HARRIS

RTX 2000™

May 1989

Real Time Express™ 16-Bit Microcontroller

Features

- Fast 100ns Machine Cycle
- Single Cycle Instruction Execution
- Direct Execution of Forth Language
  - Eliminates Assembly Language Programming
- Single Cycle 16-bit Multiply
- Fast Division, Square Root
- Single Cycle Subroutine Call/Return
- Four Cycle Interrupt Latency
- Two On-Chip 256 Word Stacks
- On-Chip Interrupt Controller
- Three On-Chip 16-bit Timer/Counters
- ASIC Bus™ for Off-Chip Extension of Architecture
- 1 Megabyte Total Address Space
- Word and Byte Memory Access
- Low Power CMOS .... 5mA/MHz Typical
- Fully Static .... D.C. to 10MHz Operation
- 84-Pin PGA and 84-Lead PLCC Package
- Available in Harris Standard Cell Library

Description

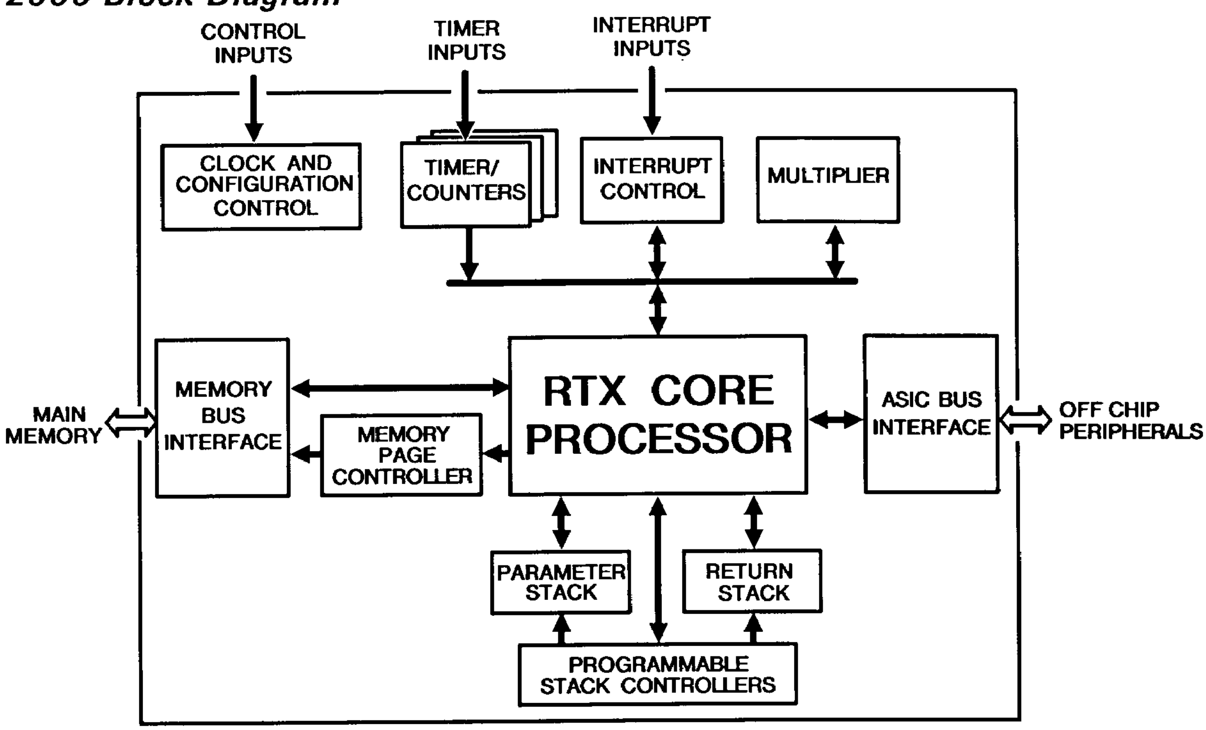
The RTX 2000 is a high performance 16-bit microcontroller with on-chip timers, interrupt controller, and multiplier. A unique feature of this processor is the high performance ASIC Bus, which provides for architecture extension using off-chip hardware acceleration logic and application specific I/O devices.

Utilizing a stack oriented, non-pipelined, multiple bus architecture with one or two cycle instruction times, the RTX 2000 allows the efficient implementation of such real-time applications as Digital Signal Processing (DSP), Digital Control Processing, Image Processing, Robotics, Graphics, Simulation, and Animation. Because these applications can be supported entirely with high level languages such as Forth and C on the RTX 2000, the development cycle time to system implementation is drastically reduced.

The RTX 2000 Microcontroller is an exceptionally powerful device with the ability to meet numerous application specific needs. The advantages of the RTX may be further enhanced through the use of RTX specific peripherals and by the development system support which Harris provides for the RTX hardware.

The Harris Advanced Standard Cell and Compiler Library was used to design and fabricate the RTX 2000. As part of the Harris family of compatible cell libraries, the RTX 2000 can be incorporated into customer ASIC designs.

RTX 2000 Block Diagram

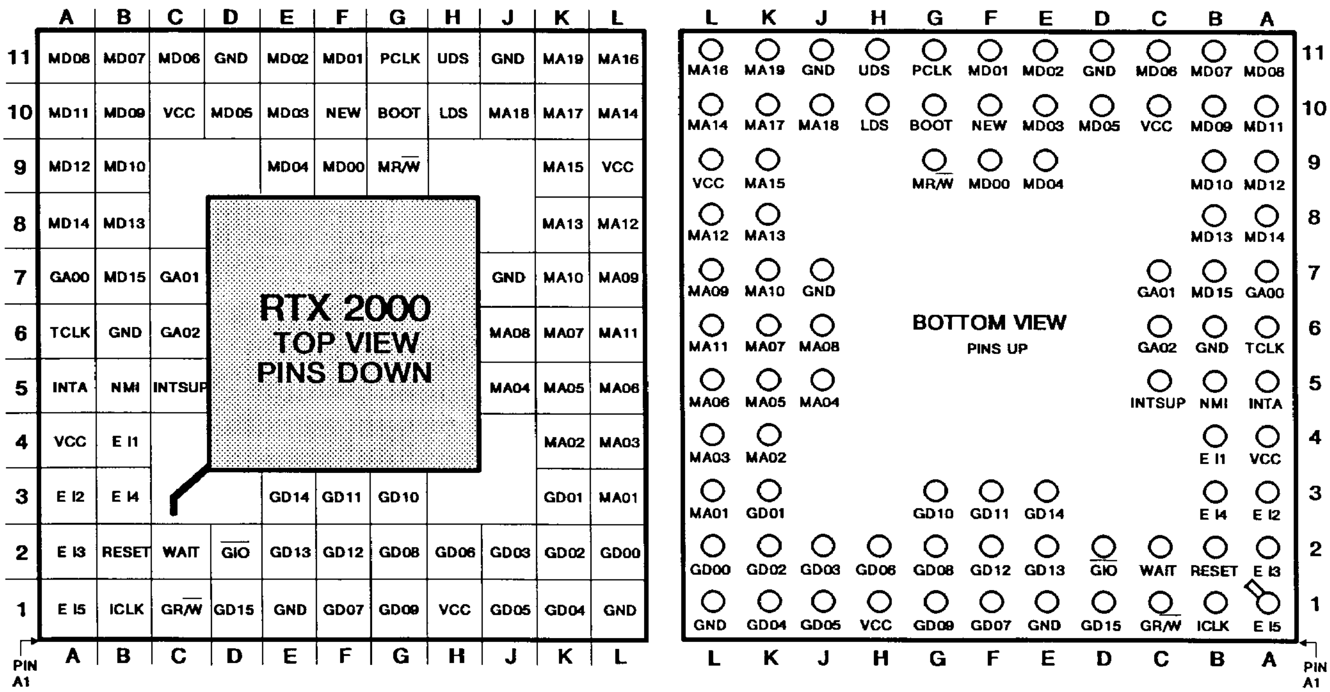


Real Time Express ™, RTX™, RTX 2000™ and ASIC Bus™ are Trademarks of Harris Corporation

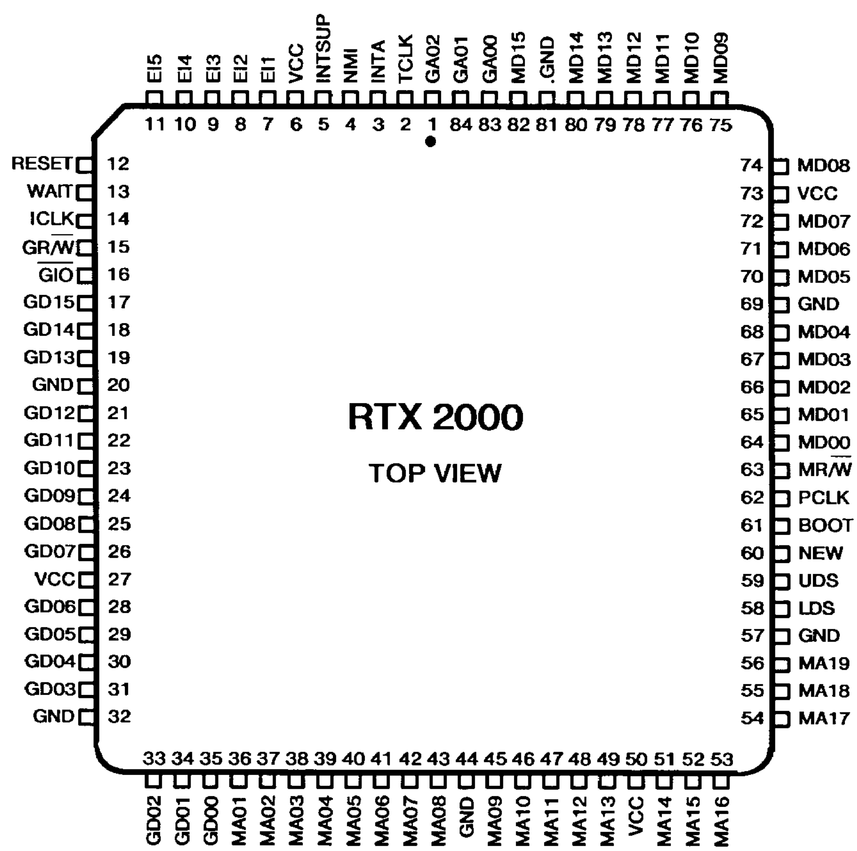
CAUTION: These devices are sensitive to electrostatic discharge. Proper I.C. handling procedures should be followed.

Pinouts

84 PIN PGA PACKAGE



84 LEAD PLCC PACKAGE



Note: An overbar on a signal name represents an active LOW signal.

TABLE 1. PGA AND PLCC PIN/SIGNAL ASSIGNMENTS

PLCC LEAD	PGA PIN	SIGNAL NAME	TYPE	PLCC LEAD	PGA PIN	SIGNAL NAME	TYPE
1	C6	GA02	Output; Address Bus	43	J6	MA08	Output; Address Bus
2	A6	TCLK	Output	44	J7	GND	Ground
3	A5	INTA	Output	45	L7	MA09	Output; Address Bus
4	B5	NMI	Input	46	K7	MA10	Output; Address Bus
5	C5	INTSUP	Input	47	L6	MA11	Output; Address Bus
6	A4	VCC	Power	48	L8	MA12	Output; Address Bus
7	B4	EI1	Input	49	K8	MA13	Output; Address Bus
8	A3	EI2	Input	50	L9	VCC	Power
9	A2	EI3	Input	51	L10	MA14	Output; Address Bus
10	B3	EI4	Input	52	K9	MA15	Output; Address Bus
11	A1	EI5	Input	53	L11	MA16	Output; Address Bus
12	B2	RESET	Input	54	K10	MA17	Output; Address Bus
13	C2	WAIT	Input	55	J10	MA18	Output; Address Bus
14	B1	ICLK	Input	56	K11	MA19	Output; Address Bus
15	C1	GR/W	Output	57	J11	GND	Ground
16	D2	GIO	Output	58	H10	LDS	Output
17	D1	GD15	I/O; Data Bus	59	H11	UDS	Output
18	E3	GD14	I/O; Data Bus	60	F10	NEW	Output
19	E2	GD13	I/O; Data Bus	61	G10	BOOT	Output
20	E1	GND	Ground	62	G11	PCLK	Output
21	F2	GD12	I/O; Data Bus	63	G9	MR/W	Output
22	F3	GD11	I/O; Data Bus	64	F9	MD00	I/O; Data Bus
23	G3	GD10	I/O; Data Bus	65	F11	MD01	I/O; Data Bus
24	G1	GD09	I/O; Data Bus	66	E11	MD02	I/O; Data Bus
25	G2	GD08	I/O; Data Bus	67	E10	MD03	I/O; Data Bus
26	F1	GD07	I/O; Data Bus	68	E9	MD04	I/O; Data Bus
27	H1	VCC	Power	69	D11	GND	Ground
28	H2	GD06	I/O; Data Bus	70	D10	MD05	I/O; Data Bus
29	J1	GD05	I/O; Data Bus	71	C11	MD06	I/O; Data Bus
30	K1	GD04	I/O; Data Bus	72	B11	MD07	I/O; Data Bus
31	J2	GD03	I/O; Data Bus	73	C10	VCC	Power
32	L1	GND	Ground	74	A11	MD08	I/O; Data Bus
33	K2	GD02	I/O; Data Bus	75	B10	MD09	I/O; Data Bus
34	K3	GD01	I/O; Data Bus	76	B9	MD10	I/O; Data Bus
35	L2	GD00	I/O; Data Bus	77	A10	MD11	I/O; Data Bus
36	L3	MA01	Output; Address Bus	78	A9	MD12	I/O; Data Bus
37	K4	MA02	Output; Address Bus	79	B8	MD13	I/O; Data Bus
38	L4	MA03	Output; Address Bus	80	A8	MD14	I/O; Data Bus
39	J5	MA04	Output; Address Bus	81	B6	GND	Ground
40	K5	MA05	Output; Address Bus	82	B7	MD15	I/O; Data Bus
41	L5	MA06	Output; Address Bus	83	A7	GA00	Output; Address Bus
42	K6	MA07	Output; Address Bus	84	C7	GA01	Output; Address Bus

TABLE 2. I/O SIGNAL DESCRIPTION

SIGNAL	PLCC LEAD	DESCRIPTION
INPUTS		
WAIT	13	WAIT: A HIGH on this pin causes PCLK to be held LOW and the current cycle to be extended.
ICLK	14	INPUT CLOCK: Internally divided by 2 to generate all on-chip timing (CMOS input levels).
RESET	12	A HIGH level on this pin resets the RTX. Must be held high for at least 2 ICLK cycles (CMOS input levels).
EI2, EI1	8, 7	EXTERNAL INTERRUPTS 2, 1: Active HIGH level-sensitive inputs to the Interrupt Controller. Sampled on the rising edge of PCLK. See Timing Diagrams for detail.
EI5-EI3	11-9	EXTERNAL INTERRUPTS 5, 4, 3: Dual purpose inputs; active HIGH level-sensitive Interrupt Controller inputs; active HIGH edge-sensitive Timer/Conter inputs. As interrupt inputs, they are sampled on the rising edge of PCLK. See Timing Diagrams for detail.
NMI	4	NON-MASKABLE INTERRUPT: Active HIGH edge-sensitive Interrupt Controller input capable of interrupting any processor cycle. See the Interrupt Suppression Section. (CMOS input levels)
INTSUP	5	INTERRUPT SUPPRESS: A HIGH on this pin inhibits all maskable interrupts, internal and external.

