National Semiconductor

PRELIMINARY
November 1988

HPCCOREA High-Performance Microcontroller Core

General Description

The HPCCOREA Megacell is the ASIC HPC™ Core Megacell available in the cell based design library. The HPC core is the basis for the HPC family of High Performance microcontrollers available as standard products from National Semiconductor. Each member of the family is built around this core CPU with a unique memory and I/O configuration to suit specific applications. Therefore, designers can now customize the peripherals surrounding the HPC core to meet their precise needs, while integrating additional system logic onto the same die. A single chip system solution becomes possible. All Cell Based IC's are fabricated in National's advanced 2μ dual layer metal microCMOS technology. This process combined with an advanced architecture offers fast, flexible I/O control, efficient data manipulation and high speed computation.

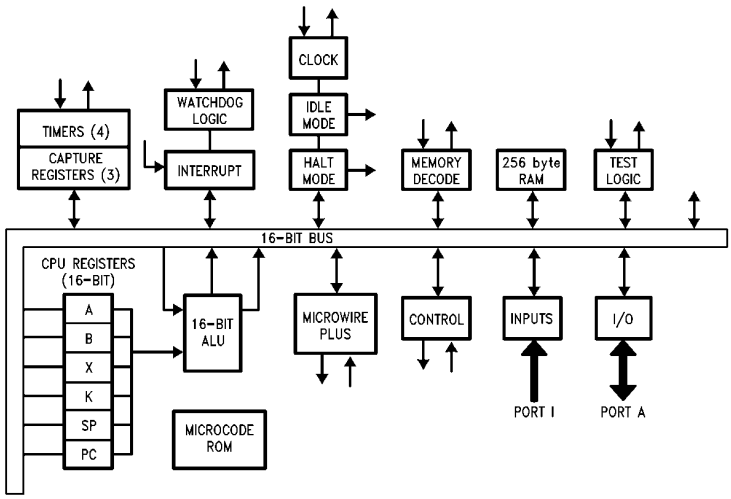
The HPC core Megacell permits a complete microcomputer environment and more on a single chip. Customization of microcontroller applications and inclusion of surrounding system logic and memory, produce a cost effective system solution for high performance applications. Core functions such as 4 16-bit timers with 3 input capture registers, vectored interrupts, WATCHDOG™ logic, and MICROWIRE/PLUS™ provide efficient paths for system integration. The ability to address up to 64 kbytes of memory enables the HPC core to be used in powerful applications typically performed by microprocessors and expensive peripheral chips.

The microCMOS process results in very low current drain and enables the user to select the optimum speed/power product for his system. The IDLE and HALT modes provide further current savings.

HPC Core Features

- 16-bit architecture, both byte and word
- 16-bit data bus, ALU and registers
- 64 kbytes of memory mapped addressing
- FAST—240 ns for register instructions when using 17.0 MHz clock
- High code efficiency—most instructions are single byte
- 16 x 16 multiply and 32 x 16 divide
- Eight vectored interrupt sources
- Four 16-bit timer/counters
- Three 16-bit capture registers
- WATCHDOG logic
- MICROWIRE/PLUS serial I/O interface
- CMOS—very low power save modes: IDLE and HALT
- Programmable wait states
- 16-bit general purpose/address data bus I/O port
- 8-bit general purpose input port
- 256 byte RAM
- Megacell Test Logic
- Enhanced access to core features

Block Diagram—HPC Core Megacell



TL/U/9982–1

TRI-STATE® is a registered trademark of National Semiconductor Corporation.
HPC™, MICROWIRE™, MICROWIRE/PLUS™, MOLE™ and WATCHDOG™ are trademarks of National Semiconductor Corporation.

 TL/U/9982

RRD-B30M115/Printed in U. S. A.

Signal Descriptions

Signal	Reset	Active	Type	Load Factor	Description
PORT A					
Port A is a 16-bit bidirectional I/O port with a data direction register to enable each separate pin to be individually defined as an input or output. When accessing external memory, Port A may be used as the multiplexed address/data bus.					
PA(0:15)*	Z		B		Port A—this may be configured as a 16-bit Addr/Data bus or a 16-bit general purpose I/O port.
PI(0:15)			I	2	Port A data in.
PD(0:15)	U		O		Port A data out.
PE(0:15)	1		O		Port A bidirectional control.
EXAC			I	1	Defines the configuration of Port A. 1 = Address/Data bus, 0 = General purpose I/O port.
RDEX		H	I	1	Read Port A onto the internal data bus. All memory locations connected to Port A should be decoded and input on this pin when in Addr/Data bus mode.
*PA is created when PI, PD and PE are connected to a 4 mA I/O buffer macro. Allowable macros include BF0B4D, BF0B4N, BF0B4U, BB0B4D, BB0B4N, and BB0B4U.					
PORT I					
Port I is an 8-bit input port that can be read as general purpose inputs or can be used for the following functions:					
I0			I	1	General purpose input.
I1			I	1	Port I: bit 1 of Port I. Interrupt: rising edge triggered non-maskable interrupt (NMI). IRPD: bit 1 is set when interrupt is pending and is reset when interrupt is acknowledged. Halt & Idle mode: rising edge will exit Halt or Idle mode. Entering Halt with NMI high: stop oscillator, stop C1 & C2 at C1 = 0 C2 = 1. Entering Halt with NMI low: logic is stopped, but oscillator is kept alive.
I2			I	1	Port I: bit 2 of Port I. Interrupt: Enabled if bit 2 of ENIR is set. Rising/falling edge trigger is selected if bit 2 of IRCD is set/reset. IRPD: bit 2 is set when interrupt is pending and is reset when interrupt is acknowledged. Capture trigger input: Value of Timer T0 is copied into I2CR register when interrupt occurs.
I3			I	1	Port I: bit 3 of Port I. Interrupt: Enabled if bit 3 of ENIR is set. Rising/falling edge trigger is selected if bit 3 of IRCD is set/reset. IRPD: bit 3 is set when interrupt is pending and is reset when interrupt is acknowledged. Capture trigger input: Value of Timer T0 is copied into I3CR register when interrupt occurs.
I4			I	1	PORT I: bit 4 of Port I. Interrupt: Enabled if bit 4 of ENIR is set. Rising/falling edge trigger is selected if bit 4 of IRCD is set/reset. IRPD: bit 4 is set when interrupt is pending and is reset when interrupt is acknowledged. Capture trigger input: Value of Timer T0 is copied into I4CR register when interrupt occurs.
I5			I	1	Port I: Bit 5 of Port I. UWIRE: Serial input to SIO shift register, bit 0 (MWIN).
I6			I	1	General purpose input.
I7			I	1	General purpose input.

Z = Tristate U = Unknown I = Input O = Output
H = Active High L = Active Low B = Bidirectional

Signal Descriptions (Continued)

Signal	Reset	Active	Type	Load Factor	Description
INTERRUPT SIGNALS					
The core contains eight external interrupts.					
I1–I4			I	1	Refer to Port I.
INT5		H	I	1	Interrupt: Enabled if bit 5 of ENIR is set. Level interrupt ORed with internal Timer interrupt signals. IRPD: Bit 5 is set when interrupt pending.
INT6		H	I	1	Interrupt: Enabled if bit 6 of ENIR is set. Level interrupt. IRPD: Bit 6 is set when interrupt pending.
INT7		H	I	1	Interrupt: Enabled if bit 7 of ENIR is set. Level interrupt. IRPD: Bit 7 is set when interrupt pending.
RSET		H	I	1	Core reset, level sensitive.
CLOCK SIGNALS					
CKIN			I	1	System clock. Typically connected to the output of an oscillator macro.
C1			O		Buffered system clock, CKIN.
C2			O		Buffered CKIN divided by 2.
NSTP	1	L	O		Signal to stop an oscillator during halt mode.
WATCHDOG SIGNALS					
WDOT	0	H	O		Signal indicating an illegal condition has been detected by the watchdog logic.
WDIN		L	I	1	Signal to reset watchdog logic.
STWD		H	I	1	Signal to force watchdog to trip.
MICROWIRE/PLUS SIGNALS					
MWOT	U		O		μWIRE output: serial output from SIO register bit 7. μCODE serial dump: serial output from μCODE shift register. This output is multiplexed with the μWIRE output.
I5			I	1	MWIN: μWire input shared by Port I.
OSKI			I	1	μWIRE/PLUS clock input.
OSKO	Z		O	1	μWIRE/PLUS clock output, tristable.
MWMS	0		O		μWIRE Master/Slave mode status. 1 = Master 0 = Slave
TIMER SIGNALS					
T2IN			I	1	Timer T2 external clock input (edge-triggered).
T3IN			I	1	Timer T3 external clock input (edge-triggered).
T2OT	U	H	O		Timer T2 output pulse (1 C1 wide) flagging underflow condition and triggering reload of T2 with the data from R2.
T3OT	U	H	O		Timer T3 output pulse (1 C1 wide) flagging underflow condition and triggering reload of T3 with the data from R3.
TDIV	U		O		General purpose programmable clock. Time base is selected through the DIVBY register.
T0CY	I	H	O		T0 carry output indicating T0 overflow condition. (1 C2 wide pulse.)
ADDRESS/DATA BUS CONTROL SIGNALS					
DB(0:15)	1/0		B		Internal 16-bit precharged address/data bus.
NRD	1	L	O		Read. Useable with internal address/data bus or port A.
MTB	0	H	O		Memory to Bus. Useable with internal address/data bus, only.
NWR	1	L	O		Write. Useable with internal address/data bus and Port A.
ALE		H	O		Address latch enable.
HBE	0	H	O		High byte enable.

Z = Tristate U = Unknown I = Input O = Output
H = Active High L = Active Low B = Bidirectional

Signal Descriptions (Continued)

Signal	Reset	Active	Type	Load Factor	Description
CPU CONTROL SIGNALS					
ST1	0		O		Bus Cycle Status: indicates first opcode fetch.
ST2	0		O		Bus Cycle Status: indicates machine states (skip, interrupt or instruction start).
RDY		H	I	1	Ready acknowledge. Used to extend the bus cycle for slower memories.
NHLD		L	I	1	HOLD request, TRI-STATE Port A for external control of address/data bus. Internal bus placed in precharge state.
NHDA	I	L	O		Acknowledge of HOLD request.
WATA		H	I	1	All memory locations requiring wait states should be decoded and input on this pin.
SEL8			I	1	Select bit to indicate 8-bit or 16-bit mode. 16-bit-pull low. 8-bit-memory locations external to core must be decoded on this pin.
HALT	0	H	O		Halt/Idle mode acknowledge. Indicates clock is halted at C1 = 0 and C2 = 1 for Halt mode.
ADDRESS DECODE SIGNALS					
SELA	0	H	O		Decode output indicating user peripheral address block 0100–011F.
SELB	0	H	O		Decode output indicating user peripheral address block 0120–013F.
SELC	0	H	O		Decode output indicating user peripheral address block 0140–015F.
SELD	0	H	O		Decode output indicating user peripheral address block 0160–017F.
SELX	0	H	O		Decode output indicating user peripheral address block 0200–EFFF.
SROM	0	H	O		Decode output indicating user peripheral address block F000–FFFF.
SLIO	0	H	O		Decode output indicating user peripheral address block 00E0–00FF.
TEST SIGNALS*					
TEST		H	I	12	Selects test mode.
TST1			I	2	Input for RDY/HOLD signals in test mode. Not used in non test mode.
TST2			I	1	Input for T2IN/T3IN signals in test mode. Not used in non test mode.
TRST		H	I	1	Input for RSET signal in test mode. Not used in non test mode.
TCLK			I	1	Input for clock signal in test mode. Not used in non test mode.
MUX(0:8)			O		Output of test multiplexers. User outputs when not in test mode.
USR(0:8)I					User logic inputs to output test multiplexers. Use to access test I/O pins in user logic mode.

