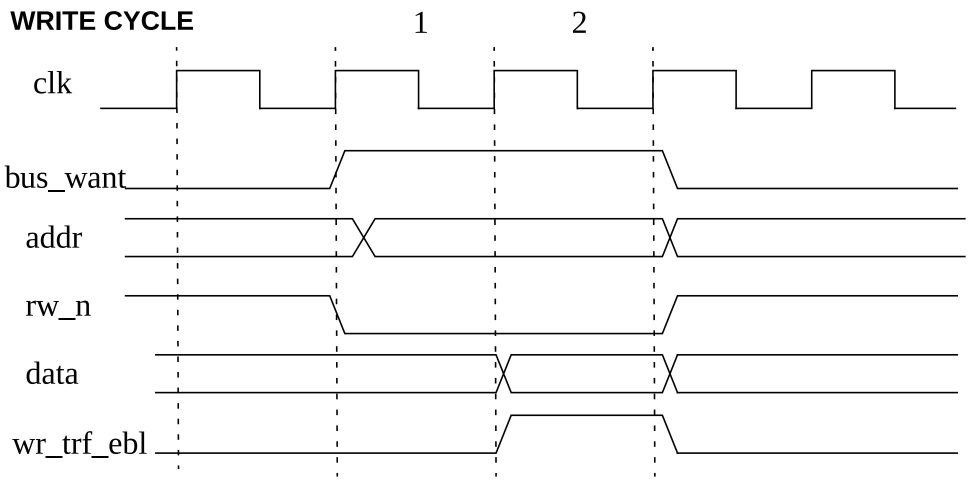
5 Timing

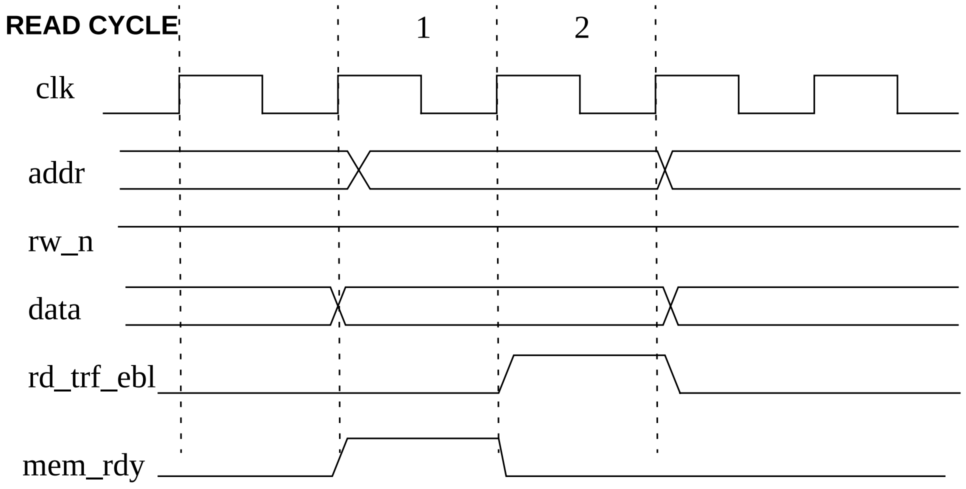
The next chapter shows the timing for the iniCPU interface. Note that this is not the external representation of the bus interface, but an internal bus that is needed for on-chip communication.