TABLE 2. I/O SIGNAL DESCRIPTION (Continued)

SIGNAL	PLCC LEAD	RESET LEVEL	DESCRIPTION
OUTPUTS			
NEW	60	1	NEW: A HIGH on this pin indicates that an Instruction Fetch is in progress.
BOOT	61	1	BOOT: A HIGH on this pin indicates that Boot Memory is being accessed. This pin can be set or reset by accessing bit 3 of the Configuration Register (CR).
MR/$\overline{W}$	63	1	MEMORY READ/WRITE: A LOW on this pin indicates that a Memory Write operation is in progress.
UDS	59	1	UPPER DATA SELECT: A HIGH on this pin indicates that the high byte of memory (MD15-MD08) is being accessed.
LDS	58	1	LOWER DATA SELECT: A HIGH on this pin indicates that the low byte of memory (MD07-MD00) is being accessed.
$\overline{GIO}$	16	1	ASIC I/O: A LOW on this pin indicates that an ASIC Bus operation is in progress.
GR/$\overline{W}$	15	1	ASIC READ/WRITE: A LOW on this pin indicates that an ASIC Bus Write operation is in progress.
PCLK	62	0	PROCESSOR CLOCK: Runs at half the frequency of ICLK. All processor cycles begin on the rising edge of ICLK. Held low extra cycles when WAIT is asserted.
TCLK	2	0	TIMING CLOCK: Same frequency and phase as PCLK but continues running during Wait cycles.
INTA	3	0	INTERRUPT ACKNOWLEDGE: A HIGH on this pin indicates that an Interrupt Acknowledge cycle is in progress.
ADDRESS BUSES (OUTPUTS)			
GA02	1		ASIC ADDRESS: 3-bit ASIC Address Bus.
GA01	84		
GA00	83		
MA19-MA14	56-51		MEMORY ADDRESS: 19-bit Memory Address Bus.
MA13-MA09	49-45		
MA08-MA01	43-36		
DATA BUSES (I/O)			
GD15-GD13	17-19		ASIC DATA: 16-bit bidirectional external ASIC Data Bus.
GD12-GD07	21-26		
GD06-GD03	28-31		
GD02-GD00	33-35		
MD15	82		MEMORY DATA: 16-bit bidirectional Memory Data Bus.
MD14-MD08	80-74		
MD07-MD05	72-70		
MD04-MD00	68-64		
POWER CONNECTIONS			
VCC	6, 27, 50, 73		Power supply +5 volt connections. A 0.1 µF, low impedance decoupling capacitor should be placed between VCC and GND. This should be located as close to the RTX package as possible.
GND	20, 32, 44, 57, 69, 81		Power supply ground return connections.

RTX 2000 Processor

The RTX 2000 processor is based on a two-stack architecture. The two stacks, which are Last-in-first-out (LIFO) structures, are called the Parameter Stack and the Return Stack.

Two internal registers provide the top two elements of the 16-bit wide Parameter Stack, while the remaining elements are contained in on-chip memory ("stack memory").

The Return Stack is 21-bits wide with the top element stored internally in registers while the remaining elements are contained in stack memory.

The RTX architecture is optimized for minimal Subroutine Call/Return overhead. A Subroutine Call takes one cycle, while a Subroutine Return usually takes zero cycles.

The RTX 2000 core has eight 16-bit internal registers, an ALU, internal data buses, and control hardware to perform instruction decoding and sequencing.

The relationship between the RTX core and the on-chip peripherals is shown in Figure 1, along with the off-chip user interfaces. Due to the highly parallel architecture of the RTX processor, peak execution rates during simultaneous bus operations can reach the equivalent of 40 million Forth language operations per second at a clock rate of 10MHz. Typical execution rates exceed 10 million operations per second.

RTX 2000 Operation

Control of all data paths, including the **Program Counter Register** (PC), is provided by the Instruction Decoder. This hardware determines what function is to be performed by looking at the contents of the **Instruction Register**, IR , and subsequently determines the sequence of operations through data path control.

Instructions which do not perform Memory accesses execute in a single clock cycle while the next instruction is being fetched. As shown in Figure 2, the instruction is latched into IR at the beginning of a clock cycle. The instruction is then decoded by the processor. All necessary internal operations are performed simultaneously with fetching the next instruction.

Instructions which access Memory require two clock cycles to be executed. During the first cycle, the instruction is decoded, the address of the memory location to be accessed is placed on the address bus, (MA19-MA01), and

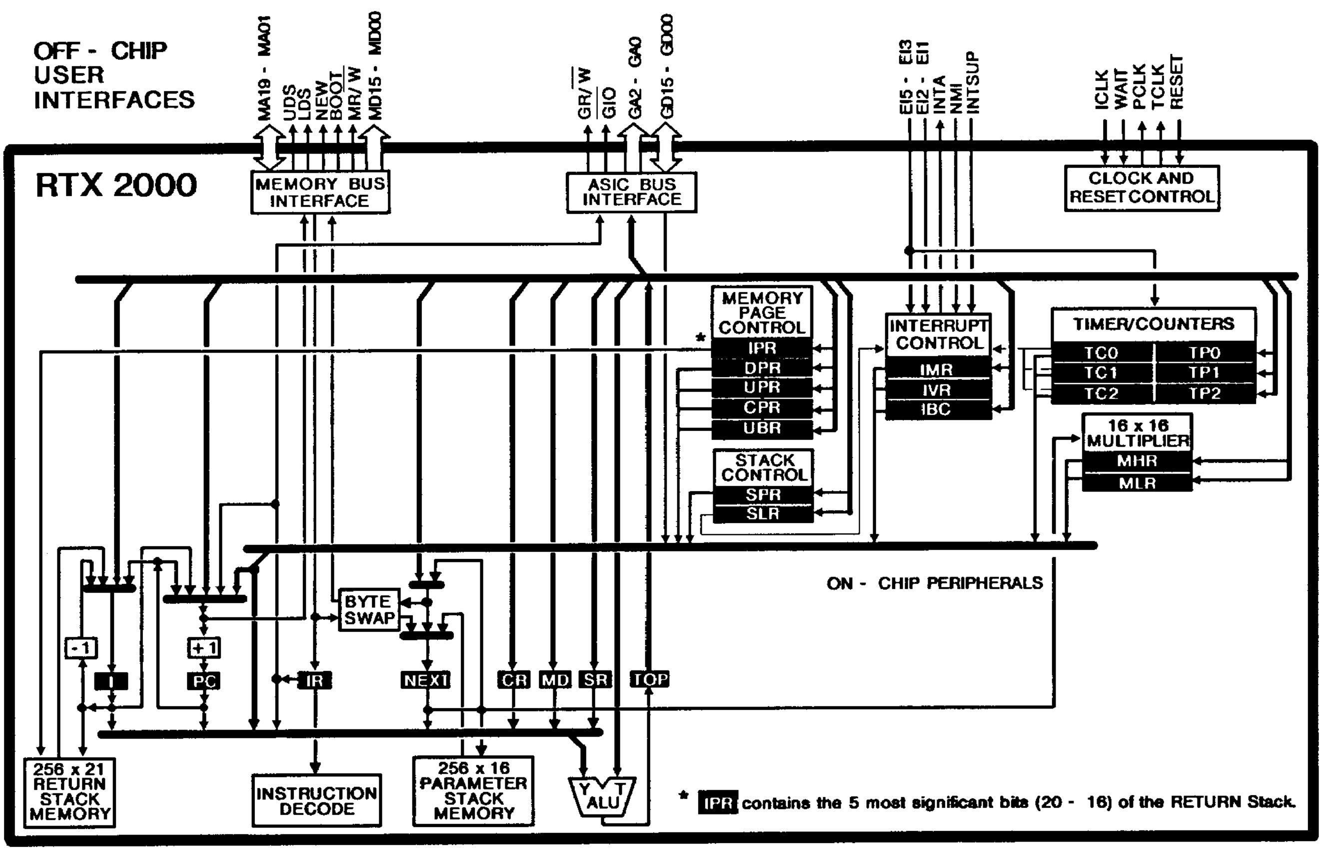


FIGURE 1. RTX 2000 FUNCTIONAL BLOCK DIAGRAM

the memory data, (MD15-MD00), is read or written. During the second cycle, ALU operations are performed, the address of the next instruction to be executed is placed on the Memory Address Bus, and the next instruction is fetched (see Figure 2).

RTX Data Buses and Address Buses

Unidirectional data paths and simultaneous operation of some data buses allow for maximum efficiency of internal data flow in the RTX.

External data flow is provided by the ASIC Data Bus (GD15-GD00) and the Memory Data Bus (MD15-MD00), both of which are bidirectional. Address information for accessing external memory or external ASIC devices is output via either the Memory Address Bus (MA19-MA01) or the ASIC Address Bus (GA02-GA00) (See Table 3).

RTX Internal Registers

The eight 16-bit internal registers are:

TOP : The **Top Register** contains the top element of the Parameter Stack. The contents of TOP may be directed to any I/O device or to any processor register except the **Instruction Register.** TOP is also the T input to the ALU. Input to TOP must come through the ALU. This register holds the most significant 16 bits of 32-bit products and 32-bit dividends.

NEXT : The **Next Register** holds the second element of the Parameter Stack. During a stack "push", the contents of NEXT are transferred to stack memory, and the contents of TOP are put into NEXT . This register is used to hold the least significant 16 bits of 32-bit products.

IR : The **Instruction Register** is a latch which contains the instruction currently being executed. Input to this register comes from Main Memory (see Tables 11-20).

TABLE 3. EXTERNAL RTX 2000 DATA BUSES AND ADDRESS BUSES

BUS NAME	FLOW DIRECTION	DESCRIPTION
Memory Data Bus	Bidirectional	Carries data to and from Main Memory (MD15-MD00).
ASIC Data Bus	Bidirectional	Carries data to and from off-chip I/O devices (GD15-GD00).
Memory Address	Output Only	Carries address information for Main Memory (MA19-MA01).
ASIC Address Bus	Output Only	Carries address information for external ASIC Bus devices (GA02-GA00).

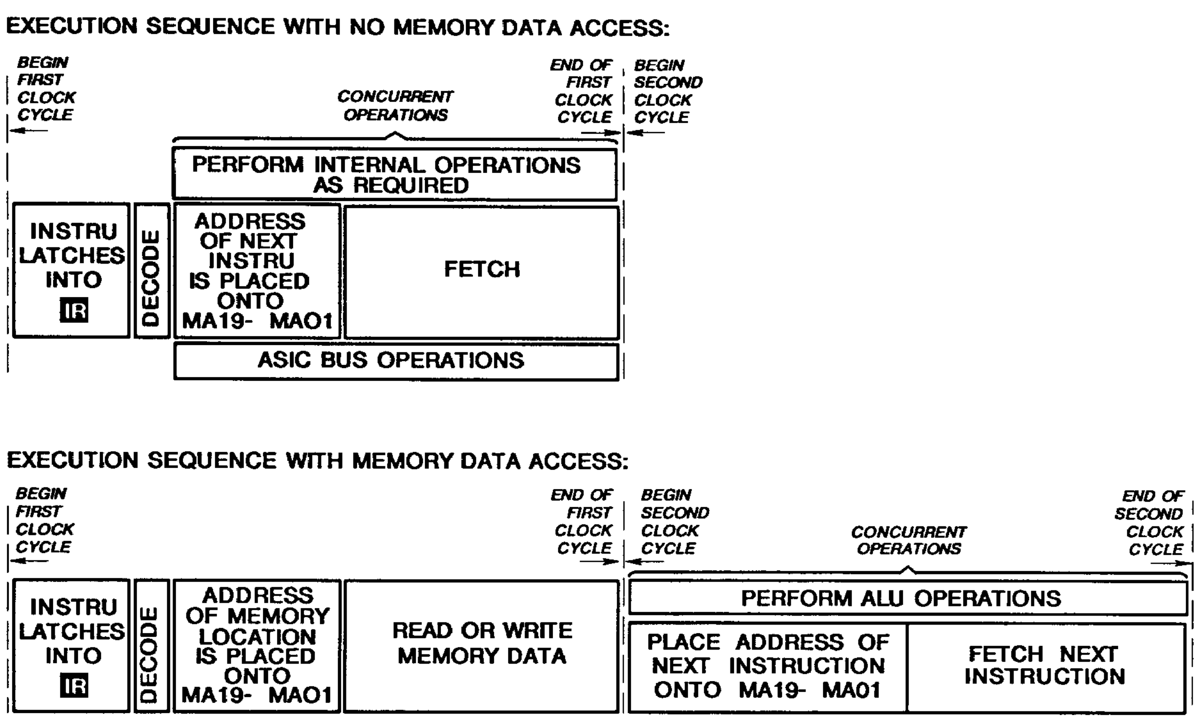


FIGURE 2. INSTRUCTION EXECUTION SEQUENCE

CR : The **Configuration Register** contains bits which indicate the current status/setup of the RTX processor. See Figure 3.