*All test signals must be brought to an appropriate I/O macro.

Z = Tristate U = Unknown I = Input O = Output
H = Active High L = Active Low B = Bidirectional

HPC Core DC Characteristics V_{CC} = 5.0V ±10%

Symbol	Parameter	Test Conditions	Min	Typ	Max	Units
I_{CC1}	Active Current	V_{CC} = 5.0V fc = 17.0 MHz		30		mA
		V_{CC} = 5.0V fc = 2.0 MHz		3.5		mA
I_{CC2}	Idle Mode Current	V_{CC} = 5.0V fc = 17.0 MHz		3.0		mA
		V_{CC} = 5.0V fc = 2.0 MHz		0.35		mA
I_{CC3}	Halt Mode Current	V_{CC} = 5.0V fc = 0 kHz		200		μA
		V_{CC} = 2.5V fc = 0 kHz		100		μA
V_{RAM}	RAM Keep Alive Voltage		2.5			V
C_{BUS}	Address/Data Bus Capacitive Load (DB(0:15))					pF

HPC Core AC Timing Parameters $V_{CC} = 5.0V \pm 10\%$

Symbol	Description	Min	Typ	Max	Units
f_c = Input Frequency (CKIN)	Operating Frequency	2		17.0	MHz
fcdut = 50% ±10%	Operation Frequency Duty Cycle	40		60	%
tC1 = 1/fc	Clock Period	59		500	ns
tC2 = 2/fc	Cycle Time	118		1000	ns
WS = 1tC2	Wait States	118		1000	ns
fxin = 1/19tC1 (T2IN, T3IN)	External Timer Input Frequency		892		kHz
txin	Pulse Width for Timer Inputs	40	177		ns
fMW = 1/19tC1	External MICROWIRE/PLUS Clock Input Frequency (MWIN)		892		kHz

Symbol	Description	Ref	Best Case*		Worst Case*		Units
			FO = 1	d/FO	FO = 1	d/FO	
tCKC1	C1 Delay from CKIN (Note 1)	CKIN, RE	4.4	0.01	31.9	0.1	ns
tCKC2	C2 Delay from CKIN (Note 1)	CKIN, FE	4.9	0.01	32.4	0.03	ns
RESET MODE							
tRSET = 16tC2	Reset Pulse Width	C2, RE					
READ CYCLE TIMING—*Figure 2*							
tALE	Address Latch Enable Active	C1, RE	3.3	0.2	16.5	1.1	ns
tDBXv	Address Databus Valid (Note 1)	C2, FE	1.4	0.1	7.2	0.4	ns
tDBXiv	Address Databus Invalid (Note 1)	C1, FE	4.8	0.1	22.2	0.4	ns
tMTB	Bus Read (Note 1)	C2, RE	1.4	0.03	7.0	0.1	ns
			15 pF	**d/pF**	**15 pF**	**d/pF**	
tPAv	Port A Valid (Notes 1, 3)	C2, FE	4.8	0.1	19.8	0.1	ns
tPAiv	Port A Invalid (Notes 1, 3)	C1, FE	5.2	0.1	22.2	0.1	ns
			FO = 1	**d/FO**	**FO = 1**	**d/FO**	
tNRDa	Read Active	MTB, RE	2.3	0.2	11.5	1.1	ns
tNRDia	Read Inactive	C2, FE	2.8	0.2	14.5	1.1	ns
tHBEv	High Byte Enable Active	C2, FE	3.9	0.2	27.9	1.1	ns
tWATAs = 10 ns	Wait Address Setup Time (Note 1)	C2, FE					
tWATAh = tC2 + WS	Wait Address Hold Time (Note 1)	C2, FE					
tRDEXs = 20 ns	Read External Setup Time (Note 1)	C2, FE					
tRDEXL = tCZ + WS	Read External Hold Time (Note 1)	C2, FE					
tSEL8s = 10 ns	8-Bit Mode Setup Time (Note 1)	C2, FE					
tSEL8L = tC2 + WS	8-Bit Mode Hold Time (Note 1)	C2, FE					

HPC Core AC Timing Parameters V_{CC} = 5.0V ±10% (Continued)

Symbol	Description	Min**	Typ**	Max**	Units
READ CYCLE TIMING—*Figures 3 and 4* fc = 17 MHz One Wait State					
tDBAv = ¼tC2−6	Databus Address Valid to ALE Trailing Edge	24			ns
tDBAiv = ¼tC2−5	ALE Trailing Edge to Databus Address Invalid	24			ns
tACCl = tC2 + WS − 15	Address Valid to Input Data Valid—Data Bus		221		ns
tDBDh	Data Bus Data In Hold to MTB Inactive		0		ns
tAMTB = ¼tC2 − 10	ALE Trailing Edge to MTB Active		20		ns
tPAAV = ¼tC2 − 12	Port A Address Valid to ALE Trailing Edge	18			ns
tPAAiv = ¼tC2 − 10	ALE Trailing Edge to Port A Address Invalid	20			ns
tACCE = tC2 + WS = 55	Address Valid to Input Data Valid—Port A		181		ns
tPADs	Port A Data Setup to NRD Inactive		50		ns
tPADh	Port A Data Hold to NRD Inactive		0		ns
tANRD = ¼tCD2 − 5	ALE Trailing Edge to NRD Active	24			ns
tNRDDv = ½tC2 + WS − 65	NRD Active to Port A Data Valid		112		ns
tNRDp = ½tC2 + WS − 10	NRD Pulse Width	167			ns

Symbol	Description	Ref	Best Case*		Worst Case*		Units
			FO = 1	d/FO	FO = 1	d/FO	
WRITE CYCLE TIMING—*Figure 5*							
tNWRa	Write Active	C1, RE	2.3	0.2	14.6	1.1	ns
tNWRia	Write Inactive	C2, RE	3.3	0.2	17.5	1.1	ns

Symbol	Description	Min**	Typ**	Max**	Units
WRITE CYCLE TIMING—*Figures 6* and *7* fc = 17 MHz One Wait State					
tDBDV = ½tC2 + WS − 15	Data Bus Data Out to NWR Inactive	165			ns
tDBDiv	NWR Inactive to Data Bus Data Out Invalid		0		ns
tPADv = ½tC2 + WS − 20	Port A Data Out to NWR Inactive	157			ns
tPADiv = ¼tC2 − 10	NWR Inactive to Port A Data Out Invalid	20			ns
tANWR = ½TC2 − 5	ALE Trailing Edge to NWR Active	54			ns

**Typical times for signal relationship information, only. Actual numbers depend on actual design loading of the signals involved. Parameters referencing Port A include 4 mA I/O buffer on Port A and the delay of control signals brought through I/O buffers to pads.

Symbol	Description	Ref	Best Case*		Worst Case*		Units
			FO = 1	d/FO	FO = 1	d/FO	
READY MODE—*Figure 8*							
tRDYs = 40 ns	Ready Request Setup Time	C2, RE					
tRDYh = 50 ns	Ready Request Hold Time	C2, RE					
HALT/IDLE MODE—*Figure 9*							
tHALT	HALT/IDLE Request Acknowledge (Note 1)	C2, RE	6.4	0.8	31.1	4.3	ns
tNSTP	Stop Oscillator for HALT (Note 1)	C2, RE	5.6	0.4	26.6	2.4	ns
tNSTPia	Restart Oscillator in HALT (Note 1)	NMI, FE	5.6	0.4	26.6	2.4	ns

HPC Core AC Timing Parameters V_{CC} = 5.0V ±10% (Continued)

Symbol	Description	Ref	Best Case*		Worst Case*		Units
			FO = 1	d/FO	FO = 1	d/FO	
HOLD MODE—*Figure 10*							
tNHLDs = 40 ns	Hold Request Setup Time	C2, FE					
tNHLDh = 50 ns	Hold Request Hold Time	C2, FE					
tNHDA	Hold Acknowledge Active	C1, RE	2.4	0.3	12.8	1.6	ns
tPAZ	Port A TRI-STATEs (Note 1, 3)	C1, RE	5.4	0.1	16.2	0.1	ns
SELECT OUTPUTS—*Figure 11*							
tSROM	Address Select Bit	C1, RE	3.9	0.2	20.6	1.1	ns
tSELX	Address Select Bit	C1, RE	3.8	0.2	19.5	1.1	ns
tSELA	Address Select Bit	C1, RE	3.8	0.2	19.5	1.1	ns
tSELB	Address Select Bit	C1, RE	3.8	0.2	19.5	1.1	ns
tSELC	Address Select Bit	C1, RE	3.8	0.2	19.5	1.1	ns
tSELD	Address Select Bit	C1, RE	3.8	0.2	19.5	1.1	ns
tSLIO	Address Select Bit	C1, RE	4.3	0.2	22.7	1.1	ns
WATCHDOG—*Figure 12*							
tWDOT	System Clock to Watch Out (Note 1)	C2, RE	4.1	0.4	21.1	2.4	ns
tSTWDp = 1 tC2	Set Watchdog Pulse Width (Note 1)						
tWDOTp = 16tC2 → 32tC2	WDOT Pulse Width (Note 1)						
TIMERS—INTERNAL CLOCK SOURCE—*Figure 13*							
tT2OT	Clock to Timer 2 Underflow	C2, RE	1.9	0.3	25.8	1.6	ns
tT20Tp = 1tC1 − 20	Timer 2 Underflow Pulse (Note 1)						
tT3OT	Clock to Timer 3 Underflow	C2,RE	1.9	0.3	25.8	1.6	ns
tT30Tp = 1tC1 − 20	Timer 3 Underflow Pulse (Note 1)						
tT0CYp = 1tC2 − 20	Timer 0 Overflow Pulse (Note 1)						
tDIV	Clock to Time Base Generator	C1, RE	2.9	0.3	17.5	1.6	ns

HPC Core AC Timing Parameters V_{CC} = 5.0V ±10% (Continued)