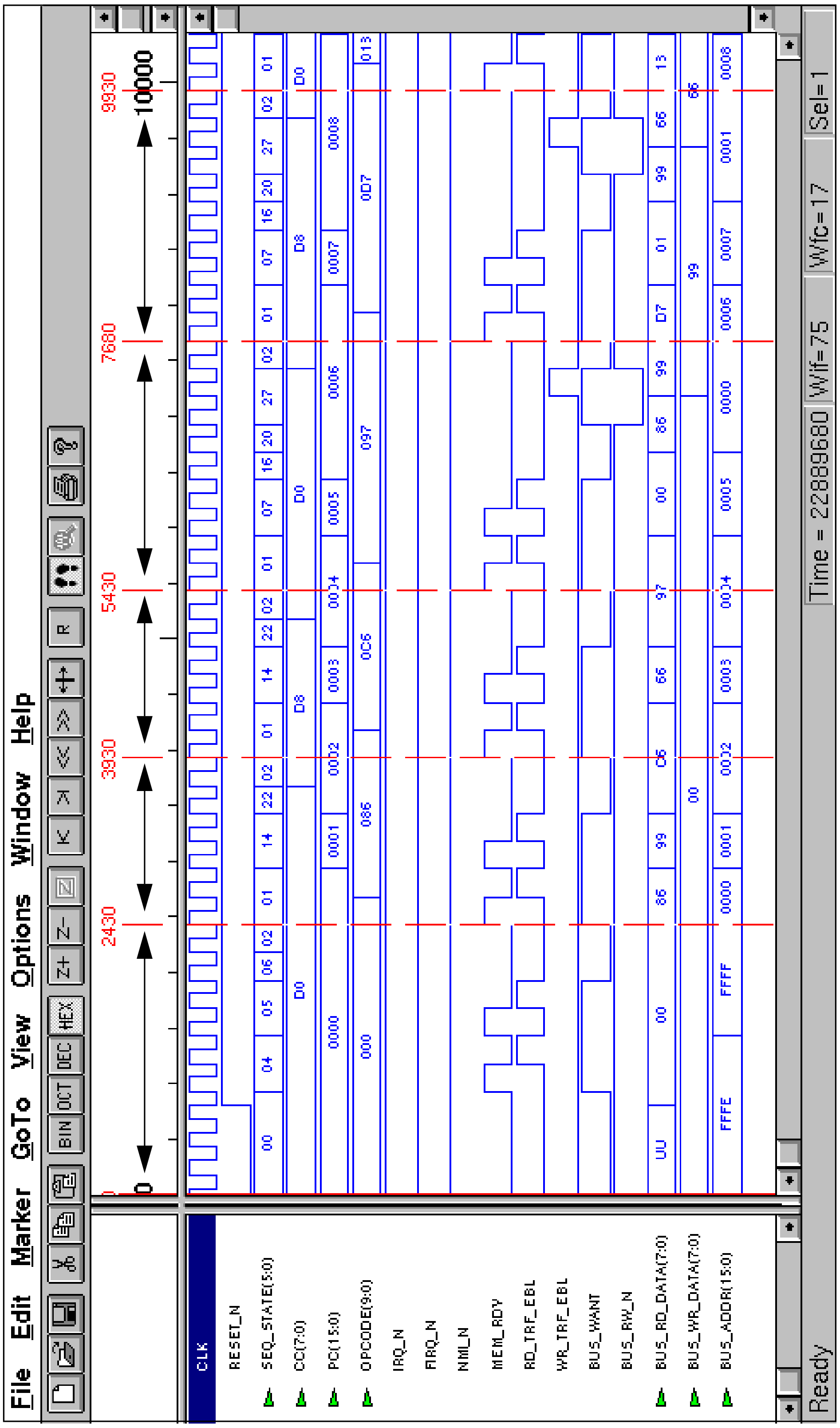
5.1 Internal BUS Interface Timing

For communicating with the on chip peripheral, the iniCPU core uses a synchronous parallel bus. The wr_trf_ebl signal acts as enable for data transfers. Two clock cycles are used to drive a normal bus cycle.