PC : The **Program Counter Register** contains the address of the next instruction to be fetched from Main Memory.

I : The **Index Register** contains 16 bits of the 21-bit top element of the Return Stack, and is also used to hold the count for streamed and loop instructions (see Figure 12). In addition, I can be used to hold data and can be written from TOP. The contents of I may be accessed in either the push/pop mode in which values are moved to/from stack memory as required, or in the read/write mode in which the stack memory is not affected. When the Streamed Instruction Mode is used, a value (count) is written to I and the next instruction is executed that number of times plus one (i.e. n+1).

MD : The **Multistep Divide Register** holds the divisor during multistep math operations. MD may also be used as a general purpose scratch pad register.

SR : The **Square Root Register** holds the intermediate values used during calculation of square roots. SR may also be used as a general purpose scratch pad register.

On-Chip Peripheral Registers

TIMER/COUNTER REGISTERS:

TC0 , TC1 , TC2 (Read Only): The **Timer/Counter Registers** are 16-bit registers which contain the current count value for each of the three Timer/Counters. The counter is decremented at each rising clock edge of TCLK. Reading from these registers at any time does not disturb their contents. The sequence of Timer/Counter operations is shown in Figure 14 in the Timer/Counters section.

TP0 , TP1 , TP2 (Write Only): The **Timer Pre-load Registers** contain the initial 16-bit count values which are written to each timer. After a timer counts down to zero, the pre-load register for that timer writes its pre-load count value to it at the next rising clock edge, synchronously with TCLK.

MULTIPLIER REGISTERS:

MHR (Read Only): The **Multiplier High Product Register** holds the most significant 16 bits of the 32-bit product generated by the RTX Multiplier. If the IBC register's **ROUND** bit is set, this register contains the rounded 16-bit output of the multiplier.

MLR (Read Only): The **Multiplier Lower Product Register** holds the least significant 16 bits of the 32-bit product generated by the RTX Multiplier.

INTERRUPT CONTROLLER REGISTERS:

IMR : By setting the appropriate bit, the **Interrupt Mask Register** allows each interrupt, except the Non-Maskable Interrupt, to be masked. See Figure 4 for bit assignments for this register.

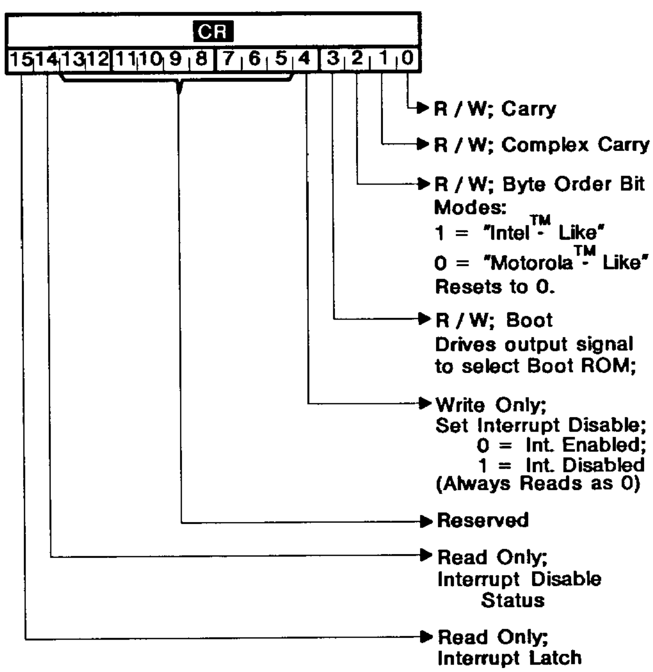


FIGURE 3. CONFIGURATION RESISTER (CR) BIT ASSIGNMENTS

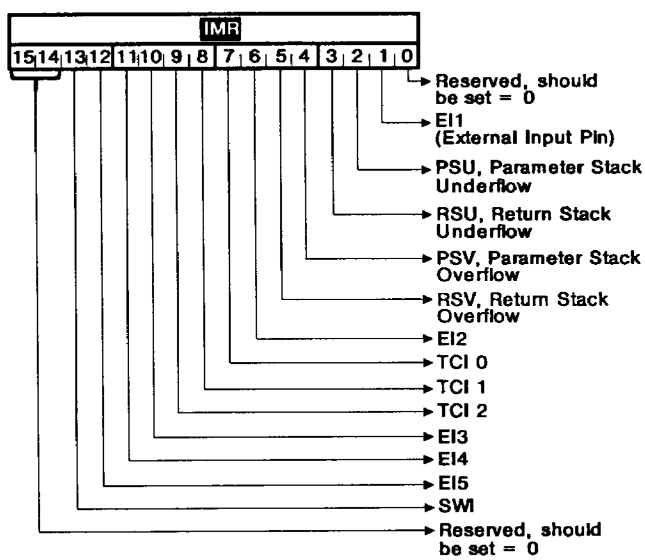


FIGURE 4. INTERRUPT MASK REGISTER (IMR) BIT ASSIGNMENTS

Motorola™ is a Registered Trademark of Motorola Inc.
Intel™ is a Registered Trademark of Intel Corporation

IBC : The **Interrupt Base/Control Register** is used to store the Interrupt Vector base address and to specify software options, as indicated by the bit assignments in Figure 5.

IVR (Read Only): The **Interrupt Vector Register** holds the current Interrupt Vector value. See Figure 6 and Table 5.

STACK CONTROLLER REGISTERS:

SLR (Write Only): The **Stack Limit Register** holds the upper limit values (0 to 255) for the depth of the Parameter Stack (bits 0-7) and the Return Stack (bits 8-15). These must be accessed together. See Figure 7.

SPR : The **Stack Pointer Register** holds the stack pointer value for each stack. Bits 8-15 represent the stack memory location being accessed for the Return Stack, while bits 0-7 represent the stack memory location being accessed for the Parameter Stack. These must be accessed together, as SPR. See Figure 8.

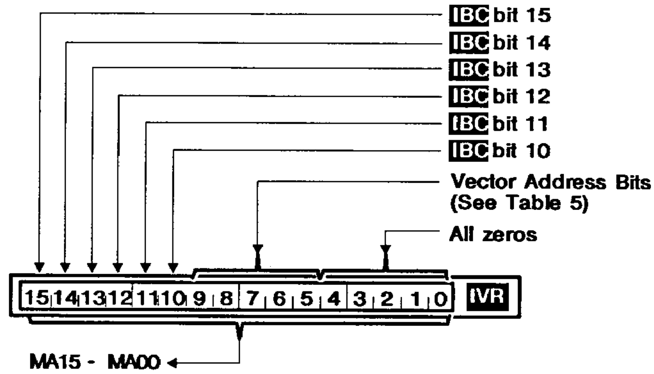
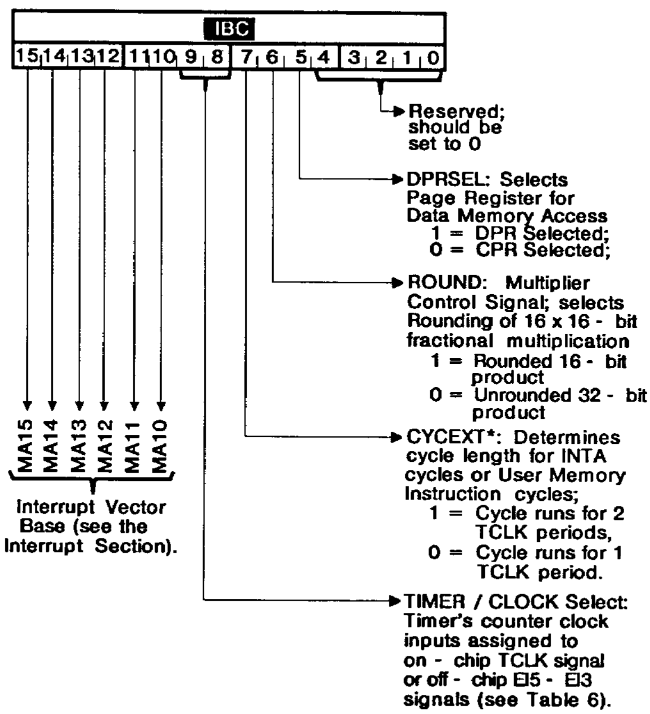


FIGURE 6. INTERRUPT VECTOR REGISTER (IVR) BIT FIELD ASSIGNMENTS



* NOTE: For f(TCLK) > 8MHz, set CYCEXT = 1.

FIGURE 5. INTERRUPT BASE/CONTROL REGISTER (IBC) BIT ASSIGNMENTS

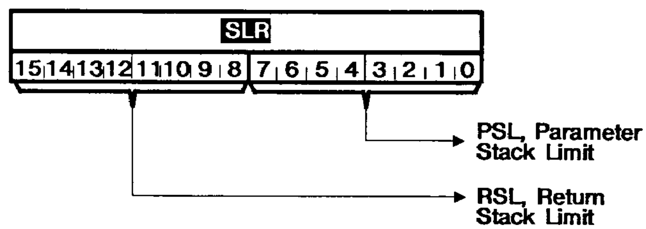


FIGURE 7. STACK LIMIT REGISTER (SLR) BIT ASSIGNMENTS

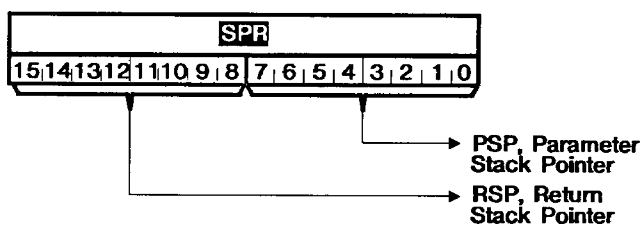


FIGURE 8. STACK POINTER REGISTER (SPR) BIT ASSIGNMENTS

MEMORY PAGE CONTROLLER REGISTERS:

CPR : The **Code Page Register** contains the value for the current Code page. See Figure 9 for bit field assignments.

DPR : The **Data Page Register** contains the value for the current Data page. See Figure 10 for bit field assignments.

UPR : The **User Page Register** contains the value for the current 32K-word User page. See Figure 11 for bit field assignments.

UBR : The **User Base Address Register** contains the base address for User Memory Instructions. See Figure 11 for bit field assignments.

IPR : The **Index Page Register** extends the **Index Register** (**I**) by 5 bits, i.e. when a Return From Subroutine is performed, the **IPR** contains the Code page from which the subroutine was called, and comprises the 5 most significant bits of the top element of the Return Stack. See Figure 12.