Symbol	Description	Ref	Best Case*		Worst Case*		Units
			FO = 1	d/FO	FO = 1	d/FO	
MICROWIRE/PLUS—MASTER MODE—*Figure 14*							
tOSKO	C1 to MW Master Clock	C1, RE	6.2	0.4	31.7	2.4	ns
tMWOTM	Master Clock to Data Out (Note 1)	OSKO, FE	6.2	0.4	36.7	2.4	ns
tMWINMs = 10 ns	Data Setup Time (Note 1)	C1, RE					
tMWINMh = 50 ns	Data Hold Time (Note 1)	C1, RE					
MICROWIRE/PLUS—SLAVE MODE—*Figure 15*							
tOSKId = 1 C2 Cycle	Data Reg. Load to Slave Clock Begin	SIO Load					
tMWOTS	Slave Clock to Data Out (Notes 1, 2)	OSKI, FE	6.2	0.4	56.7	2.4	ns
tMWINSs = 20 ns	Data Setup Time (Note 1)	OSKI, RE					
tMWINSh = 50 ns	Data Hold Time (Note 1)	OSKI, RE					
TEST LOGIC MUXES (MUX0–MUX8)—*Figure 16*							
tMUXnlh n = 0−5	Output MUX—User Mode (Note 1)		1.79	0.11	5.41	0.39	ns
tMUXnhl n = 0−5	Output MUX—User Mode (Note 1)		1.30	0.13	8.27	0.48	ns
tMUXnlh n = 6−8	Output MUX—User Mode (Note 1)		0.46	0.07	2.53	0.25	ns
tMUXnhl n = 6−8	Output MUX—User Mode (Note 1)		0.58	0.07	3.23	0.24	ns
PORT I INPUTS							
tI1 = 40 ns Minimum	NMI Pulse Width (Note 1)						
tI2 = 40 ns Minimum	I2 Pulse Width (Note 1)						
tI3 = 40 ns Minimum	I3 Pulse Width (Note 1)						
tI4 = 40 ns Minimum	I4 Pulse Width (Note 1)						

Note 1: This is guaranteed by design and not tested.

Note 2: Values assume equivalent loading on OSKO, OSKI and MWOT outputs. Actual delay limits will vary due to specific design loading.

Note 3: Timing includes connection of BF0B4D 4 mA CMOS I/O Buffer to Port A pins.

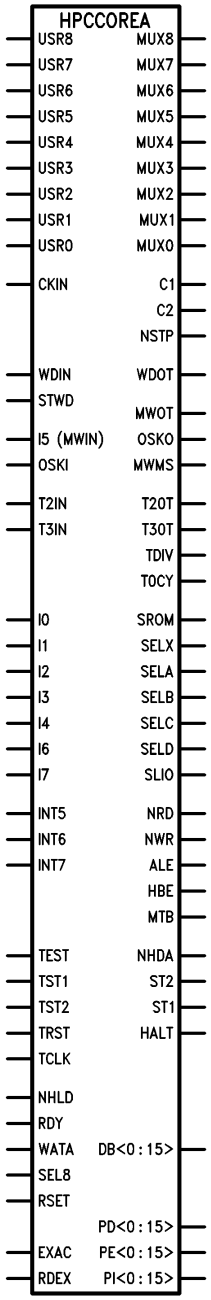
Best Case: Best Process, 5.5V, −40°C.

Worst Case: Worst Process, 4.5V, +100°C.

*Delays are referenced to signal in Ref Column. To relate to CKIN input clock edge, add the appropriate delay from CKIN to the referenced clock (C1, C2).

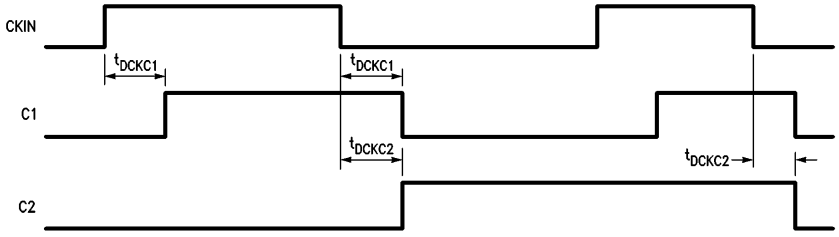
Connection Diagram

Cell Based Library ICON



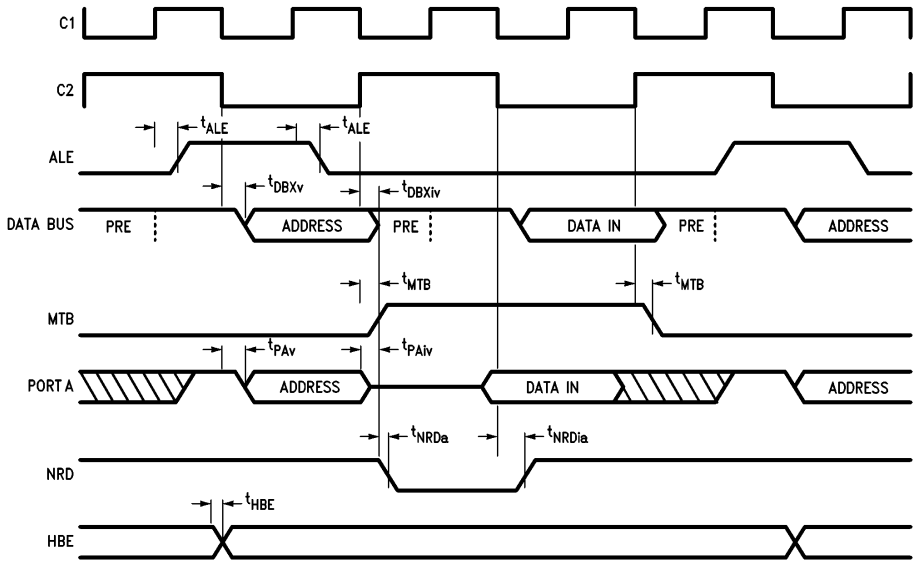
TL/U/9982–2

Timing Waveforms



TL/U/9982–3

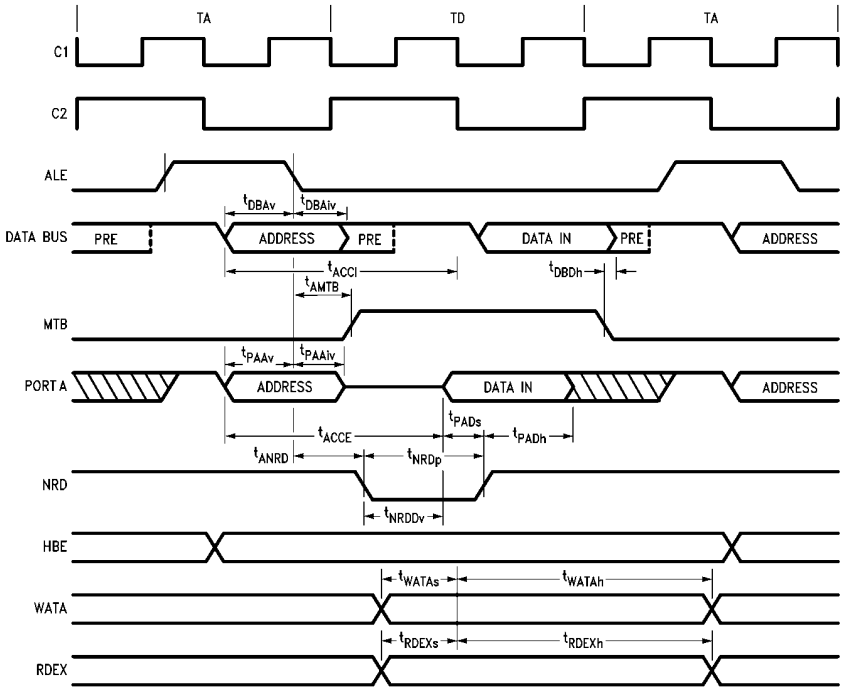
FIGURE 1. Clock Timings



TL/U/9982–28

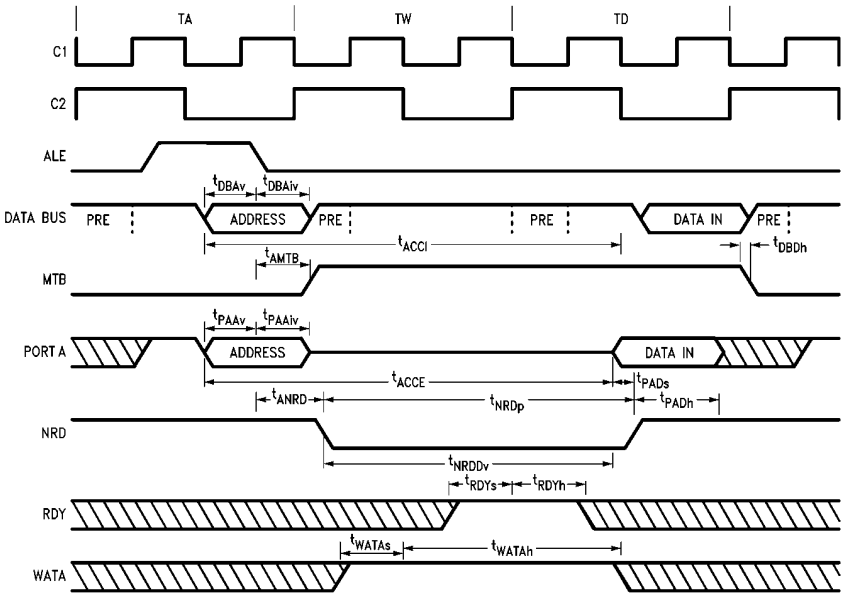
FIGURE 2. Read Cycle No Wait States

Timing Waveforms (Continued)