The bus arbitration is handled by a bus arbiter which is not located inside the iniCPU core. The reason for this is that the bus interface can be designed to fulfill the application needs without changing the iniCPU core. The bus control between the iniCPU and the arbiter is handled by two signals: bus_want and bus_busy. The iniCPU requests the bus by setting the bus_want signal. By the next rising clock cycle the iniCPU evaluates the bus_busy signal. Is the bus_busy signal set to ‘1’ then the iniCPU goes into a wait-state mode and waits until bus_busy is reseted. The iniCPU will only jump into this wait-sate mode when a bus_want isn’t granted (bus_busy = 1). The bus_want signal will be set as long the bus_busy is set and will be reseted one clock cycle later than the bus_busy signal (see timing diagram ‘arbitration write cycle’ and ‘arbitration read cycle’).
Using this communications scheme DMA and other on-chip functions can be integrated without changing anything on the iniCPU design.

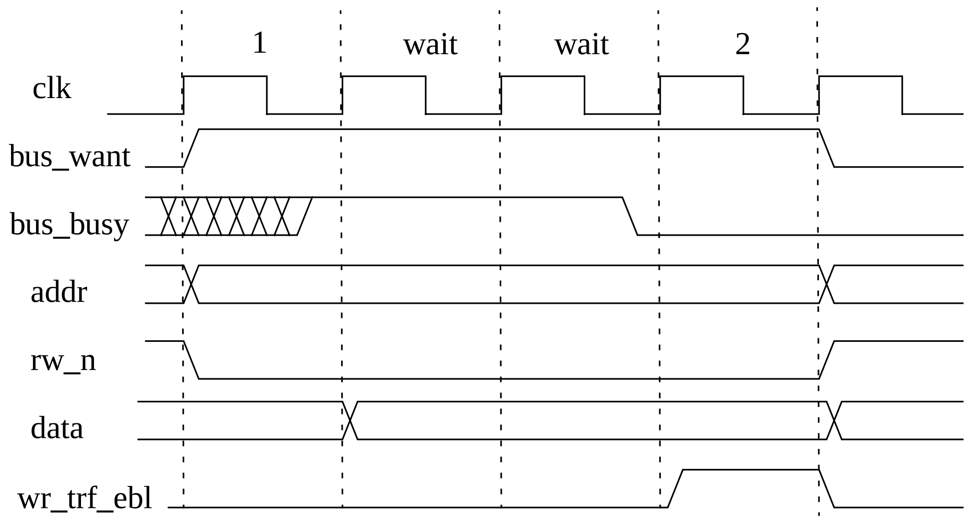




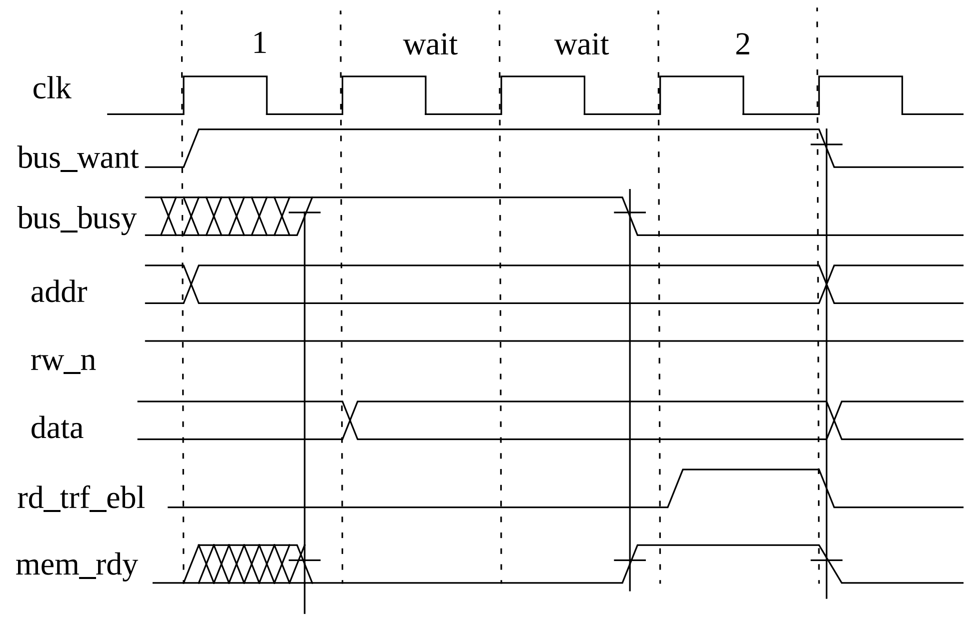

File Edit Marker GoTo View Options Window Help
BIN OCT DEC HEX Z+ Z- R
0 2430 3930 5430 7680 9930 10000
CLK
RESET_N
SEQ_STATE(5:0)
CC(7:0)
PC(15:0)
OPCODE(9:0)
IRQ_N
FIRQ_N
NMI_N
MEM_RDY
RD_TRF_EBL
WR_TRF_EBL
BUS_WANT
BUS_RW_N
BUS_RD_DATA(7:0)
BUS_WR_DATA(7:0)
BUS_ADDR(15:0)
Ready
Time = 22889680
Wf= 75
Wfc=17
Sel=1

The following timing diagram shows how the bus_busy signal forces the iniCPU into wait-sates.

ARBITRATION WRITE CYCLE



ARBITRATION READ CYCLE



The original motorola 6809 bus timing can be easily handled with the bus_want and the bus_busy signals and a external (iniCPU core external) 6809 uP-Interface.

5.2 Interrupts

For interrupting the iniCPU, there are 3 level-sensitive interrupts:

- **irq_n**: standard interrupt or interrupt request, low_active,
  pushes all registers on stack:
  => PC(Low), PC(High), U(Low), U(High), Y(Low), Y(High), X(Low), X(High), DP, B, A, CC
  Latency[1] is 1.35us@40MHz
- **firq_n**: fast interrupt, pushes only PC and CC on stack.
  => PC(Low), PC(High), CC
  Latency is 0.5us@40MHz
- **nmi_n**: non maskable interrupt, pushes all registers on stack.
  => PC(Low), PC(High), U(Low), U(High), Y(Low), Y(High), X(Low), X(High), DP, B, A, CC