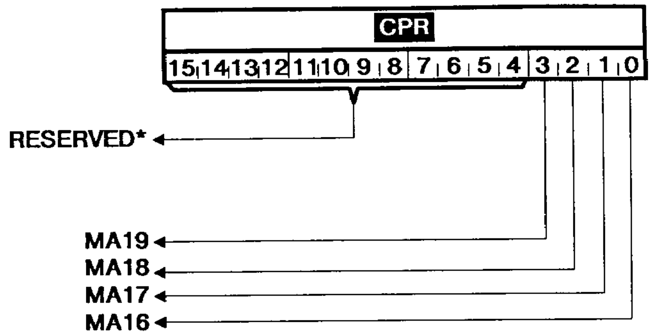


FIGURE 9. CODE PAGE REGISTER (CPR) BIT FIELD ASSIGNMENTS

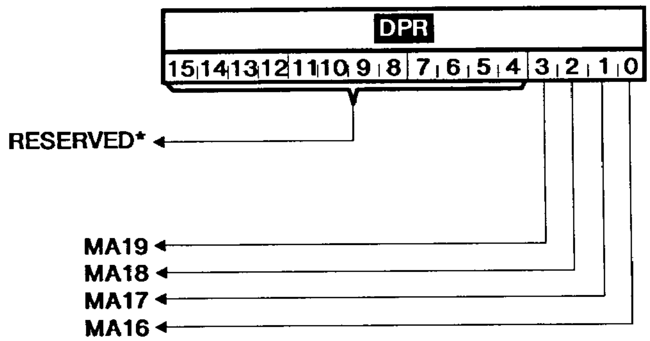


FIGURE 10. DATA PAGE REGISTER (DPR) BIT FIELD ASSIGNMENTS

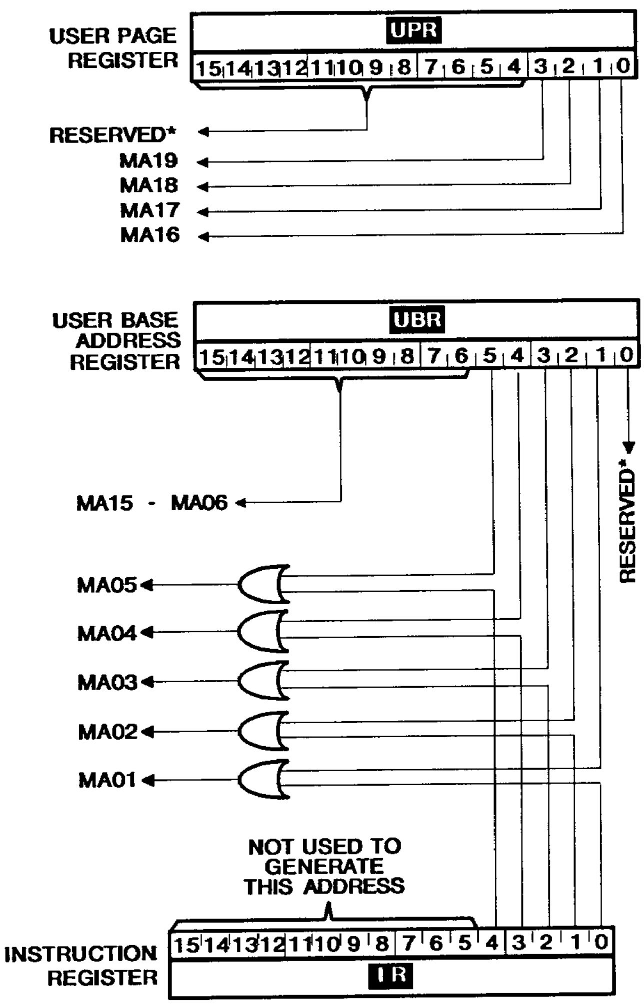


FIGURE 11. MEMORY ADDRESS (UBR and UPR) BIT FIELD ASSIGNMENTS

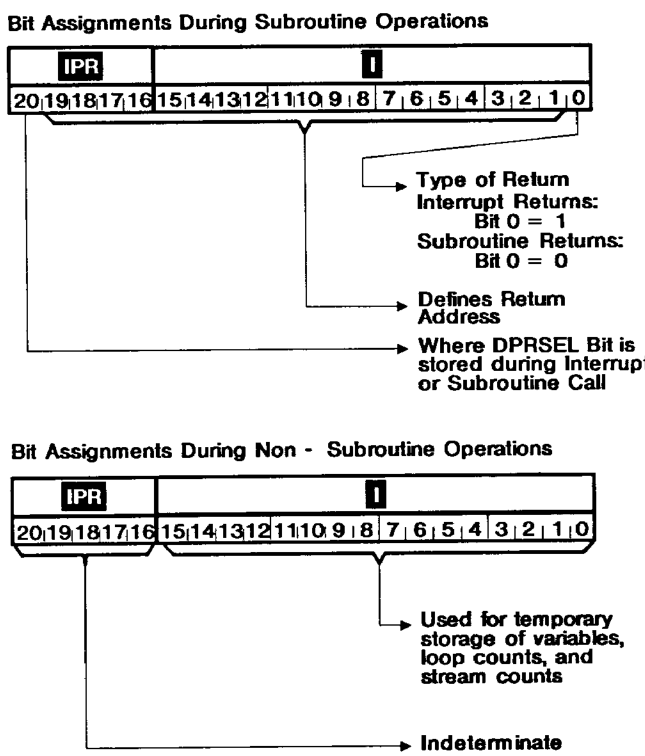


FIGURE 12. RETURN STACK (I and IPR) BIT FIELD ASSIGNMENTS

* NOTE: Always read as "0". Should be set = 0 during Write operations.

Initialization of Registers

A HIGH level on the RTX RESET pin initializes the on-chip circuits. The registers initialize as shown in Table 4. While the RESET input is HIGH, the TCLK and PCLK clock outputs are held reset in the LOW state.

RTX Interrupt Controller

The RTX 2000 Interrupt Controller manages interrupts for the RTX 2000 Microcontroller core. Its sources include several on-chip peripherals and 6 external interrupt inputs.

TABLE 4. REGISTER INITIALIZATION AND ASIC ADDRESS ASSIGNMENTS

REGISTER	HEX	INTIALIZED CONTENTS	COMMENTS
TOP		0000 0000 0000 0000	TOP Register: No ASIC address is assigned for this register; TOP is the implicit source or destination for certain instructions.
NEXT		1111 1111 1111 1111	NEXT Register: No ASIC address is assigned for this register; NEXT is the implicit source or destination for certain instructions and for external memory data.
IR		0000 0000 0000 0000	Instruction Register: No ASIC address is assigned for this register; IR is the implicit source for certain instructions and cannot be read or written directly.
I	00H 01H 02H	1111 1111 1111 1111	Index Register: The address used for I determines what type of operation will be performed (see Table 10).
CR	03H	0100 0000 0000 1000	Configuration Register: Boot = 1; Interrupts Disabled; Byte Order = 0.
MD	04H	1111 1111 1111 1111	Multistep Divide Register.
SR	06H	0000 0000 0000 0000	Square Root Register.
PC	07H	0000 0000 0000 0000	Program Counter Register.
IMR	08H	0000 0000 0000 0000	Interrupt Mask Register.
SPR	09H	0000 0000 0000 0000	Stack Pointer Register: The beginning address for each stack is set to a value of "0".
IVR	0BH	0000 0010 0000 0000	Interrupt Vector Register: Read only; this register holds the current Interrupt Vector value, and is initialized to the "No Interrupt" value.
SLR	0BH	1111 1111 1111 1111	Stack Limit Register: Each stack limit is set to a value of 255.
IPR	OCH	0000 0000 0000 0000	Index Page Register.
DPR	ODH	0000 0000 0000 0000	Data Page Register: The Data Address Page is set for page '0'.
UPR	0EH	0000 0000 0000 0000	User Page Register: The User Address Page is set for page '0'.
CPR	0FH	0000 0000 0000 0000	Code Page Register: The Code Address Page is set for page '0'.
IBC	10H	0000 0000 0000 0000	Interrupt Base/Control Register.
UBR	11H	0000 0000 0000 0000	User Base Address Register: The User base address is set to '0' within the user page.
TC0/TP0	13H	0000 0000 0000 0000	Timer/Counter Register 0: Set to time out after 65536 clock periods or events.
TC1/TP1	14H	0000 0000 0000 0000	Timer/Counter Register 1: Set to time out after 65536 clock periods or events.
TC2/TP2	15H	0000 0000 0000 0000	Timer/Counter Register 2: Set to time out after 65536 clock periods or events.
MLR	16H	1111 1111 0000 0000	Multiplier Lower Product Register: Read Only.
MHR	17H	1111 1111 1111 1111	Multiplier High Product Register: Read Only.

When one of the sources requests an interrupt, the Interrupt Controller checks whether the interrupt is masked. If it is not, the controller attempts to interrupt the processor. If processor interrupts are enabled, the processor will execute an Interrupt Acknowledge cycle and disable interrupts. In response to the Interrupt Acknowledge cycle, the Interrupt Controller places an Interrupt Vector on the internal ASIC Bus, based on the highest priority pending interrupt. The processor performs a Subroutine Call to the address in Memory page 0 contained in the vector. When the Interrupt Handler executes a Return From Subroutine, the processor returns to the interrupted code and re-enables interrupts. Before the Interrupt Handler returns, it must ensure that the condition that caused the interrupt is cleared, or else the processor will be immediately interrupted as soon as it returns.

Processor interrupts are enabled and disabled by clearing and setting the **Interrupt Disable Flag**. When the RTX is reset, this flag is set (bit 04 of the CR = 1), disabling the interrupts. This bit is a Write-Only bit that always reads as 0, allowing interrupts to be enabled in only 2 cycles with a simple read/write operation in which the processor reads the bit value, then writes it back to the same location. The actual status of the Interrupt Disable Flag can be read from bit 14 of the CR.

In addition to disabling them at the processor level, all interrupts except the **Non-Maskable Interrupt (NMI)** can be individually masked by the Interrupt Controller by setting the appropriate bit in the **Interrupt Mask Register** (IMR). After resetting the RTX 2000, all of the bits in the IMR are cleared, thereby unmasking all interrupts.

The Interrupt Controller prioritizes interrupt requests, and generates an Interrupt Vector for the highest priority interrupt request. The address that the vector points to is determined by the source of the interrupt and the contents of the **Interrupt Base/Control Register** (IBC). Because address bits MA19 - MA16 are always zero in an Interrupt Acknowledge cycle, the entry point to the Interrupt Handlers must reside on Memory Page zero. The rest of the vector is generated as indicated in Table 5.

The Interrupt Vector can also be read from the **Interrupt Vector Register** (IVR) directly. This allows interrupt requests to be monitored by software, even if they are disabled by the processor. If no interrupts are being requested, bit 09 of the IVR will be 1.

External interrupts EI5-EI1 are active HIGH level-sensitive inputs. Therefore, the Interrupt Handlers for these interrupts must clear the source of interrupt prior to returning to the interrupted code. The external **NMI**, however, is an edge-sensitive input which requires a rising edge to request an interrupt.

The two classes of on-chip peripherals that produce interrupts are the Stack Controllers and the Timer/Counters.

The Stack Controllers request an interrupt whenever a stack overflow or underflow condition exists. These interrupts can be cleared by pushing or popping the stack to alleviate the condition, or by rewriting SPR. See the section on "Dual Stack Architecture" for more information regarding how the limits set into SLR can be used to generate interrupts.

TABLE 5. INTERRUPT SOURCES, PRIORITIES AND VECTORS

PRIORITY	INTERRUPT SOURCE		SENSITIVITY	IMR BIT	VECTOR ADDRESS BITS 09	08	07	06	05
0 (High)	NMI	Non-Maskable Interrupt	Pos Edge	N/A	0	1	1	1	1
1	EI1	External Interrupt 1	High Level	01	0	1	1	1	0
2	PSU	Parameter Stack Underflow	High Level	02	0	1	1	0	1
3	RSU	Return Stack Underflow	High Level	03	0	1	1	0	0
4	PSV	Parameter Stack Overflow	High Level	04	0	1	0	1	1
5	RSV	Return Stack Overflow	High Level	05	0	1	0	1	0
6	EI2	External Interrupt 2	High Level	06	0	1	0	0	1
7	TCI0	Timer/Counter 0	Edge	07	0	1	0	0	0
8	TCI1	Timer/Counter 1	Edge	08	0	0	1	1	1
9	TCI2	Timer/Counter 2	Edge	09	0	0	1	1	0
10	EI3	External Interrupt 3	High Level	10	0	0	1	0	1
11	EI4	External Interrupt 4	High Level	11	0	0	1	0	0
12	EI5	External Interrupt 5	High Level	12	0	0	0	1	1
13 (Low)	SWI	Software Interrupt	High Level	13	0	0	0	1	0
N/A	None	No Interrupt	N/A	N/A	1	0	0	0	0

The timers generate edge-sensitive interrupts whenever they time out. Because they are edge-sensitive and are cleared during an Interrupt Acknowledge cycle or during the direct reading of IVR by software, no action is required by the handlers to clear the interrupt request.

Finally, a mechanism is provided by which interrupts can be requested by using software commands. A **SWI** is requested by setting an internal flip-flop attached to one input of the Interrupt Controller. The **SWI** is reset by clearing the flip-flop. The flip-flop is accessed by I/O Reads and Writes.

Because the interrupt may not be serviced immediately, the instructions which immediately follow the **SWI** should not depend on whether or not the interrupt has been serviced, and should cause a one or two cycle idle condition.

INTERRUPT SUPPRESSION

The RTX 2000 allows maskable interrupts to be suppressed, delaying them temporarily while critical operations are in progress. Critical operations are instruction sequences and hardware operations that, if interrupted, would result in the loss of data or misoperation of the hardware.

Standard critical operations during which interrupts are automatically suppressed by the system include Streamed instructions (see the description of the I register), Long Call sequences (see "Subroutine Calls And Returns"), and Multiplier Access instructions (see "RTX 2000 On-Chip Multiplier"). In addition to this, user defined, external devices can suppress interrupts during critical operations by applying a HIGH level on the **INTSUP** pin for as long as required.

Since the **NMI** can still cause the processor to perform an Interrupt Acknowledge cycle in the middle of these critical operations, preventing a normal return to the interrupted instruction, a "Return From Subroutine" should not be performed from the **NMI** service routine.

Interrupts which have occurred while interrupt suppression is in effect will be recognized as soon as the suppression terminates.

Dual Stack Architecture

The RTX 2000 features a dual stack architecture. The two stacks are the Parameter Stack and the Return Stack, both of which may be accessed in parallel by a single instruction. The functional structure of each of these stacks is shown in Figure 13. This architecture minimizes overhead in passing parameters between subroutines.

The Parameter Stack is used for temporary storage of data and for passing parameters between subroutines. The top two elements of this stack are contained in the TOP and NEXT registers of the processor, and the remainder of this stack is located in stack memory. The stack memory assigned to the Parameter Stack is 256 words deep by 16 bits wide.