TL/U/9982–24

FIGURE 3. Read Cycle No Wait States

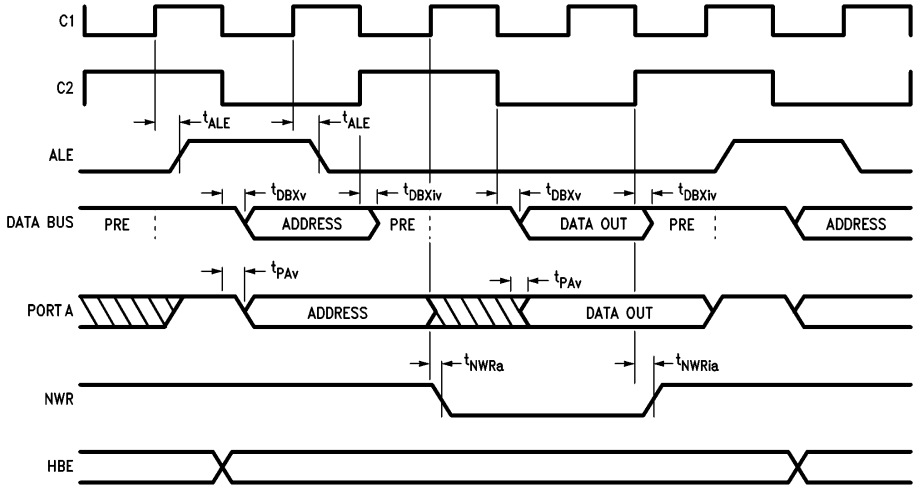


TL/U/9982–29

TA: Address Strobe State
TD: Data Transfer State
TW: Bus Wait State

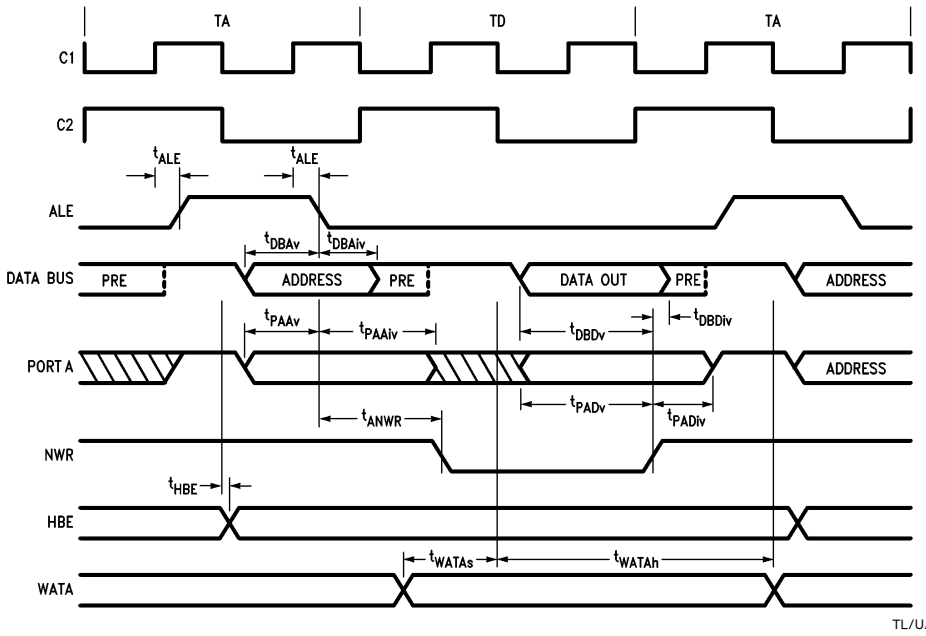
FIGURE 4. Read Cycle One Programmed Wait State

Timing Waveforms (Continued)



TL/U/9982–30

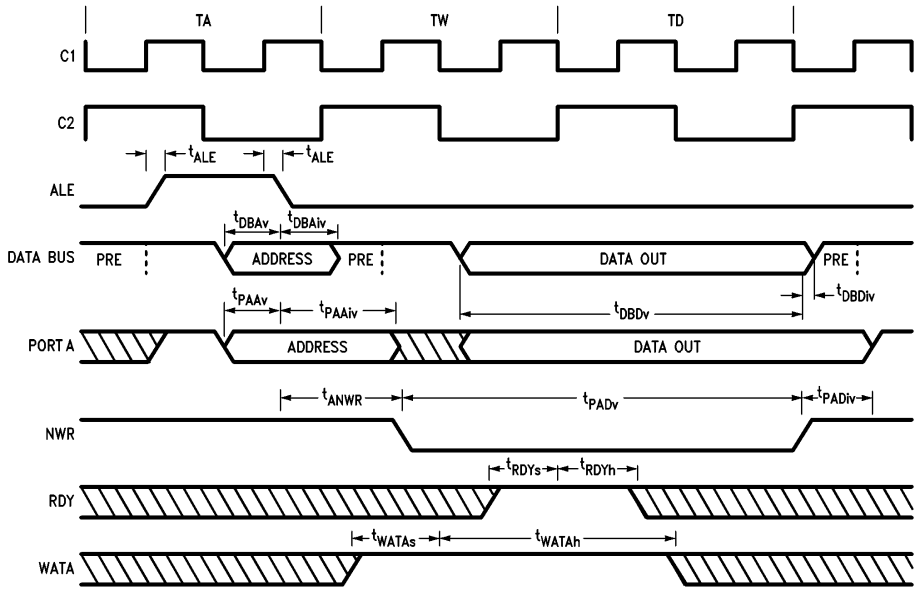
FIGURE 5. Write Cycle No Wait States



TL/U/9982–23

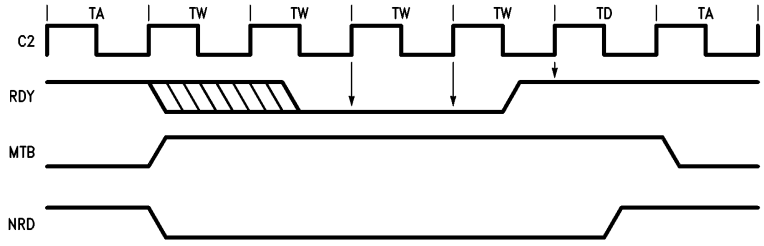
FIGURE 6. Write Cycle No Wait States

Timing Waveforms (Continued)



TL/U/9982–31

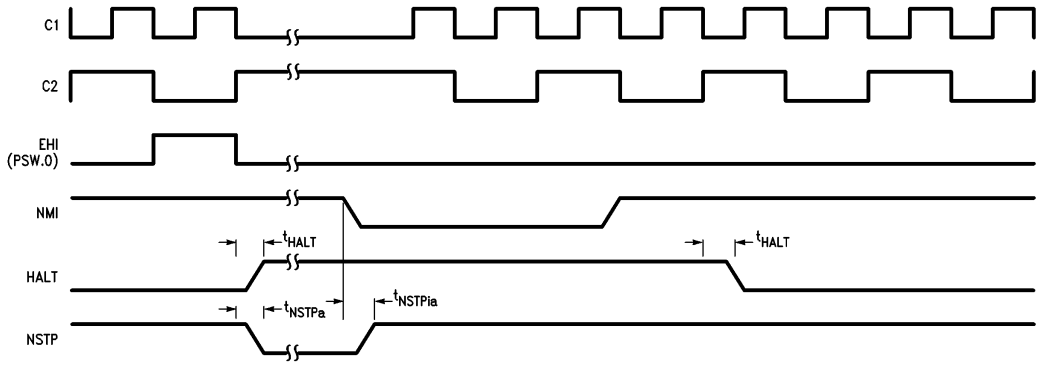
FIGURE 7. Write Cycle One Programmed Wait State



TL/U/9982–32

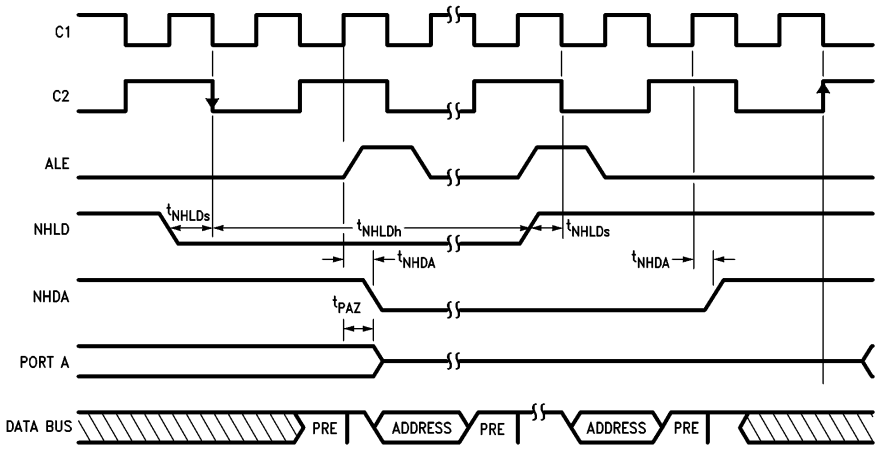
FIGURE 8. Ready Mode Timing Two Pre Programmed Wait States

Timing Waveforms (Continued)



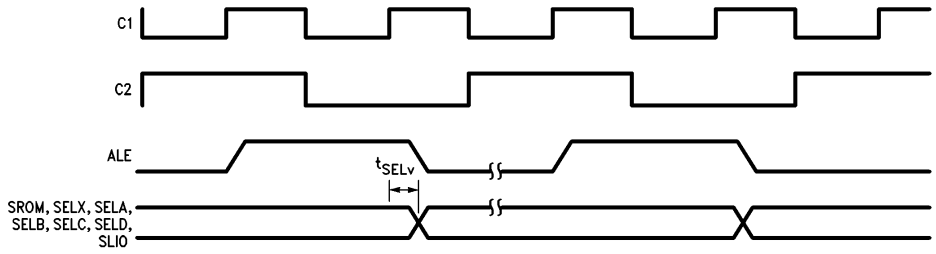
TL/U/9982–33

FIGURE 9. HALT Mode Using NMI to Exit



TL/U/9982–7

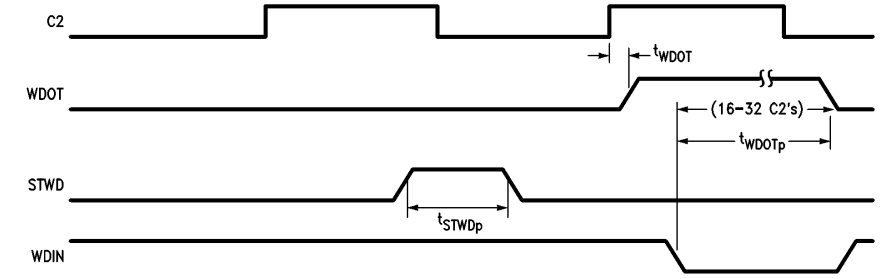
FIGURE 10. Hold Mode Timing



TL/U/9982–25

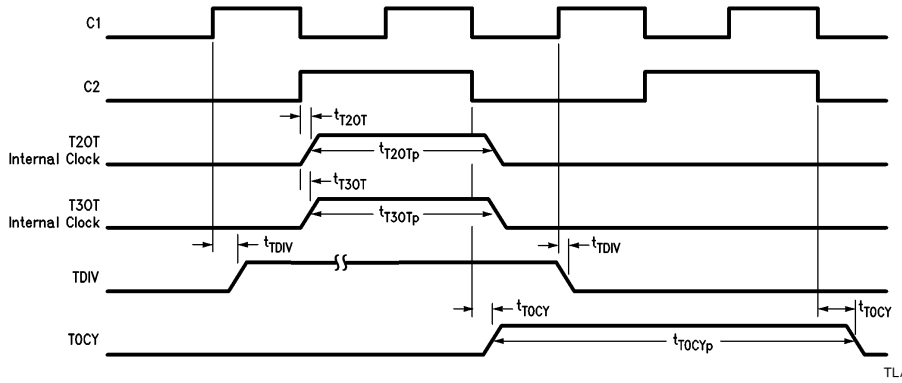
FIGURE 11. Address Block Decode Timings

Timing Waveforms (Continued)