  Latency is equal to irq_n. The nmi_n is masked after reset until the system stack pointer is initialized.

The different interrupts will fetch the corresponding vector for the interrupt in the memory. For special applications, the iniCPU core can be modified by adding high speed interrupts with a minimum of latency.

1. Latency is defined here as the time to react on the interrupt when iniCPU is in idle state.

6 Peripherals and add-ons

For interfacing with external peripherals, there are different modules available, which all can be connected to the internal bus. Different modules may be used for the same system (with different access speeds):

- **Standard Bus**
  Any type of external bus interfaces may be connected: 6809 like, 8051 like, 68000 like, user defined.

- **SRAM interface**
  SRAMs can be directly connected to this module, since all glue logic is already included.

- **DRAM interface**
  DRAMs can be directly connected to this module. Address multiplexing is done in the module. This enables cheap uP systems with large memory space.

- **serial EEPROM**
  For EPROMless systems, the boot code may be stored in an external EEPROM (small, cheap). This memory is loaded after reset in the RAM (internal or external), and the iniCPU is started after the bootload process.

- **Software download**
  After reset_n, the software can be downloaded via UART etc. through a standard protocol. Data is stored via DMA in the uP's memory. After download, the CPU is started by the start-up handler.

- **BUS scrambler**
  For protecting the software, data and address bus may be scrambled to inhibit disassembling code in the external ROM.

- **Special Interfaces**
  For AD/DA converters, existing parts etc., a glue less customized interface may be connected.

- **DMA**
  Any type of DMA controllers (internal or external) may be connected, since all structures for memory arbitration are already included.

- **Co-Processors**
  Hardware multipliers etc. may be added and controlled by special opcodes. This raises the performance of dedicated functionality in a high performance level without having an expensive CPU.

- **Debugging**
  Single step modes, internal register access, break points etc. make it easy to debug even an on chip CPU, where the bus is not available externally!