The Return Stack is used for storing return addresses when performing Subroutine Calls, or for storing values temporarily. Because the RTX 2000 uses a separate Return Stack, it can

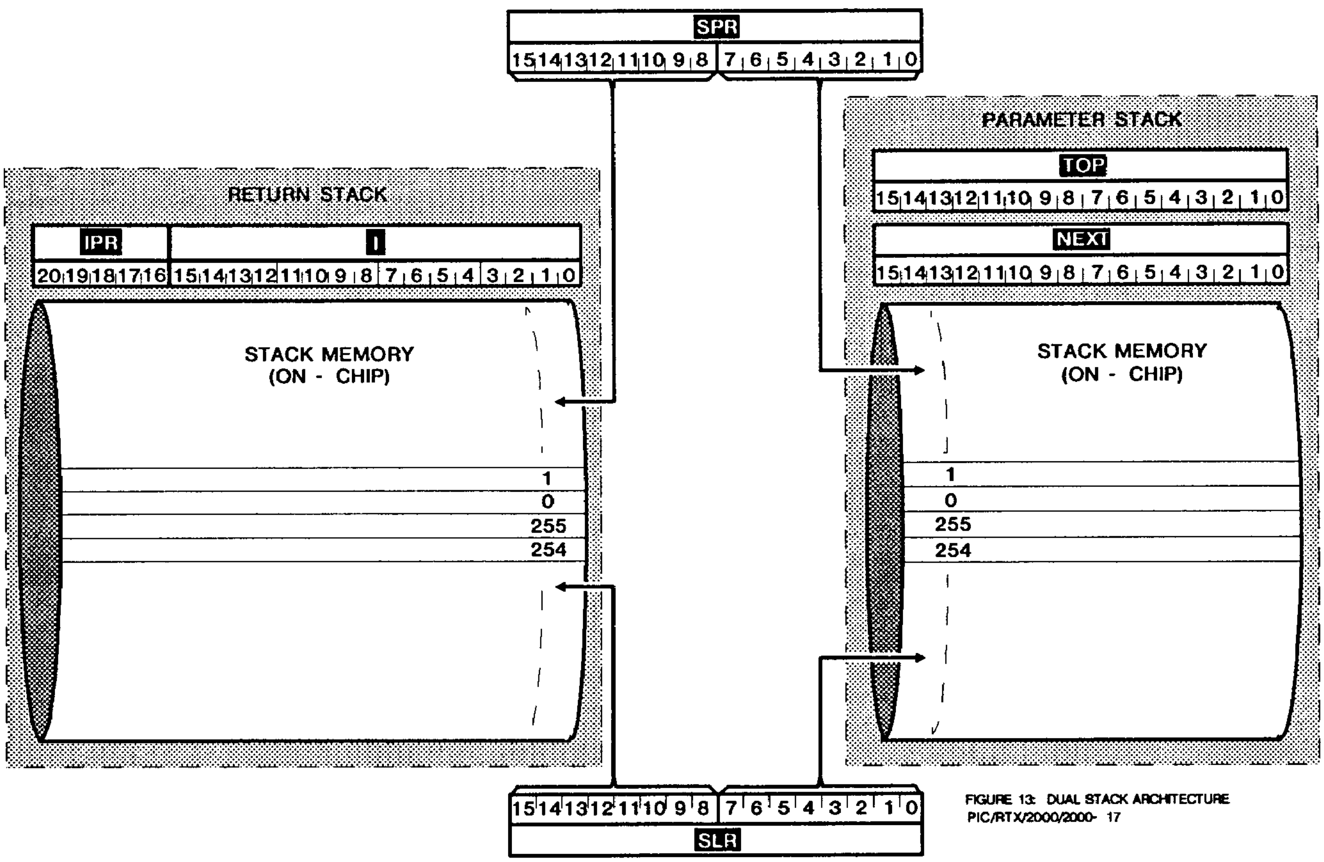


FIGURE 13. DUAL STACK ARCHITECTURE

call and return from subroutines and interrupts with a minimum of overhead. The top element of the Return Stack is 21 bits wide. The 16-bit **Index Register, I**, and the 5-bit **Index Page Register, IPR**, hold the top element of this stack, while the remainder is located in stack memory. The stack memory portion of the Return Stack is also 21 bits wide, and is 256 words deep.

The bit fields of the Return Stack take on different assignments, depending upon whether the processor is performing a subroutine operation, as indicated in Figure 12.

RTX 2000 STACK CONTROLLERS

The two stacks of the RTX 2000 are controlled by identical Programmable Stack Controllers.

The **Stack Pointer Register, SPR**, for the Programmable Stack Controllers contains the current top stack memory address for both areas of stack memory. Bits 8-15 of this register contain the current top stack memory address for the Return Stack, while bits 0-7 contain the current top stack memory address for the Parameter Stack. The value for each stack memory address can range from 0 to 255.

Each stack memory pointer points at the position of the "top" item in its stack memory. This is the item that was most recently pushed into this stack memory. On reset the stack memory pointer for both areas of stack memory is set to a value of 0.

During a stack READ ("pop") operation, the stack memory pointer points at the item which can be popped from the stack memory. Once that item has been popped from the stack memory, the stack memory pointer is decremented by 1.

During a stack WRITE ("push") operation, the stack memory pointer is incremented by 1 before the new item is pushed to the memory stack.

The **Stack Limit Register, SLR**, is a write only register which holds the upper limit values for the depth of the Parameter Stack (bits 0-7) and the Return Stack (bits 8-15). These upper limits are set by the user and can be any value between 0 and 255. At Reset, the Stack Limit Register is initialized to a value of 255 for each stack. The lower limit for each stack is fixed at 0. The upper and lower stack limits are utilized by the Interrupt Controller to generate "underflow", (PSU and RSU), and "overflow", (PSV and RSV), interrupts when the number of pops or pushes of the stack reach the set limits.

During a stack Read operation, an underflow interrupt is generated when data is read from stack location 1. An overflow interrupt is generated when the stack pointer value for the location being read is equal to one more than the stack limit value for that stack.

NOTE: If access to location 0 of either stack is required, the underflow interrupts should be masked or disabled.

During a stack Write operation, an overflow interrupt is generated when the stack pointer value for the location being written to is greater than the stack limit value for that stack. An underflow interrupt is generated when data is written to locations 0 or 255.

If a stack is pushed past location 255, the stack pointer for that stack will "wrap around" back to 0, allowing earlier data to be overwritten. Unless the ability to serve as a circular 256 word buffer is required by the application, the stack limits and interrupt responses should be configured to deal with the overflow condition before the wrap occurs and data is lost.

Since the RTX can take up to four clock cycles to respond to an interrupt, the values set in the Stack Limit register (PSL and RSL) should include a safety margin which allows valid values to continue being pushed onto the stack until the processor executes the interrupt service routine.

Because it is possible to generate an interrupt, then perform stack operations which cause it to go away before it has been serviced, the user should exercise care in stack management. It is also recommended that valid code be supplied at every interrupt vector location to prevent unforeseen errors.

RTX 2000 Timer/Counters

The RTX 2000 has three 16-bit timers, each of which can be configured to perform timing or event counting. All decrement synchronously with the falling edge of TCLK. Timer registers are readable in a single machine cycle because they reside on the processor's internal ASIC Bus.

The timer selection bits of the IBC determine whether a timer is to be configured for external event counting or internal time-base timing, and the respective counter clock inputs are assigned to the on-chip TCLK signal or the off-chip generated EI5-EI3 signals according to the value of these bits. EI5, EI4 and EI3 are sychronized internally with TCLK. See Table 6 for Timer/Clock selection by IBC bit values.

TABLE 6. TIMER/CLOCK SELECTION

IBC BIT VALUES		TIMER CLOCK SOURCE		
BIT 09	BIT 08	TC2	TC1	TC0
0	0	TCLK	TCLK	TCLK
0	1	TCLK	TCLK	EI3
1	0	TCLK	EI4	EI3
1	1	EI5	EI4	EI3

The timers (TC0, TC1, and TC2) are all free-running, and when they time-out, they reload themselves automatically with the programmed initial value from their designated Timer Preload Register (TP0 → TC0, TP1 → TC1, and TP2 → TC2), then continue timing or counting.

Each timer provides an output to the Interrupt Controller to indicate when a time-out for the timer has occurred. The RTX 2000 can determine the state of a timer at any time either by reading the timer's value, or upon a time-out by using the timer's interrupt. Figure 14 shows the sequence of Timer/Counter operations.

ALU

The RTX 2000 has a 16-bit ALU capable of performing the following arithmetic and logic operations:

- ADD and SUBTRACT (A-B and B-A; with and without carry)
- AND, OR, XOR, NOR, NAND, XNOR, NOT.

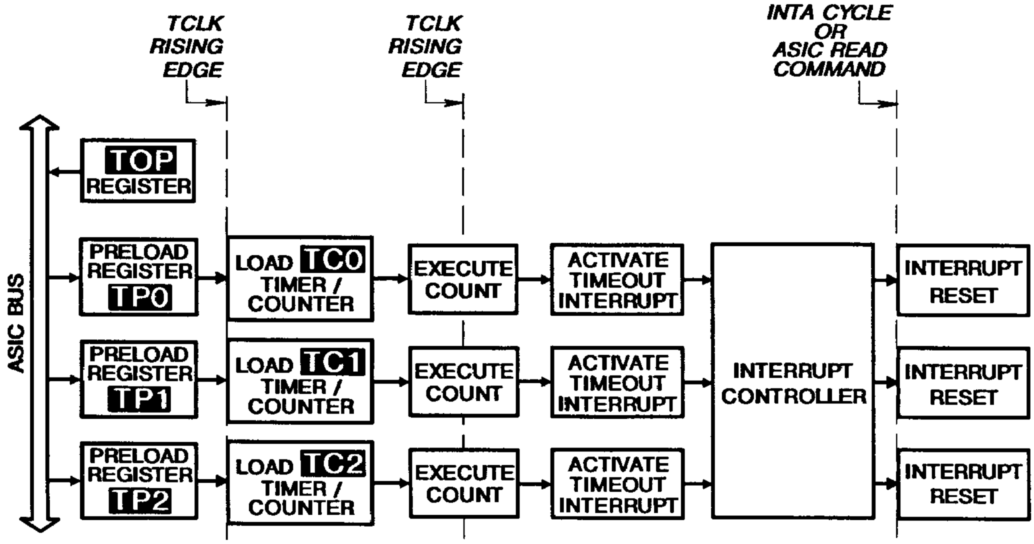
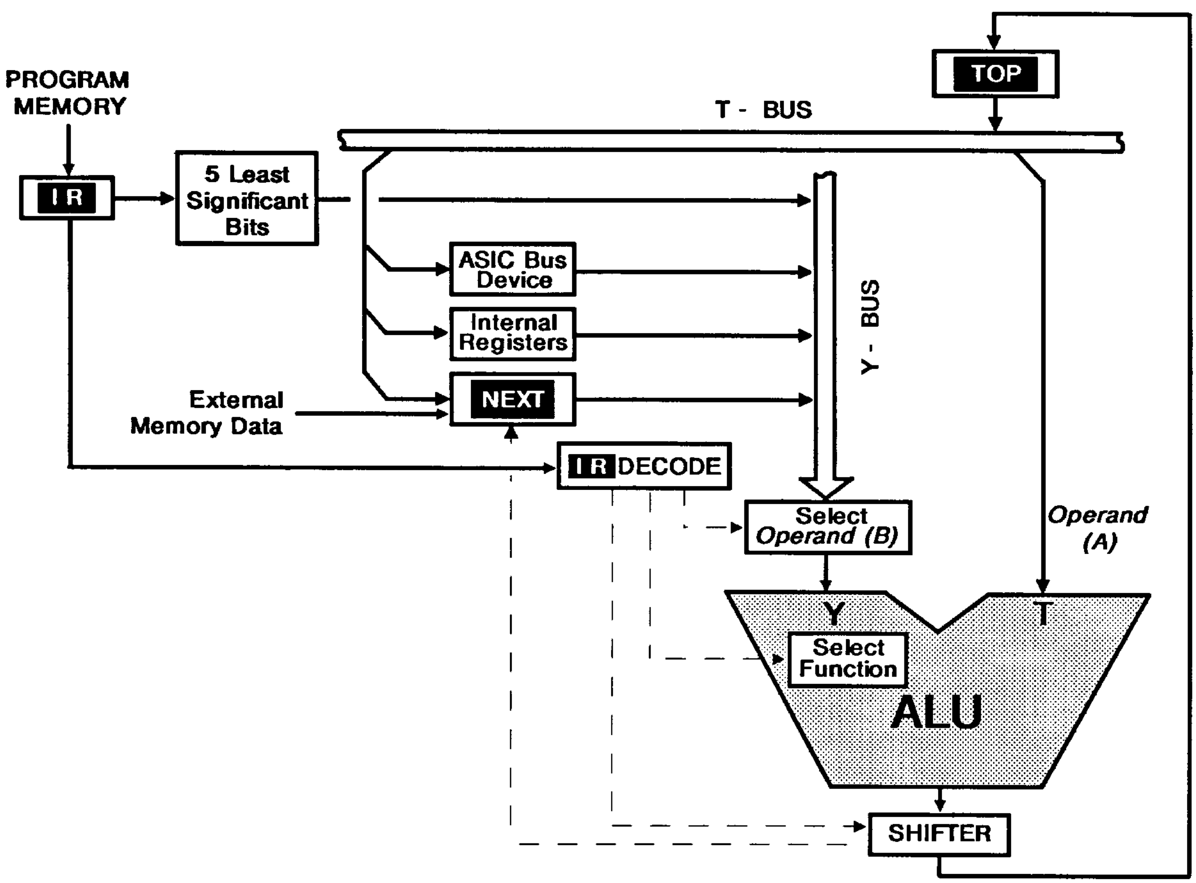


FIGURE 14. RTX 2000 TIMER/COUNTERS



NOTE: Data Paths are represented by solid lines; Control Paths are represented by dashed lines.

FIGURE 15. ALU OPERATIONS-CONTROL PATHS AND DATA FLOW

The TOP and NEXT registers can also undergo single bit shifts in the same cycle as a logic or arithmetic operation.

In Figure 15, the control and data paths to the ALU are shown. Except for TOP and NEXT, each of the Internal Core Registers can be addressed explicitly as an ASIC Bus Device, as can other internal registers in specialized applications such as in Step Math functions. In each of these cases, the input would be addressed as a device on the ASIC Bus.

When performing these functions, the math/logic operand (a) starts out in TOP and is placed on the T-bus. Operand (b) arrives at the ALU on the Y-bus, but can come from one of four sources: NEXT; an internal register; an ASIC Bus device; or from the 5 least significant bits of IR. The source of operand (b) is determined by the IR register bits used to define the ALU instruction coding. The result of the ALU operation is placed into TOP.