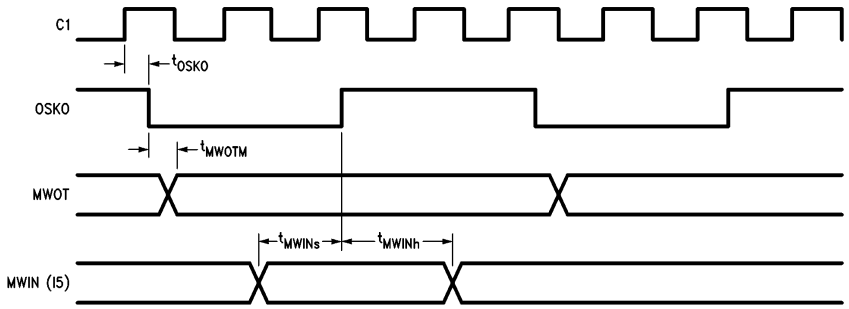
TL/U/9982–8

FIGURE 12. Watchdog Timing



TL/U/9982–26

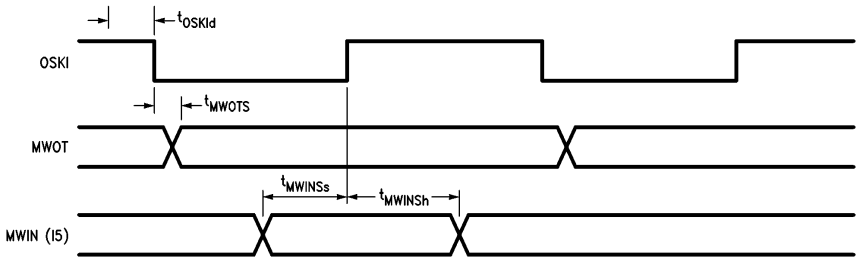
FIGURE 13. Timer Timings



TL/U/9982–11

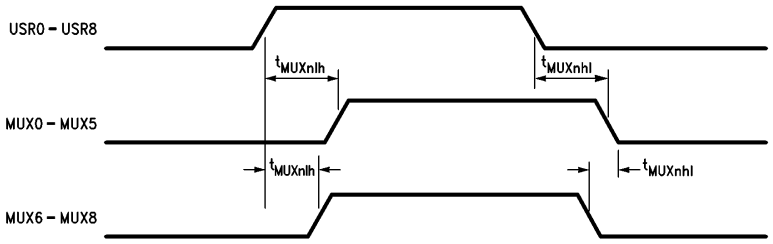
FIGURE 14. MICROWIRE™—Master Mode

Timing Waveforms (Continued)



TL/U/9982–12

FIGURE 15. MICROWIRE™—Slave Mode



TL/U/9982–13

FIGURE 16. Test Logic Muxes

Standard Features

The HPC core is the building block for a number of standard microcontrollers. Since the core is now a building block in a cell based library, microcontroller designs are not restricted to accessing the core solely through I/O ports. As a result, there are several features of the core that are accessable in the ASIC HPC core that are not accessable on the standard microcontrollers built from the core. These features include:

Functional Descriptions

— Direct core access vs. I/O port access.
— Direct access to internal 16-bit address/data bus.
— Ability to control clock oscillations with surrounding hardware.
— Ability to force a watchout condition through surrounding hardware.
— Access to MICROWIRE/PLUS shift clock as individual I/O pins.
— Access to Timer 2 and Timer 3 clocks as individual I/O pins.
— General purpose 4-bit programmable clock output (used by UART in most standard family members).
— Access to Timer 0 overflow output pulse.
— Interrupt bit 5 as a general purpose external interrupt ORed with internal Timer interrupt.
— Access to separate RDY and HOLD input pins.
— Hardware acknowledgement of halt/idle modes (HALT output bit).
— Ability to select which memory space addresses are affected by wait states.
— Direct memory control output bits (NRD, NWR, ALE, HBE).
— Access to hardware control bits to configure, read and acknowledge Port A operations.
— Access to memory map block decode bits.
— Additional registers for test; Feedback control and MUX control.

While the ASIC HPC core offers access to a greater number of design features, it must be kept in mind that the core does not include the peripherals found on the standard parts. Such peripherals would include additional PWM times, UART, A/D, HDLC and DMA channels. Some of these peripherals are available as separate building blocks in the cell based library.

Port A

Port A is a highly flexible 16-bit port used to communicate to peripherals external to the integrated circuit environment. The port may be configured as a 16-bit general purpose I/O port or as an extension of the 16-bit address/data bus. Selection is made through the EXAC pin. Port A cannot be connected to internal logic. It must connect directly to an I/O macro. During test, Port A is configured in the address/data bus mode.

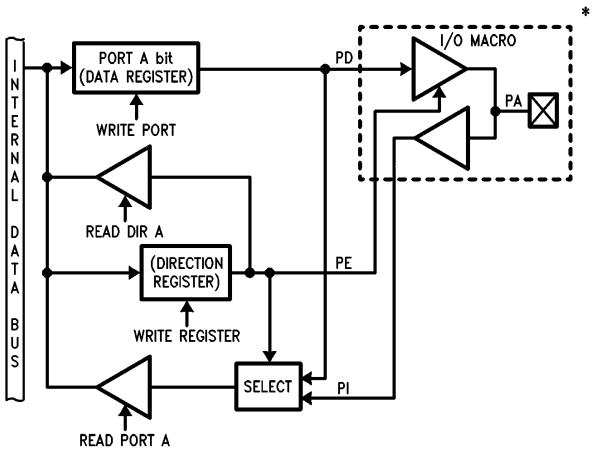
Port A (see *Figure 12*), consists of a data register and a direction register. Both control registers are read/write.

The associated direction register allows the port pins to be individually programmed as inputs or outputs. Port pins selected as inputs are placed in a TRI-STATE® mode by resetting corresponding bits in the direction register. When in the address/data bus extension mode, the direction register is cleared.

A write operation to a port pin configured as an input causes the value to be written into the data register. A read operation returns the value at the pin. Writing to port pins configured as outputs causes the pins to assume the value written into the Port A data register, while reading the pins returns the value of the data register.

Address locations communicating through Port A in the address/data bus mode should be decoded onto the RDEX input to generate a Port A read signal.

Port A may also be configured as an 8-bit bus to support 8-bit applications. In this mode, the upper byte of the 16-bit bus transfers addresses only. The lower byte of the bus is used for both address and data communication. If this mode is selected, address locations communicating through Port A in the 8-bit mode should be decoded onto the SEL8 input.



TL/U/9982–14

*Bidirectional I/O Macro must be selected and connected to Port A I/O pins.

FIGURE 12. Port A: I/O Structure

16-Bit Address/Data Bus

Access to the internal 16-bit address/data bus provides unlimited flexibility. A number of peripherals may be designed to communicate with the bus. To increase the architecture speed, the bus is precharged.

A precharged bus is different from a TRI-STATE bus. The unselected state is a ''1'' rather than a ''Z''. When not selected, the peripheral bit will remain high from the internal precharge circuitry. The same holds true when a high level is requested on the bit. Hence, response time is faster. A low level is placed on the bus when the internal transistor is pulled low. Bus precharge occurs when both clocks C1 and C2 are in a high state. To avoid excessive loading on the address/data bus, special macros have been created for bus interfacing. A bus receiver or bus driver macro, available in the cell library should be used.

Figure 13 demonstrates how a peripheral may be connected to the precharge bus. The peripheral contains registers located in the SELA address block of the memory map. The bus LSBs are monitored through receiver macros and decoded to determine which unique address locations will receive data. Peripheral registers access the data bus through a bus driver macro. Proper timing for communication with the pre-charged bus is controlled with this macro through the MTB and C2 timing signals during a read cycle. During a write cycle, data is transferred from the bus to the peripheral via the bus receiver macros and the NWR signal.

Wait States

The HPC core provides four software selectable Wait States that allow access to slower memories. The Wait States are selected by the state of two bits, Wait1 and Wait0, in the PSW register. Table I indicates the number of programmed wait states. Additionally, the RDY input may be used to extend the instruction cycle, allowing the user to interface with slow memories and peripherals. Hardware decode of the memory map address locations affected by wait states should be provided on the WATA input.

TABLE I

Bit Wait1	Bit Wait0	Wait States
0	0	4
0	1	2
1	0	1
1	1	0

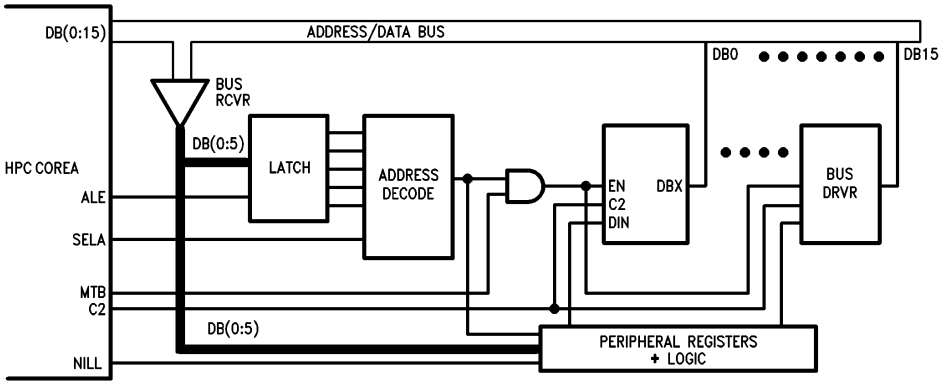
Power Save Modes

Two power saving modes are available on the HPC core: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, all on-board RAM, registers and I/O are unaffected.

HALT Mode

The HPC core is placed in the HALT mode under software control by setting bits in the PSW register. All processor activities, including the clock and timers, are stopped. The stop clock (NSTP) output is gated with the clock input (CKI) signal to inhibit clock oscillation at the core input when a clock I/O oscillator macro is used. The HALT output goes high designating that the core is in HALT mode. In this mode, power requirements for the HPC are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. V_{CC} may be decreased as low as the RAM keep alive voltage.

There are two ways of exiting the HALT mode: via the RSET or NMI interrupts. The RSET input reinitializes the processor. Exiting with an NMI input will generate a vectored interrupt. Operation will resume from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.



TL/U/9982–27

FIGURE 13. Peripheral Connections on the Address/Data Bus

Idle Mode

The HPC core is placed in the IDLE mode through the PSW register. In this mode, all processor activity, except the on-board oscillator, WATCHDOG monitoring and Timer T0, is stopped. The HALT output bit goes high to indicate the processor is entering the IDLE mode. As with the HALT mode, the processor is returned to full operation by the RSET or NMI inputs, but without waiting for oscillator stabilization. A timer T0 overflow will also cause the HPC core to resume normal operation.