Step Math functions which are performed through the ALU are divide and square root. The ALU can also perform multiplication, but does not because this function is performed more efficiently by the RTX 2000 on-chip Multiplier. Sign and scaling functions are controlled by the ALU function and shift options, which are part of the coded instruction contained in IR.

Step Divide operation assumes a double precision (32-bit) dividend, with the most significant word placed in TOP, and the less significant word in NEXT, and the divisor in MD. In each step, if the contents in TOP are greater than the contents in MD (and therefore no borrow will be generated), then the contents of MD are subtracted from the contents of TOP. The result is placed into TOP. The contents of TOP and NEXT are then jointly shifted left one bit (32-bit left shift), and a "1" is shifted into the least significant bit of NEXT if the subtract was performed. This step is required for each quotient bit.

During Step Square Root operation, the double precision argument is assumed to be in TOP and NEXT, as in the Step Divide. The first step begins with MD containing zeros. The Step Square Root is performed much like a Step Divide, except that the input from the Y-bus is the OR of the contents of SR and MD*2. When the subtraction is performed, SR is ORed into MD, and SR is shifted right on every step. One step is required for each bit of the result. At the end of the operation, the square root of the original value is in MD and NEXT, and the remainder is in TOP.

RTX 2000 On-Chip Multiplier

The Hardware Multiplier on the RTX 2000 multiplies two 16-bit numbers, yielding a 32-bit product in one clock cycle.

The multiplication function is activated by an I/O Write instruction to one of the ASIC Bus addresses assigned to the multiplier.

The multiplier's input operands come from the TOP and NEXT registers, and can be treated as either signed (two's complement) or unsigned integers, depending on the form of the instruction used. In addition, if the **ROUND** option is selected the multiplier can round the result to 16 bits. Note that the multiply instructions do not pop the Parameter Stack; the contents of TOP and NEXT remain intact.

The product is read from the **Multiplier High Product Register**, MHR, which contains the upper 16 bits of the product, and the **Multiplier Lower Product Register,** MLR, which contains the lower 16 bits. The registers may be read in either order, and there is no requirement that both registers be read. Reading either register moves its value into TOP, and pushes the original value in TOP into NEXT. The original value in NEXT is lost; it is not pushed to stack memory. This permits overwriting the original operands left in TOP and NEXT, which are not popped by the multiply operation.

If 32-bit precision is not required, the multiplier output may be rounded to 16 bits. This is accomplished by setting the **ROUND** bit in the Interrupt Base/Control Register, IBC, to 1. The **Round** operation rounds the lower 16 bits of the results into the upper 16 bits. The result is read from MHR into TOP. Following the read, the contents of TOP and NEXT should be exchanged, then a "Drop Top of Stack" instruction should be executed to discard one of the original operands. The **ROUND** bit functions independently of whether the signed or unsigned mode is used.

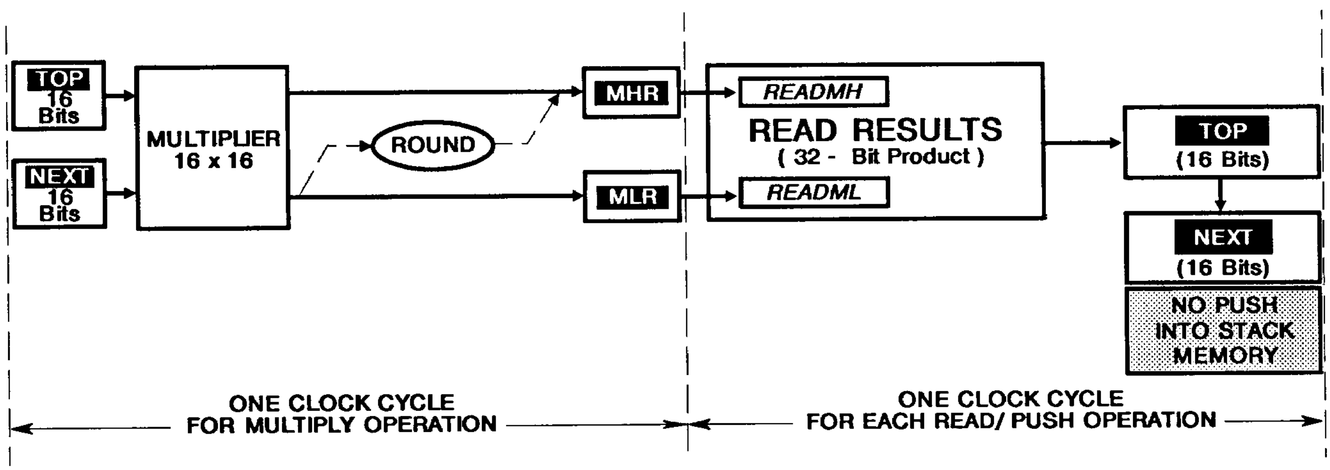


FIGURE 16. MULTIPLIER OPERATION

The multiply instructions disable interrupts during the multiplication cycle, and for the next cycle. Reading either MHR or MLR also disables interrupts during the read, and for the next clock cycle. This allows a multiplication operation to be performed, and both the upper and lower registers to be read sequentially, with no danger of a non-NMI interrupt service routine corrupting the contents of the registers between reads.

MULTIPLIER REGISTERS

MHR: The **Multiplier High Product Register** holds the most significant 16 bits of the 32-bit product generated by the RTX Multiplier. If the IBC register's **ROUND** bit is set, this register holds the rounded 16-bit output of the RTX Multiplier.

MLR: The **Multiplier Lower Product Register** holds the least significant 16 bits of the 32-bit product generated by the RTX Multiplier.

RTX 2000 Off-Chip Interfaces

ASIC BUS INTERFACE

The RTX 2000 ASIC Bus services both internal processor core registers and the on-chip peripheral registers, and eight external off-chip ASIC Bus locations. All ASIC Bus operations require a single cycle to execute and transfer a full 16-bit word of data. The external ASIC Bus maps into the last eight locations of the 32 location ASIC Address Space. The three least significant bits of the address are available as the ASIC Address Bus. The addresses therefore map as shown in Table 7.

TABLE 7. ASIC BUS MAP

ASIC BUS SIGNAL			ASIC ADDRESS
GA02	GA01	GA00	
0	0	0	18H
0	0	1	19H
0	1	0	1AH
0	1	1	1BH
1	0	0	1CH
1	0	1	1DH
1	1	0	1EH
1	1	1	1FH

RTX 2000 MEMORY BUS INTERFACE

The RTX 2000 can address 1 Megabyte of memory, divided into 16 non-overlapping pages of 64K bytes. The page accessed depends on whether the memory access is for **Code** (instructions and literals), **Data**, **User Memory**, or **Interrupt Code.** The page selected also depends on the contents of the Page Control Registers: the **Code Page Register** (CPR), the **Data Page Register** (DPR), the **User Page Register** (UPR) and the **Index Page Register** (IPR). Furthermore, the **User Base Address Register** (UBR) and the **Interrupt Base/Control Register** (IBC) are used to determine the complete address for User Memory accesses and Interrupt Acknowledge cycles. External memory data is accessed through NEXT.

When executing code other than during an Interrupt Service routine, the memory page is determined by the contents of the CPR. Bits 03-00 generate address bits MA19-MA16, as shown in Figure 9. The remainder of the address (MA15-MA01) comes from the **Program Counter Register** (PC). After resetting the processor, both the PC and the CPR are cleared and execution begins at page 0, word 0.

A new Code page is selected by writing a 4-bit value to the CPR. The value for the Code page is input to the CPR through a pre-load procedure which withholds the value for one clock cycle before loading the CPR to ensure that the next instruction is executed from the same Code page as the instruction which set the new Code page. Execution immediately thereafter will continue with the next instruction in the new page.

An Interrupt Acknowledge cycle is a special case of an Instruction Fetch cycle. When an Interrupt Acknowledge cycle occurs, the contents of the CPR and PC are saved on the Return Stack and then the CPR is cleared to point to page 0. The Interrupt Controller generates a 16-bit address, or "vector", which points to the code to be executed to process the interrupt. To determine how the Interrupt Vector is formed, refer to Figure 6 for the register bit assignments, and also to the Interrupt Controller section.

The page for data access is provided by either DPR or CPR, as shown in Figures 9 and 10. Data Memory Access instructions can be used to access data in a memory page other than that containing the program code. This is done by writing the desired page number into the **Data Page Register** (DPR) and setting bit 5 (**DPRSEL**) of the IBC register to 1. If DPR is set to equal CPR, or if **DPRSEL** = 0, data will be accessed in the Code page. When the RTX 2000 is reset, DPR points to page 0 and **DPRSEL** resets to 0, selecting the CPR.

The last memory area to be discussed is the User Memory area. User Memory consists of blocks of 32 words that can be located anywhere in memory. The word being accessed in a block is pointed to by the five least significant bits of the User Memory instruction (see Table 18), eliminating the need to explicitly load an address into TOP before reading or writing to the location. Upon RTX 2000 reset, UBR is cleared and points to the block starting at word 0, while UPR is cleared so that it points to page 0. The word in the block is pointed to by the five least significant bits of the User Memory instruction and bits 05–01 of the UBR. These bits from these two registers are logically OR'ed to produce the address of the word in memory. See Figure 11.

SUBROUTINE CALLS AND RETURNS

The RTX can perform both "short" subroutine calls and "long" subroutine calls. A short subroutine call is one for which the subroutine code is located within the same Code page as the Call instruction, and no processor cycle time is expended in reloading the CPR.

Performing a long subroutine call involves transferring execution to a different Code page. This requires that the CPR be loaded with the new Code page as described earlier, followed immediately by the Subroutine Call instruction. This adds two additional cycles to the execution time for the subroutine call.

For all instructions except Subroutine Calls or Branch instructions, bit 5 of the instruction code represents the Subroutine Return Bit. If this bit is set to 1, a Return is performed whereby the return address is popped from the Return Stack, as shown in Figure 11. The page for the return address comes from the IPR. The contents of the I register are written to the PC, and the contents of the IPR are written to the CPR so that execution resumes at the point following the Subroutine Call. The Return Stack is also popped at this time.

WORD AND BYTE MEMORY ACCESS

Using Main Memory Access instructions, the RTX 2000 can perform either word or single byte Main Memory accesses, as well as byte swapping within 16-bit words.

Bit 12 of the Memory Access Opcode (see Table 17), is used to determine whether byte or word operations are to be performed (where bit 12 = 0 signifies a word operation, and bit 12 = 1 signifies a byte operation). In addition, the determination of whether a byte swap is to occur depends on which mode (the "Motorola-Like" or the "Intel-Like") is in effect, and on whether an even or odd address is being accessed (see Figures 17 and 18).

Whenever a word of data is read by a Data Memory operation into the processor, it is first placed in the NEXT register. By the time the instruction that reads that word of data is completed, however, the data may have been moved, optionally inverted, or operated on by the ALU, and placed in the TOP register. Whenever a Data Memory operation writes to memory, the data comes from the NEXT register.

The Byte Order Bit is bit 2 of the **Configuration Register**, CR (see Figure 3 in the "RTX Internal Registers" Section). This bit is used to determine whether the default ("Motorola-Like") or byte swap ("Intel-Like") mode will be used in the Data Memory accesses.

Word Access - (Memory Access Opcode, IR Bit 12 = 0)

CR Bit 2 = 0: The "Motorola-Like" mode of word access (also known as the "Big Endian" mode) to an even address (A0 = 0) results in an unaltered transfer of data, as shown in Figure 17. Word access to an odd address (A0 = 1) while in this mode will effectively cause the Byte Order Bit to be complemented and will result in the bytes being swapped.

CR Bit 2 = 1: The "Intel-Like" mode of word access (also known as the "Little Endian" mode) to an even address (A0 = 0) results in a data transfer in which the bytes are swapped. Word access to an odd address (A0 = 1) while in this mode will effectively cause the Byte Order Bit to be complemented with the net result that no byte swap takes place when the data word is transferred. See Figure 17.

Byte Access - (Memory Access Opcode, IR Bit 12 = 1)

CR Bit 2 = 0: During byte mode Memory access, a ***Byte Read*** from an even address in the "Motorola-Like" mode will cause the upper byte (MD15-MD08) of memory data to be read into the lower byte position (MD07-MD00) of NEXT, while the upper byte (MD15 - MD08) is set to 0. A ***Byte Write*** operation accessing an even address will cause the byte to be written from the lower byte position (MD07-MD00) of NEXT into the upper byte position (MD15-MD08) of Memory. The data in the lower byte position (MD07-MD00) in Memory will be left unaltered. Accessing an odd address for either of these operations will cause the Byte Order Bit to be complemented, with the net result that no swap will occur. See Figure 18.

NOTE: These features are for Main Memory data access only, and have no effect on instruction fetches, long literals, or User Data Memory.

CR Bit 2 = 1: Accessing an even address in the "Intel-Like" Mode means that a ***Byte Read*** operation will cause the lower byte of data to be transferred without a swap operation. A ***Byte Write*** in this mode will also result in an unaltered byte transfer. Conversely, accessing a odd address for a Byte operation while in the "Intel-Like" Mode will cause the Byte Order bit to be complemented. In a ***Byte Read*** operation, this will result in the upper byte (MD15-MD08) of data being swapped into the lower byte position (MD07-MD00), while the upper byte is set to 0 (MD15 - MD08 set to 0). See Figure 18. A ***Byte Write*** operation accessing an odd address will cause the byte to be swapped from the lower byte position (MD07-MD00) of the processor register into the upper byte position (MD15-MD08) of the Memory location. The data in the lower byte position (MD07-MD00) in that Memory location will be left unaffected.