Interrupts

Complex interrupt handling is easily accommodated by the HPC core's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table II.

TABLE II. Interrupts

Vector Address	Interrupt Source	Arbitration Ranking
FFFF:FFFE	RESET	0
FFFD:FFFC	Nonmaskable External on Rising Edge of I1 Pin	1
FFFB:FFFA	External on I2 Pin	2
FFF9:FFF8	External on I3 Pin	3
FFF7:FFF6	External on I4 Pin	4
FFF5:FFF4	Timeout of Internal Timers ORed W/External On INT5 Pin	5
FFF3:FFF2	External on INT6 Pin	6
FFF1:FFF0	External on INT7 Pin	7

Interrupt Arbitration

The HPC core contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. The arbitration ranking is shown in Table II. The interrupt on reset has the highest rank and is serviced first.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the RSET, which is serviced immediately. RSET, INT6 and INT7 are active high level sensitive interrupts. INT5 is an external active high level sensitive interrupt ORed with internal interrupts from timers T0 through T3. The timer interrupts have individual enable bits. INT5 is enabled whenever interrupt bit 5 is high in the ENIR register. All other interrupts are edge sensitive. NMI is positive edge sensitive. The external interrupts on I2, I3 and I4 can be software selected to be sensitive on a rising or falling edge.

Interrupt Control Registers

The HPC core allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

Interrupt Enable Register (ENIR)

RSET and the external interrupt on I1 (NMI) are nonmaskable interrupts. The other interrupts can be individually enabled and disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to be serviced, both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

Interrupt Pending Register (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupt may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts I2–I4 are normally cleared by the HPC core after acknowledging the interrupts.

For the interrupts generated internally and on bits INT5, INT6 and INT7, the user has the responsibility of resetting the interrupt pending flags through software.

The NMI bit is read only and I2, I3 and I4 are designed to only allow a zero to be written to the pending bit (writing a one has no affect). A LOAD IMMEDIATE instruction is to be the only instruction used to clear a bit or bits in the IRPD register. This allows a mask to be used, thus ensuring that the other pending bits are not affected.

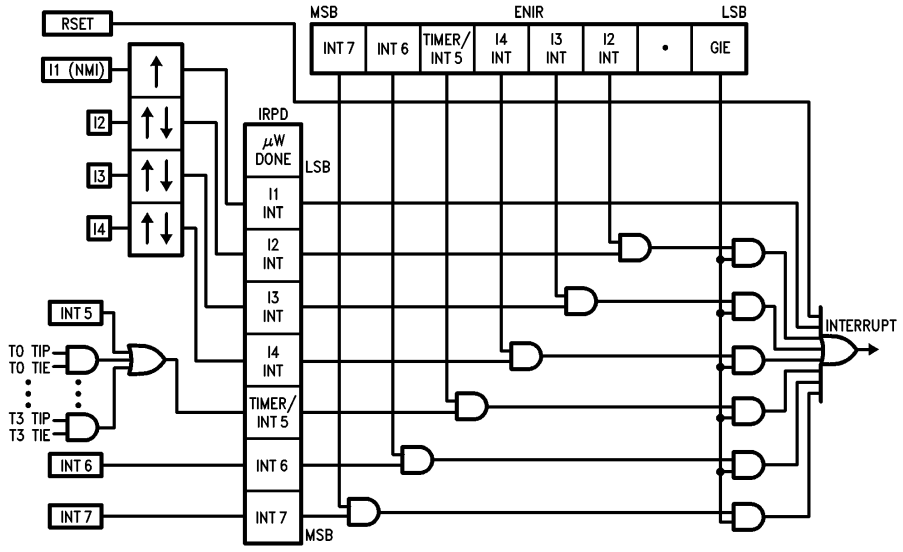
Interrupt Condition Register (IRCD)

Three bits of the IRCD register select the input polarity of the external interrupts on I2, I3 and I4.

Servicing the Interrupts

The interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable bit (GIE) is copied into the CGIE bit of the PSW register, it is then reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack and re-enables interrupts, or RET to just pop the stack, and then returns to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. *Figure 13* shows the Interrupt Enable Logic.

Servicing the Interrupts (Continued)



TL/U/9982–15

FIGURE 13. Block Diagram of Interrupt Logic

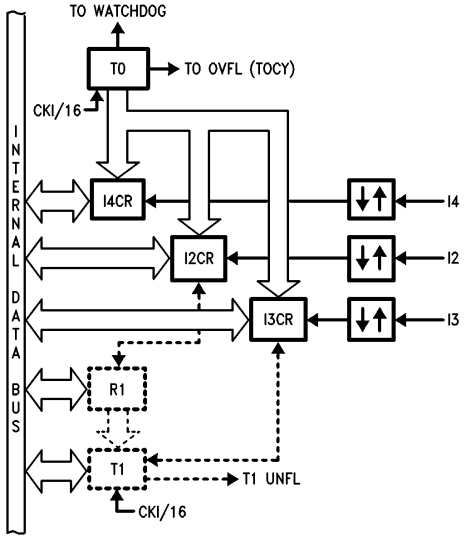
Reset

The RSET input initializes the processor and sets Port A in the TRI-STATE condition. The internal data bus oscillates at a C2 clock rate is an active high signal. If RSET is connected to logic outside of the device it should be brought in through a schmitt trigger input. The processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location.

Timer Overview

The HPC core contains a powerful set of flexible timers enabling the HPC to perform extensive timer functions not usually associated with microcontrollers.

The core contains four 16-bit timers. Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock/16) rate. It is used for Watchdog logic, high speed event capture and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. In addition, the overflow of T0 is a core output, T0CY, and may be utilized by any external logic. Upon timer overflow a pulse of width CKI/2 occurs on T0CY. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR and I4CR. A control bit in the register T0CON configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR and I4CR respectively, record the value of timer T0 when specified events occur on the interrupt pins I2, I3 and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see *Figure 14*).



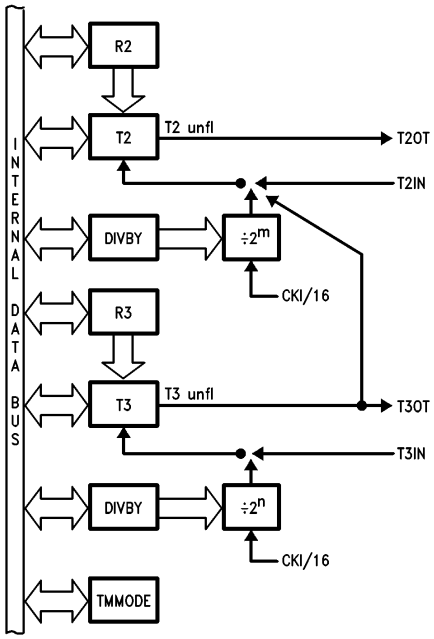
TL/U/9982–16

FIGURE 14. Timers T0–T1 Block

Timer Overview (Continued)

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, T2IN and T3IN, respectively, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see *Figure 15*).

The timers T1 through T3 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register is automatically loaded into the timer. An underflow on timers T2 and T3 generates an output pulse on pins T2OT and T3OT, respectively. The pulse is one CKI width.



TL/U/9982–17

FIGURE 15. Timers T2–T3 Block

Timer Registers

There are three control registers designated for timer programming. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch and enable interrupts from timers T0 through T3. The T0/capture configuration register (T0CON) determines if (CKIN) capture registers I2CR and I3CR are to be used as timer registers R1 and T1 or as capture registers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. In addition, four bits are available to control a general purpose programmable clock brought out on pin TDIV.

Timer Applications

The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC core.

Frequencies can be generated by using the timer/register pairs with a toggle flip-flop. A programmable waveform may be created by connecting the timer underflow pulse output to the flip-flop clock. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.

TL/U/9982–18

FIGURE 16. Square Wave Frequency Generation

Watchdog Logic

The Watchdog Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the Watchdog logic are potentially infinite loops and illegal addresses. Should the Watchdog register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts of CKI/16, an infinite loop condition is assumed to have occurred. An illegal condition also occurs when the processor generates an illegal address. The designer specifies which address locations are illegal through the STWD input. As well, external logic may use the STWD input to declare a watchout, forcing the Watchdog logic to trigger for an illegal application specific condition.

Any illegal condition forces the Watchdog output (WDOT) pin high. The WDOT signal must trigger a low level on the WDIN input for Watchdog timeout to begin. This can occur from one of several ways. If the Watchdog feature is being used external to the integrated circuit environment a special watchdog I/O macro must be used. The macro has an external open drain output, active low. An internal output, active low should connect back to the WDIN input on the core, triggering a watchdog timeout. If the watchdog feature is contained inside the IC, WDOT may be inverted and brought back in to WDIN. The watchout signal can be connected to the RSET or NMI interrupt input logic or be brought out for external logic usage.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see *Figure 17*). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using 15 (MWIN) as the input and MWOT as the output. The clock signal for the serial shift register (SIO) can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register and appears as a core output on OSKO. An external clock rate may be selected on the OSKI input of the core. Usage of OSKI or OSKO as the shift clock is determined through the IRCD register. A DONE flag indicates when the data shift is completed. The MICROWIRE/PLUS capability enables the core to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., A/D converters, display drivers, EEPROMs).

MICROWIRE/PLUS (Continued)

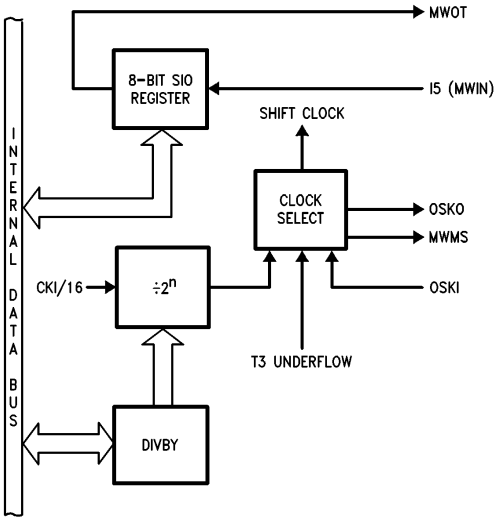


FIGURE 17. MICROWIRE/PLUS

MICROWIRE/PLUS Operation

The HPC core can enter the MICROWIRE/PLUS mode as a master or slave. A control bit in the IRCD register determines whether the HPC core is the master or slave mode.

An externally generated shift clock placed on the OSKI pin is used when the HPC is configured as a slave. The OSKO output is placed into TRI-STATE during this operation. The shift clock is generated internally when the HPC is configured as a master. The DIVBY register programs the frequency of the OSKO clock. This register allows the OSKO clock frequency to be programmed in 14 selectable steps from 122 Hz to 1 MHz with CKI at 16.0 MHz, or from a timer T3 underflow. Hardware mode detection is provided on the MICROWIRE master/slave output MWMS.

The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is clocked out on the falling edge of the shift clock. Serial data on the MWIN pin is clocked in on the rising edge of the shift clock.

Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is multiprocessing applications where two CPUs share a common memory block. The HPC core supports shared memory access through Port A with two pins. These pins are the NHLD input pin and the NHDA output pin.

The host uses DMA to interface with the HPC core. A low level on the NHLD input of the HPC core from the host initiates a data transfer. In response, the HPC core places Port A in a TRI-STATE mode, freeing it for use by the host. The host waits for the acknowledge signal (NHDA) from the HPC core indicating that the port is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the NHLD request and the HPC core resumes normal operations.

Figure 18 illustrates an application of the shared memory interface between the HPC core and a series 32000 system.

Memory

The HPC core has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed. The core contains 256 bytes of RAM, usable for instruction execution. This 256 byte RAM block is available as a stand alone macro for inclusion of additional memory.

Program memory addressing is controlled by the 16-bit program counter on a byte basis. Memory can be addressed directly by instruction or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always addressed on even-byte boundries. The HPC core uses memory-mapped organization to support registers, I/O, and on-chip peripheral functions.

The HPC core memory address space extends to 64 kbytes. Registers and I/O are mapped as shown in Table III.

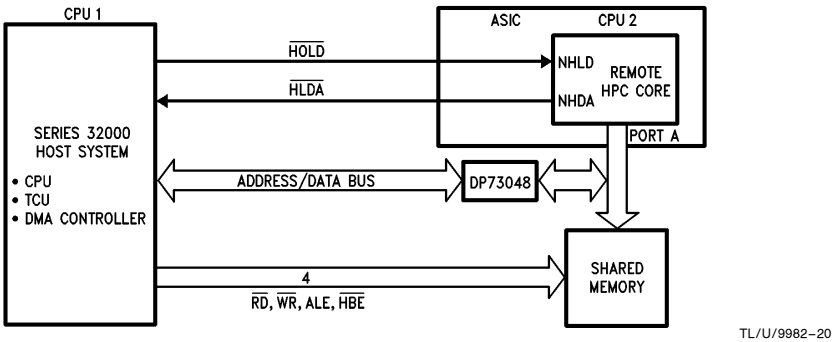


FIGURE 18. Shared Memory Application: HPC Core Interface to Series 32000 System

Memory (Continued)

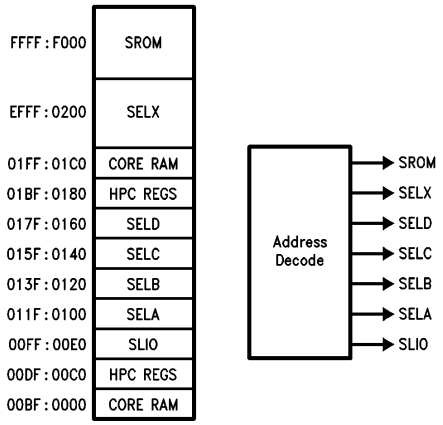
TABLE III. HPC Core Memory Map

FFFF : FFF0	Interrupt Vectors	User Memory
FFEF: FFD0	JSRP Vectors	
FFCF : FFCE : : : : 0201 : 0200	User Memory	
01FF : 01FE : : 01C1 : 01C0	Core RAM	User RAM
0195 : 0194	Watchdog Register	Watchdog Logic
0192 0191 : 0190 018F : 018E 018D : 018C 018B : 018A 0189 : 0188 0187 : 0186 0185 : 0184 0183 : 0182 0181 : 0180	T0C0N Register TMMODE Register DIVBY Register T3 Timer R3 Register T2 Timer R2 Register I2CR Register/R1 I3CR Register/T1 I4CR Register	Timer Block T0 : T3
017F : 017E : : 0101 : 0100	Register Expansion	
00FF : 00FE 00FC	Feedback Register MUX Control Register	Test Control Registers
00F5 : 00F4 00F3 : 00F2 00F1 : 00F0 00E6 00E3 : 00E2 00E1 : 00E0	(Reserved) (Reserved) DIR A Register (Reserved) (Reserved) Port A	Port A Registers
00DE 00DD : 00DC 00D8 00D6 00D4 00D2 00D0	Microcode ROM Dump Halt Enable Register Port I Input Register SIO Register IRCD Register IRPD Register ENIR Register	Port Control & Interrupt Control Registers
00CF : 00CE 00CD : 00CC 00CB : 00CA 00C9 : 00C8 00C7 : 00C6 00C5 : 00C4 00C3 : 00C2 00C1 : 00C0	X Register B Register K Register A Register PC Register SP Register (Reserved) PSW Register	HPC Core Registers
00BF : 00BE : : 0001 : 0000	Core RAM	User RAM

Address Decode

Address decode is a common function required when accessing memory. To aid in this function a group of 7 memory access decode signals are available. The 64k memory map has been divided into eleven uneven blocks. A decode signal points to a respective block of user space. The remaining four blocks are accessed by internal register or RAM memory. The latched address select bits may be used with the required number of LSBs to decrease the amount of circuitry necessary to decode a unique address.

Figure 19 shows the signal decode mapping. The select bit SLIO points to a 32 byte block of memory containing Port A registers and test registers. The remaining bytes in this block are intended for future expansions. However, they may be utilized as write only register locations, if desired.



TL/U/9982–21

FIGURE 19. Memory Map Partitioning

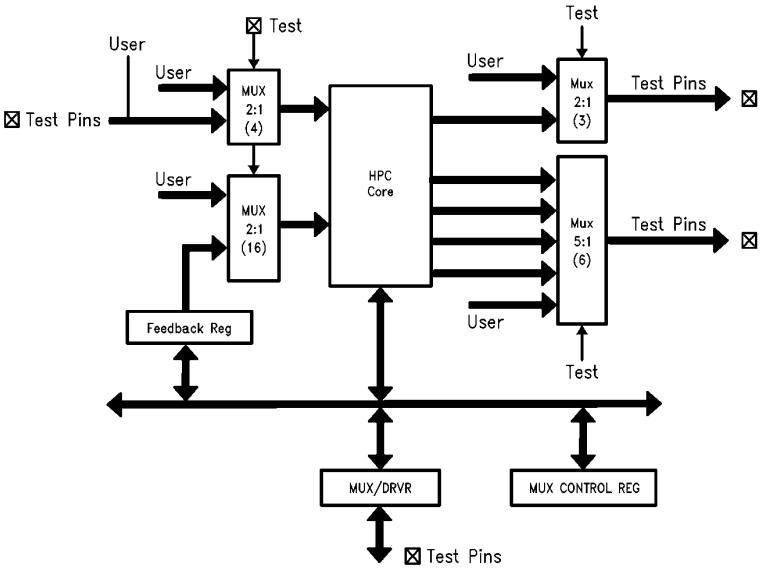
Test Logic

The HPC core contains built in test logic to aid in device testing. This logic permits test vectors created for the standard family of products to be run while testing the Megacell inside a cell based design. Therefore, test vector generation to validate this complicated portion of the integrated circuit is removed from the circuit designer. Modifications may not be made to the test logic. Common to the strategy for all cell based Megacells, the test logic follows the ''parallel testing'' methodology.

The parallel testing methodology isolates the core from the remainder of the standard cell circuitry. This is accomplished by surrounding the core with banks of multiplexers (see *Figure 20*). Respective multiplexer inputs and outputs are brought to I/O pins of the device. Thus, during test the HPC core appears as the only logic in the chip. Thirty I/O pins are utilized by the HPC core during test. Twenty-nine of these pins may be used for alternate functions when not in the test mode. Sixteen pins are dedicated to Port A operation. The TEST pin determines which stimulus will reach the core and I/O pins. This is the only I/O pin dedicated to the test circuitry.

Two software registers assist the hardware during test. An 8-bit MUX control register (address 00FD) and a 16-bit feedback control register (address 00FE:00FF) are referred to as MUXC and FDBK, respectively in the following text.

Three groups clearly define the test pins. Nine pins are outputs, 16 pins are bidirectional, and 5 pins are inputs. The 16 bidirectional pins are part of Port A. Port A is configured as an address/data bus accessing the SROM address space when in test mode. With the aid of 5:1 multiplexers and the MUX Control register, the excessive number of outputs on the HPC core have been brought out through 6 pins. The outputs have been logically combined into four groups of six outptus each. In conjunction with the TEST pin and the MUXC register for group selection, each group is probed separately. The ALE, NRD and NWR signals are brought out through 2:1 MUXes.



TL/U/9982–22

FIGURE 20. HPC Core Test Logic

Test Logic (Continued)

The 16-bit FDBK register has been created to control input stimulus through the HPC software. A binary pattern is loaded into the FDBK register and then shifted onto the HPC inputs. Four of the inputs do not lend themselves to a simple software solution for test. Therefore, the hardware has been provided to access these pins through direct multiplexer stimulation. An input test pin is shared by T2IN and T3IN. Selection of the timer input is controlled by the MUXC register bit TS2. The RDY and HOLD inputs share a test pin, as well. The mode is controlled internally by a HOLDMODE bit. CKIN and RSET are brought in through 2:1 MUXes.

The test logic hardware is part of the HPC macro and may not be altered. When not in test mode, logic connected to the user side of the isolating multiplexers will stimulate the core and I/O pins. The USR0:USR8 inputs on the HPC macro are the user logic inputs for the 9 MUXes connected to the output side of the HPC. The designer is responsible for connecting the appropriate I/O macros to the required macro test pins.