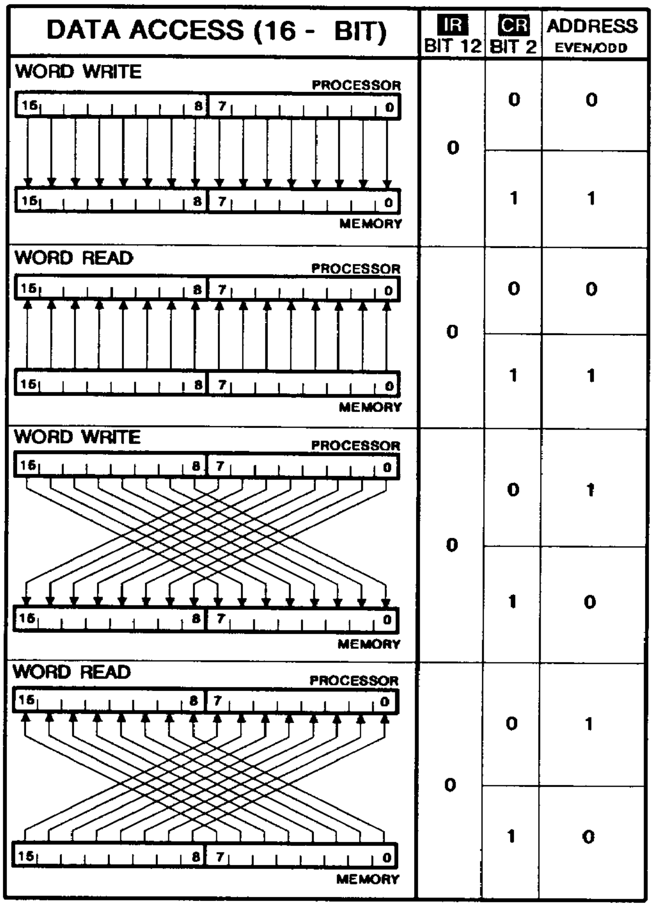


FIGURE 17. MEMORY ACCESS (WORD)

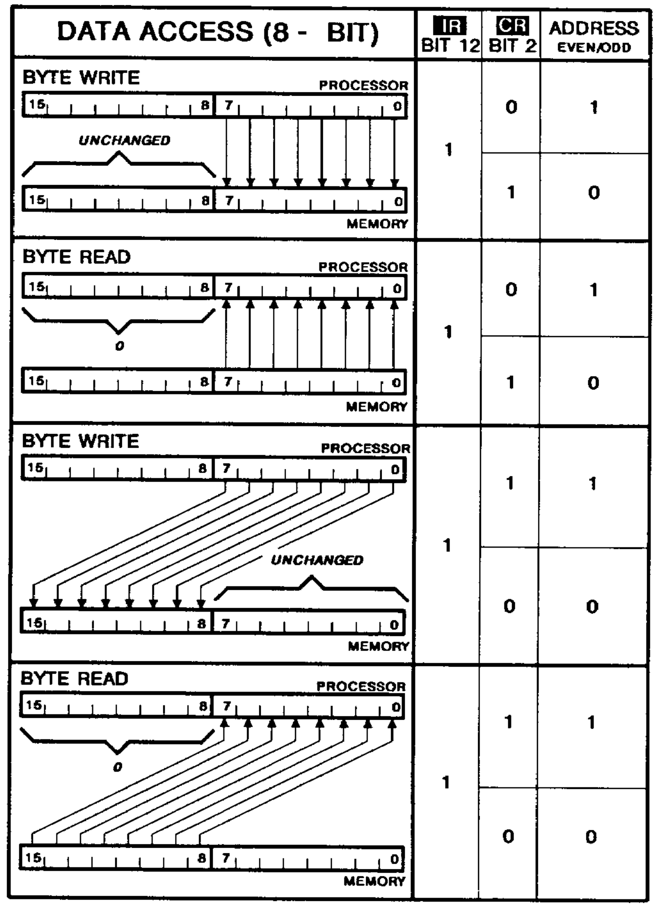


FIGURE 18. MEMORY ACCESS (BYTE)

Software Information

The RTX 2000 TForth compiler from Harris translates Forth-83 source code to RTX 2000 machine instructions. This compiler also provides support for all of the RTX 2000 instructions specific to the processor's registers, peripherals, and ASIC Bus. See the tables which follow for instruction set information.

The compiler instruction set combines multiple high level instructions into single machine instructions. This optimization produces an effective throughput which is faster than the processor's clock speed.

Compilers will be available for C and Prolog for applications which require those languages.

TABLE 8. INSTRUCTION SET SUMMARY

NOTATIONS	
m-read	Read data (byte or word) from memory location addressed by contents of TOP register into NEXT register.
m-write	Write contents (byte or word) of NEXT register into memory location addressed by contents of TOP register.
g-read	Read data from the ASIC address (address field *ggggg* of instruction) into TOP register. A read of one of the on-chip peripheral registers can be done with a ***g-read*** command.
g-write	Write contents of TOP register to ASIC address (address field *ggggg* of instruction). A write to one of the on-chip peripheral registers can be done with a ***g-write*** command.
u-read	Read contents (word only) of User Space location (address field *uuuuu* of instruction) into TOP register.
u-write	Write contents (word only) of TOP register into User Space location (address field *uuuuu* of instruction).
SWAP	Exchange contents of TOP and NEXT registers
DUP	Copy contents of TOP register to NEXT register, pushing previous contents of NEXT onto Stack Memory.
OVER	Copy contents of NEXT register to TOP register, pushing original contents of TOP to NEXT register and original contents of NEXT register to Stack Memory.
DROP	Pop Parameter Stack, discarding original contents of TOP register, leaving the original contents of NEXT in TOP and the original contents of the top Stack Memory location in NEXT.
inv	Perform 1's complement on contents of TOP register, if i bit in instruction is 1.
alu-op	Perform appropriate *cccc* or *aaa* ALU operation from Table 21 on contents of TOP and NEXT registers.
shift	Perform appropriate shift operation (*ssss* field of instruction) from Table 22 on contents of TOP and/or NEXT registers.
d	Push short literal *d* from *ddddd* field of instruction onto Parameter Stack (where *ddddd* contains the actual value of the short literal). The original contents of TOP are pushed into NEXT, and the original contents of NEXT are pushed onto Stack Memory.
D	Push long literal *D* from next sequential location in program memory onto Parameter Stack. The original contents of TOP are pushed into NEXT, and the original contents of NEXT are pushed onto Stack Memory.
R	Perform a Return From Subroutine if bit = 1.
x	Bit fields containing x's are ignored by the processor.

TABLE 9. INSTRUCTION REGISTER BIT FIELDS (BY FUCTION)

FUNCTION CODE	DEFINITION
ggggg	Address field for ASIC Bus locations
uuuuu	Address field for User Space memory locations
cccc *aaa*	ALU functions (see Table 21)
ddddd	Short literals (containing a value from 0 to 31)
ssss	Shift Functions (see Table 22)

TABLE 10. RTX 2000 I AND PC ACCESS OPERATIONS*

OPERATION *(g-read, g-write)*	RETURN BIT VALUE	ASIC ADDRESS *ggggg*	REGISTER	FUNCTION
R	0	00000	I	Pushes the contents of I into TOP (with no pop of the Return Stack)
R	1	00000	I	Pushes the contents of I into TOP, then performs a Subroutine Return
W	0	00000	I	Pops the contents of TOP into I (with no push of the Return Stack)
W	1	00000	I	Performs a Subroutine Return, then pushes the contents of TOP into I
R	0	00001	I	Pushes the contents of I into TOP, popping the Return Stack
R	1	00001	I	Pushes the contents of I into TOP without popping the Return Stack, then executes the Subroutine Return
W	0	00001	I	Pushes the contents of TOP into I popping the Parameter Stack
W	1	00001	I	Performs a Subroutine Return, then pushes the contents of TOP into I
R	0	00010	I	Pushes the contents of I shifted left by one bit, into TOP (the Return Stack is not popped)
R	1	00010	I	Pushes the contents of I shifted left by one bit, into TOP (the Return Stack is not popped), then performs a Subroutine Return
W	0	00010	I	Pushes the contents of TOP into I as a "stream" count, indicating that the next instruction is to be performed a specified number of times; the Parameter Stack is popped
W	1	00010	I	Performs a Subroutine Return, then pushes the stream count into I
R	0	00111	PC	Pushes the contents of PC into TOP
R	1	00111	PC	Pushes the contents of PC into TOP, then performs a Subroutine Return
W	0	00111	PC	Performs a Subroutine Call to the address contained in TOP, popping the Parameter Stack
W	1	00111	PC	Pushes the contents of TOP onto the Return Stack before executing the Subroutine Return

* See the RTX Programmer's Reference Manual for a complete listing of typical software functions as well as instructions unique to the RTX 2000.

TABLE 11. SUBROUTINE CALL INSTRUCTIONS

INSTRUCTION CODE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OPERATION
0	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	Call word address aaaa aaaa aaaa aaa 0, in the page indicated by CPR. This address is produced when the processor performs a left shift on the address in the instruction code.

Subroutine Call Bit
(Bit 15 = 0: Call)
(Bit 15 = 1: No Call)

TABLE 12. SUBROUTINE RETURN

INSTRUCTION CODE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OPERATION
x	x	x	x	x	x	x	x	x	x	R	x	x	x	x	x	Return from subroutine

Subroutine Return Bit*
(Bit 5, R = 0: No Return
R = 1: Return)

* Does not apply to Subroutine Call or Branch Instructions.

TABLE 13. BRANCH INSTRUCTIONS

INSTRUCTION CODE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OPERATION
1	0	0	0	0	b	b	a	a	a	a	a	a	a	a	a	DROP and branch if TOP = 0
1	0	0	0	1	b	b	a	a	a	a	a	a	a	a	a	Branch if TOP = 0;
1	0	0	1	0	b	b	a	a	a	a	a	a	a	a	a	Unconditional branch
1	0	0	1	1	b	b	a	a	a	a	a	a	a	a	a	Branch and decrement I if I ≠ 0 Pop I if I = 0

Branch Address* (bits 10–0)

* See the Programmer's Reference Manual for further information regarding the branch address field.

TABLE 14. REGISTER AND I/O ACCESS INSTRUCTIONS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OPERATION	
1	0	1	1	0	0	0	i	0	0	R	g	g	g	g	g	*g-read* DROP	inv
1	0	1	1	1	1	1	i	0	0	R	g	g	g	g	g	*g-read*	inv
1	0	1	1	c	c	c	c	0	0	R	g	g	g	g	g	*g-read* OVER	alu-op
1	0	1	1	0	0	0	i	1	0	R	g	g	g	g	g	DUP *g-write*	inv
1	0	1	1	1	1	1	i	1	0	R	g	g	g	g	g	*g-write*	inv
1	0	1	1	c	c	c	c	1	0	R	g	g	g	g	g	*g-read* SWAP	alu-op

TABLE 15. SHORT LITERAL INSTRUCTIONS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OPERATION	
1	0	1	1	0	0	0	i	x	1	R	d	d	d	d	d	d DROP	inv
1	0	1	1	1	1	1	i	0	1	R	d	d	d	d	d	d	inv
1	0	1	1	c	c	c	c	0	1	R	d	d	d	d	d	d OVER	alu-op
1	0	1	1	1	1	1	i	1	1	R	d	d	d	d	d	d SWAP DROP	inv
1	0	1	1	c	c	c	c	1	1	R	d	d	d	d	d	d SWAP	alu-op

TABLE 16. LONG LITERAL INSTRUCTIONS

INSTRUCTION CODE																OPERATION	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	(1ST CYCLE)	(2ND CYCLE)
1	1	0	1	0	0	0	i	x	0	R	x	x	x	x	x	D SWAP	inv
1	1	0	1	1	1	1	i	0	0	R	x	x	x	x	x	D SWAP	SWAP inv
1	1	0	1	c	c	c	c	0	0	R	x	x	x	x	x	D SWAP	SWAP OVER alu-op
1	1	0	1	1	1	1	i	1	0	R	x	x	x	x	x	D SWAP	DROP inv
1	1	0	1	c	c	c	c	1	0	R	x	x	x	x	x	D SWAP	alu-op

TABLE 17. MEMORY ACCESS INSTRUCTIONS

INSTRUCTION CODE																OPERATION	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	(1ST CYCLE)	(2ND CYCLE)
1	1	1	s	0	0	0	i	0	0	R	x	x	x	x	x	***m-read*** SWAP	inv
1	1	1	s	1	1	1	i	0	0	R	x	x	x	x	x	***m-read*** SWAP	SWAP inv
1	1	1	s	c	c	c	c	0	0	R	x	x	x	x	x	***m-read*** SWAP	SWAP OVER alu-op
1	1	1	s	0	0	0	p	0	1	R	x	x	x	x	x	{SWAP DROP} DUP ***m-read*** SWAP	NOP
1	1	1	s	1	1	1	p	0	1	R	d	d	d	d	d	{SWAP DROP} ***m-read d***	NOP
1	1	1	s	a	a	a	p	0	1	R	d	d	d	d	d	{SWAP DROP} DUP ***m-read*** SWAP d SWAP alu-op	NOP
1	1	1	s	0	0	0	i	1	0	R	x	x	x	x	x	OVER SWAP ***m-write***	inv
1	1	1	s	1	1	1	i	1	0	R	x	x	x	x	x	OVER SWAP ***m-write***	DROP inv
1	1	1	s	c	c	c	c	1	0	R	x	x	x	x	x	***m-read*** SWAP	alu-op
1	1	1	s	0	0	0	p	1	1	R	x	x	x	x	x	{OVER SWAP} SWAP OVER ***m-write***	NOP
1	1	1	s	0	0	0	p	1	1	R	d	d	d	d	d	{OVER SWAP} ***m-write*** d	NOP
1	1	1	s	a	a	a	p	1	1	R	d	d	d	d	d	{OVER SWAP} SWAP OVER ***m-write*** d SWAP alu-op	NOP