HPC Core CPU

The HPC core CPU has a 16-bit ALU and six 16-bit registers:

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

Addressing Modes—Accumulator as Destination
Register Indirect

This is the ''normal'' mode of addressing for the HPC core (instructions are single byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

Addressing Modes—Direct Memory as Destination
Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- of 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect Using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action
ARITHMETIC INSTRUCTIONS		
ADD	Add	MA + Meml → MA Carry → C
ADC	Add with Carry	MA + Meml + C → MA Carry → C
ADDS	Add Short Imm8	A + Imm8 → A Carry → C
DADC	Decimal Add with Carry	MA + Meml + C → MA (Decimal Carry → C
SUBC	Subtract with Carry	MA − Meml + C → MA Carry → C
DSUBC	Decimal Subtract w/Carry	MA − Meml + C → MA (Decimal Carry → C
MULT	Multiply (Unsigned)	MA*Meml → MA & X, 0 → K, 0 → C
DIV	Divide (Unsigned)	MA/Meml → MA, Rem. → X, 0 → K, 0 → C
DIVD	Divide Double Word (Unsigned)	X & MA/Meml → MA, Rem → X, 0 → K, Carry → C
IFEQ	If Equal	Compare MA & Meml, Do Next if Equal
IFGT	If Greater Than	Compare MA & Meml, Do Next if MA > Meml
AND	Logical AND	MA AND Meml → MA
OR	Logical OR	MA OR Meml → MA
XOR	Logical Exclusive-OR	MA XOR Meml → MA
MEMORY MODIFY INSTRUCTIONS		
INC	Increment	Mem + 1 → Mem
DECSZ	Decrement, Skip if 0	Mem − 1 → Mem, Skip Next if Mem = 0
BIT INSTRUCTIONS		
SBIT	Set Bit	1 → Mem. Bit (Bit is 0 to 7 Immediate)
RBIT	Rset Bit	0 → Mem. Bit
IFBIT	If Bit	If Mem Bit is True, Do Next Instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	Meml → MA
	Load, Incr/Decr X	Mem(X) → A, X ± 1 (or 2) → X
ST	Store to Memory	A → Mem
X	Exchange	A ⟷ Mem
	Exchange, Incr/Decr X	A ⟷ Mem(X), X ±1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP + 2 → SP
POP	Pop Stack to Memory	SP − 2 → SP, W(SP) → W
LDS	Load A, Incr/Decr B, Skip On Condition	Mem(B) → A, B ±1 (or 2) → B, Skip Next if B Greater/Less than K
XS	Exchange Incr/Decr B, Skip On Condition	Mem(B) ⟷ A, B ±1 (or 2) → B, Skip Next if B Greater/Less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B Immediate	Imm → B
LD K	Load K Immediate	Imm → K
LD X	Load X Immediate	Imm → X
LD BK	Load B and K Immediate	Imm → B, Imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INC A	Increment A	A + 1 → A
DEC A	Decrement A	A − 1 → A
COMP A	Complement A	1's Complement of A → A
SWAP A	Swap Nibbles of A	A 15:12 ← A 11:8 ← A 7:4 ⟷ A 3:0
RRC A	Rotate A Right Thru C	C → A15 → . . . → A0 → C
RLC A	Rotate A Left Thru C	C ← A15 ← . . . ← A0 ← C
SHR A	Shift A Right	0 → A15 → . . . → A0 → C
SHL A	Shift A Left	C ← A15 ← . . . ← A0 ← 0
SC	Sec C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do Next If C = 1
IFNC	IF Not C	Do Next if C = 0

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump Subroutine from Table	PC ⟶ W(SP), SP + 2 ⟶ SP W(Table #) ⟶ PC
JSR	Jump Subroutine Relative	PC ⟶ W(SP), SP + 2 ⟶ SP,PC + # ⟶ PC (# is +1025 to − 1023)
JSRL	Jump Subroutine Long	PC ⟶ W(SP), SP + 2 ⟶ SP,PC + # ⟶ PC
JP	Jump Relative Short	PC + # ⟶ PC (# is +32 to −31)
JMP	Jump Relative	PC + # ⟶ PC (# is + 257 to −255)
JMPL	Jump Relative Long	PC + # ⟶ PC
JID	Jump Indirect at PC + A	PC + A + 1 ⟶ PC
JIDW		then Mem(PC) + PC ⟶ PC
NOP	No Operation	PC + 1 ⟶ PC
RET	Return	SP − 2 ⟶ SP,W(SP) ⟶ PC
RETSK	Return Then Skip Next	SP − 2 ⟶ SP,W(SP) ⟶ PC, & Skip
RETI	Return from Interrupt	SP − 2 ⟶ SP,W(SP) ⟶ PC, Interrupt Re-Enabled

Note: W is 16-Bit word of memory
MA is Accumulator A or direct memory (8- or 16-bit)
Mem is 8-bit byte or 16-bit word of memory
Meml is 8- or 16-bit memory or 8- or 16-bit immediate data
Imm is 8-bit or 16-bit immediate data
Imm8 is 8-bit immediate data only

Memory Usage

Number of Bytes for Each Instruction (Number in Parenthesis is 16-Bit Field)

	Using Accumulator A						To Direct Memory			
	Reg Indir		Direct	Indir	Index	Immed.	Direct		Immed.	
	(B)	(X)					*	**	*	**
LD	1	1	2(4)	3	4(5)	2(3)	3(5)	5(6)	3(4)	5(6)
X	1	1	2(4)	3	4(5)	—	—	—	—	—
ST	1	1	2(4)	3	4(5)	—	—	—	—	—
ADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADDS	—	—	—	—	—	2	—	—	—	—
SBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DSBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADD	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
MULT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIV	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIVD	1	2	3(4)	3	4(5)	—	4(5)	5(6)	4(5)	5(6)
IFEQ	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFGT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
AND	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
OR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
XOR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)

*8-Bit Direct Address
**16-Bit Direct Address

Instructions that Modify Memory Directly

	(B)	(X)	Direct	Indir	Index	B&X
SBIT	1	2	3(4)	3	4(5)	1
RBIT	1	2	3(4)	3	4(5)	1
IFBIT	1	2	3(4)	3	4(5)	1
DECSZ	3	2	2(4)	3	4(5)	
INC	3	2	2(4)	3	4(5)	

Immediate Load Instructions

	Immed
LD B,*	2(3)
LD X,*	2(3)
LD K,*	2(3)
LD BK,*, *	3(5)

Memory Usage (Continued)

Register Indirect Instructions with Auto Increment and Decrement

Register B with Skip		
	(B+)	(B−)
LDS A,*	1	1
XS A,*	1	1

Register X		
	(X+)	(X−)
LD A,*	1	1
X A,*	1	1

Instruction Using A and C

CLR	A	1
INC	A	1
DEC	A	1
COMP	A	1
SWAP	A	1
RRC	A	1
RLC	A	1
SHR	A	1
SHL	A	1
SC		1
RC		1
IFC		1
IFNC		1

Transfer of Control Instructions

JSRP	1
JSR	2
JSRL	3
JP	1
JMP	2
JMPL	3
JID	1
JIDW	1
NOP	1
RET	1
RETSK	1
RETI	1

Stack Reference Instructions

	Direct
PUSH	2
POP	2

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the less memory support required. More integration is possible if the die is not consumed with memory or for external memory designs, production cost is lower with fewer memory elements.

The HPC core has been designed to be extremely code-efficient. Standard products developed around the core look very good in all the standard coding benchmarks. Many large jobs have been programmed using the HPC core, and the code savings over other popular microcontrollers has been considerable-often the jobs take less than one-half the memory.

Reasons for this saving of code include the following:

Single Byte Instructions

The majority of instructions on the HPC core are single byte.

Two instructions are particularly code-efficient:

JP is a 1-byte jump. The jump must be within a range of plus or minus 32 bytes. This is a valuable instruction, since many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JRSP is a 1-byte subroutine call. The user makes a table of his 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into his table; the assembler can give him this information.

Efficient Subroutine Calls

The 2-byte JSR instruction can call any subroutine within plus or minus 1k of program memory.

Multifunction Instructions for Data Movement and Program Looping

The HPC core has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory location pointed to by the B register.
2. Increment the B register.
3. Compare the B register versus the K register.
4. Generate a conditional skip if B is greater than K.

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

Bit Manipulation Instructions

Any bit of memory, I/O or register can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

The one exception to the above is with the IRPD register. A LOAD IMMEDIATE instruction is to be the only instruction used to clear a bit or bits in this register (see Interrupt Pending Register section).

Decimal Add and Subtract

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the progarmmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC core supplies 8-bit byte capability for 2-digit variables and literal variables.

Multiply and Divide Instructions

The HPC core has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling base conversion, computing indexes of arrays, etc.

Development Support

Schematic Capture

Schematic capture of designs using the HPC Megacell is available on ASIC supported workstations and the mainframe computer system. Confer with Marketing for the library revision which contains the HPC Megacell on individual workstations.

Software Tools

Several software tools are available to aid the design process. An event driven simulation software package is available on the mainframe computer system. A behavioral model of the HPC core is present to support netlist simulation.

Writing HPC software in a test vector format is very cumbersome. To alleviate this struggle, a software package exists to convert HPC Assembler code to a format familiar to the simulator. Along with the HPC code conversion, I/O pin stimulus may be included to execute synchronously with a designated HPC instruction. The conversion software creates a HILO ROM model to hold the HPC code and an I/O stimulus model for the input pin test vectors.

Simulation Procedure

An important part of cell based design is simulation. To confirm proper HPC connection and design verification, HPC code for device testing will be written by the designer. Typically, this code will not be the code written to execute the application. Rather, it will be specifically written to exercise the design for test purposes in the most efficient manner as possible. Instead of writing HPC code in a binary format, it is recommended that the HPC assembler to simulation test format conversion software be utilized.

When using the conversion software approach, a superlevel drawing of the cell based design must be created. This drawing page is a diagram of the standard cell device, the test ROM model and the I/O pin stimulus model. The ROM model may be generated as an internal or external ROM for the cell based device. This schematic page shows the interaction between these three blocks. A netlist from this level is used for the simulation. Thus, the stimulus models appear in the design netlist. When simulation is complete, only the netlist of the cell based design is considered for place and route. Refer to the HPC Megacell user's guide for more details on HPC simulation.

Emulation

The MOLE™ (Microcontroller On-Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs, TMP, 8050U and the HPC Family of products. The MOLE consists of a Brain Board, Personality Board and optional host software.

The purpose of a MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller, and assist in both the software and hardware debugging of the system. It is a self-contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

The MOLE contains three serial ports. Multiple ports are usually needed to optionally connected to a terminal, a host system, a printer or modem, or to other MOLEs in a multi-MOLE environment. MOLE can be used in either a stand alone mode or in conjunction with selected host systems, i.e., those using CP/M or PC-DOS, communicating via RS-232 port.

The MOLE product line may be used with cell based IC development of HPC designs. A 68-pin package of the HPC core Megacell has been created to operate in the HPC Personality Board. When a standard product HPC design is emulated, a connection between the MOLE board and the user's target system is made. This allows interaction between the emulator and the application system. In a cell based design, the integrated circuit will contain more than the HPC core. To accurately emulate the core in the user's system a bread board of the logic surrounding the HPC core must be created. The MOLE board would connect directly to the bread board, which would connect into the target system. The MOLE system will perform the same function as done in the emulation of the standard family product.

To improve the accuracy of bread board emulations National will provide ''kit parts'' of all cell macros not available as standard products.

Dial-A-Helper is a service provided by the MOLE applications group. If a user is having difficulty in getting a MOLE to operate in a particular mode or it is acting in a peculiar manner, they can contact us via their system and modem. They can leave messages on our electronic bulletin board which we will respond to, or they can arrange for us to actually take control of their system via modem for debugging purposes.

The applications group can then force their system to execute various commands and try to resolve the customer's problem by actually getting the customer's system to respond. The problem is solved 99% of the time. This allows us to respond in minutes instead of days when applications help is needed.

The system can also be used to download available applications software.

Packaging

The HPC core Megacell requires a minimum of 30 I/O pins for testing purposes. In addition, 4 power pairs are required. Therefore, the cell based design may be put in any available package greater than 38 pins.

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408