If s = 0, Memory is accessed by word
If s = 1, Memory is accessed by byte

If (p = 0), perform
{SWAP DROP} and
{OVER SWAP}

TABLE 18. USER SPACE INSTRUCTIONS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OPERATION	
1	1	0	0	0	0	0	i	0	0	R	u	u	u	u	u	*u-read* SWAP	inv
1	1	0	0	1	1	1	i	0	0	R	u	u	u	u	u	*u-read* SWAP	SWAP inv
1	1	0	0	c	c	c	c	0	0	R	u	u	u	u	u	*u-read* SWAP	SWAP OVER alu-op
1	1	0	0	0	0	0	i	1	0	R	u	u	u	u	u	DUP *u-write*	inv
1	1	0	0	1	1	1	i	1	0	R	u	u	u	u	u	DUP *u-write*	DROP inv
1	1	0	0	c	c	c	c	1	0	R	u	u	u	u	u	*u-read* SWAP	alu-op

TABLE 19. ALU FUNCTION INSTRUCTIONS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OPERATION	
1	0	1	0	0	0	0	i	0	0	R	0	s	s	s	s		inv shift
1	0	1	0	1	1	1	i	0	0	R	0	s	s	s	s	DROP DUP	inv shift
1	0	1	0	c	c	c	c	0	0	R	0	s	s	s	s	OVER SWAP	alu-op shift
1	0	1	0	0	0	0	i	0	1	R	0	s	s	s	s	SWAP DROP	inv shift
1	0	1	0	1	1	1	i	0	1	R	0	s	s	s	s	DROP	inv shift
1	0	1	0	c	c	c	c	0	1	R	0	s	s	s	s		alu-op shift
1	0	1	0	0	0	0	i	1	0	R	0	s	s	s	s	SWAP DROP DUP	inv shift
1	0	1	0	1	1	1	i	1	0	R	0	s	s	s	s	SWAP	inv shift
1	0	1	0	c	c	c	c	1	0	R	0	s	s	s	s	SWAP OVER	alu-op shift
1	0	1	0	0	0	0	i	1	1	R	0	s	s	s	s	DUP	inv shift
1	0	1	0	1	1	1	i	1	1	R	0	s	s	s	s	OVER	inv shift
1	0	1	0	c	c	c	c	1	1	R	0	s	s	s	s	OVER OVER	alu-op shift

TABLE 20. STEP MATH FUNCTIONS *

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OPERATION
1	0	1	0	x	x	x	x	x	x	x	1	x	x	x	x	* These instructions perform multi-step math functions such as division and square root functions; see the Programmer's Reference Manual for more information on this entire class of instructions.

TABLE 21. ALU LOGIC FUNCTIONS/OPCODES

cccc	*aaa*	FUNCTION
0010	001	AND
0011		NOR
0100	010	SWAP -
0101		SWAP - With Borrow
0110	011	OR
0111		NAND
1000	100	+
1001		+ With Carry
1010	101	XOR
1011		XOR
1100	110	-
1101		- With Borrow

TABLE 22. SHIFT FUNCTIONS

SHIFT *ssss*	NAME	FUNCTION	STATUS OF C	TOP REGISTER			NEXT REGISTER		
				T15	Tn	T0	N15	Nn	N0
0000		No Shift	CY	Z15	Zn	Z0	TN15	TNn	TN0
0001	0<	Sign extend	CY	Z15	Z15	Z15	TN15	TNn	TN0
0010	2*	Arithmetic Left Shift	Z15	Z14	Zn-1	0	TN15	TNn	TN0
0011	2*c	Rotate Left	Z15	Z14	Zn-1	CY	TN15	TNn	TN0
0100	cU2/	Right Shift Out of Carry	0	CY	Zn+1	Z1	TN15	TNn	TN0
0101	c2/	Rotate Right Through Carry	Z0	CY	Zn+1	Z1	TN15	TNn	TN0
0110	U2/	Logical Right Shift	0	0	Zn+1	Z1	TN15	TNn	TN0
0111	2/	Arithmetic Right Shift	Z15	Z15	Zn+1	Z1	TN15	TNn	TN0
1000	N2*	Arithmetic Left Shift of NEXT	CY	Z15	Zn	Z0	TN14	TNn-1	0
1001	N2*c	Rotate NEXT Left	CY	Z15	Zn	Z0	TN14	TNn-1	CY
1010	D2*	32-bit Arithmetic Left Shift	Z15	Z14	Zn-1	TN15	TN14	TNn-1	0
1011	D2*c	32-bit Rotate Left	Z15	Z14	Zn-1	TN15	TN14	TNn-1	CY
1100	cUD2/	32-bit Right Shift Out of Carry	0	CY	Zn+1	Z1	Z0	TNn+1	TN1
‡ 1101	cD2/	32-bit Rotate Right Through Carry	TN0	CY	Zn+1	Z1	Z0	TNn+1	TN1
1110	UD2/	32-bit Logical Right Shift	0	0	Zn+1	Z1	Z0	TNn+1	TN1
1111	D2/	32-bit Arithmetic Right Shift	Z15	Z15	Zn+1	Z1	Z0	TNn+1	TN1

‡ See the Programmer's Reference Manual

Where:
- T15 -Most significant bit of TOP
- Tn -Typical bit of TOP
- T0 -Least significant bit of TOP
- N15 -Most significant bit of NEXT
- Nn -Typical bit of NEXT
- N0 -Least significant bit of NEXT
- C -Carry bit
- CY -Carry bit before operation
- Zn -ALU output
- Z15 -Most significant bit 15 of ALU output
- TNn -Original value of typical bit of NEXT

Absolute Maximum Ratings

Supply Voltage .... +8.0V
Input, Output, or I/O Voltage Applied ... GND - 0.5V to VCC + 0.5V
Storage Temperature Range .... -65°C to +150°C
Maximum Package Power Dissipation .... 2 Watts
θ_{ja} .... 41°C/W (PGA Package)
θ_{jc} .... 17°C/W (PGA Package)

Gate Count 28,000
Junction Temperature .... +175°C
Lead Temperature (Soldering, Ten Seconds).......... +300°C

CAUTION: Stresses above those listed in the "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operation section of the specification is not implied.

Operating Conditions

Operating Voltage Range .... +4.5V to +5.5V
Operating Temperature Range: RTX 2000I ..... -40°C to +85°C

Maximum Rise and Fall Times For ICLK.......... 20ns

D.C. Electrical Specifications VCC = 5V, ±10%, T_A = -40°C to +85°C

SYMBOL	PARAMETER		MIN	MAX	UNITS	COMMENTS
VIH	Logical One Input Voltage	NMI, RESET, ICLK	VCC x 0.7	-	V	Tested at VCC = 5.5V
		Other Inputs	2.0	-	V	
VIL	Logical Zero Input Voltage		-	0.8	V	
VOH	High Output Voltage		3.5	-	V	IOH = -4mA
			VCC - 0.4	-	V	IOH = -100µA
VOL	Low Output Voltage		-	0.4	V	IOL = 4mA
II	Input Leakage Current		-1	1	µA	VI = VCC or GND
IIO	I/O Leakage Current		-10	10	µA	VO = VCC or GND
ICCSB	Standby Power Supply Current		-	500	µA	VI = VCC or GND (Note1)
ICCOP	Operating Power Supply Current		-	10	mA	VI = VCC or GND; f (ICLK) = 1 MHz; (Note 2) Outputs Unloaded

NOTES: 1. Typical ICCSB: 10µA. The RTX 2000 is a static CMOS part. Therefore ICCSB > 0 is due to leakage currents.

2. Operating supply current is proportional to frequency. Typical ICCOP: 5mA/MHz.

Capacitance (T_A = +25°C; All measurements referred to device GND)

SYMBOL	PARAMETER	TYP	UNITS	TEST CONDITIONS
CI	Input Capacitance	10	pF	f = 1MHz
CIO	I/O Capacitance	10	pF	f = 1MHz

A.C. Electrical Specifications VCC = 5V, ±10%, T_A = -40°C to +85°C

CLOCK, WAIT AND TIMER TIMING (Notes 1, 2, and 3)

SYMBOL	PARAMETER	8MHz		10MHz		UNITS	COMMENTS
		MIN	MAX	MIN	MAX		
REQUIREMENTS							
t1	ICLK Period	62	-	50	-	ns	
t2	ICLK High Time	24	-	20	-	ns	
t3	ICLK Low Time	24	-	20	-	ns	
t4	WAIT Set Up Time	5	-	5	-	ns	
t5	WAIT Hold Time	3	-	3	-	ns	
t6	EI High to EI High	t1x4	-	t1x4	-	ns	External Clock/Timer Input
t7	EI High Time	10	-	10	-	ns	External Clock/Timer Input
t8	EI Low Time	10	-	10	-	ns	External Clock/Timer Input
RESPONSES							
t11	ICLK to TCLK High	3	25	3	24	ns	
t12	TCLK Low Time	52	-	40	-	ns	
t13	TCLK High Time	64	-	52	-	ns	
t15	ICLK to PCLK High	3	25	3	25	ns	
t16	PCLK Low Time	52	-	40	-	ns	
t17	PCLK High Time	64	-	52	-	ns	
t19	ICLK to TCLK Low	-	35	-	32	ns	
t20	ICLK to PCLK Low	-	28	-	26	ns	

NOTES: 1. High and low input levels for A.C. test: ICLK, NMI, and RESET: 4.0V and 0.4V
Other Inputs: 2.4V and 0.4V

2. Output load: 100pF.

3. 10MHz specifications are tested with **CYCEXT** set to 1. For guaranteed operation of User Instructions and Interrupt Acknowledge cycles, **CYCEXT** should be set to 1 for f(TCLK) > 8MHz.

A.C. Electrical Specifications (Continued) VCC = 5V, ±10%, T_A = -40°C to +85°C

MEMORY BUS TIMING (Notes 1, 2, and 3)

SYMBOL	PARAMETER	8MHz		10MHz		UNITS	COMMENTS
		MIN	MAX	MIN	MAX		
REQUIREMENTS							
t21	MD Setup Time	14	-	12	-	ns	Read Cycle
t22	MD Hold Time	4	-	4	-	ns	Read Cycle
RESPONSES							
t26A	PCLK to MA Valid: User/INTA cycles	-	62	-	60	ns	(Note 5)
t26B	PCLK to MA Valid: non-User/INTA cycles	-	62	-	50	ns	(Note 5)
t28	MA Hold Time	20	-	20	-	ns	(Note 6)
t29	PCLK to MR/$\overline{W}$, UDS, LDS, NEW and BOOT Valid	-	50	-	44	ns	(Note 5)
t31	MR/$\overline{W}$, UDS, LDS, NEW, BOOT Hold Time	20	-	20	-	ns	(Note 6)
t32	PCLK to MD Valid	-	16	-	14	ns	Write Cycle
t33	MD Hold Time	20	-	20	-	ns	Write Cycle (Note 6)
t34	MD Enable Time	-2	-	-2	-	ns	Write Cycle (Note 4)
t35	PCLK to MD Disable Time	-	50	-	44	ns	Write Cycle (Notes 4, 5)

NOTES:
1. High and low input levels for A.C. test: ICLK, NMI, and RESET: 4.0V and 0.4V
   Other Inputs: 2.4V and 0.4V
2. Output load: 100pF.
3. 10MHz specifications are tested with **CYCEXT** set to 1. For guaranteed operation of User Instructions and Interrupt Acknowledge cycles, **CYCEXT** should be set to 1 for f(TCLK) > 8MHz.
4. Output enable and disable times are characterized only.
5. Tested with t1 at specified minimum and t2 = 0.5*t1.
   For t2 > 0.5*t1(min), add t2 - (0.5*t1(min)) to this specification.
6. Tested with t1 at specified minimum and t2 = 0.5*t1.
   For t2 < 0.5*t1(min), subtract (0.5*t1(min)) - t2 from this specification.

A.C. Electrical Specifications (Continued) VCC = 5V, ±10%, T_A = -40°C to +85°C

ASIC BUS AND INTERRUPT TIMING (Notes 1, 2, and 3)

SYMBOL	PARAMETER	8MHz MIN	8MHz MAX	10MHz MIN	10MHz MAX	UNITS	COMMENTS
REQUIREMENTS							
t40	GD Read Setup to PCLK	32	-	30	-	ns	Read Cycle
t41	GD Read Setup to $\overline{GIO}$	37	-	35	-	ns	Read Cycle
t42	GD Read Hold from $\overline{GIO}$	0	-	0	-	ns	Read Cycle
t43	GD Read Hold from PCLK	0	-	0	-	ns	Read Cycle
t44	EI/NMI Setup Time	15	-	15	-	ns	INT/NMI Cycle
t46	INTSUP Setup Time	20	-	20	-	ns	
t47	INTSUP Hold Time	0	-	0	-	ns	
RESPONSES							
t48	PCLK High to $\overline{GIO}$ Low	52	-	46	-	ns	
t49	$\overline{GIO}$ Low Time	52	-	40	-	ns	
t50	ICLK High to $\overline{GIO}$ Low	-	35	-	30	ns	
t51	ICLK High to $\overline{GIO}$ High	-	35	-	32	ns	
t52	PCLK to GA Valid	-	45	-	40	ns	(Note 5)
t54	$\overline{GIO}$ to GA Hold Time	12	-	12	-	ns	(Note 6)
t56	PCLK to GR/$\overline{W}$ Valid	-	52	-	46	ns	(Note 5)
t58	$\overline{GIO}$ to GR/$\overline{W}$ Read Hold Time	12	-	12	-	ns	(Note 6)
t61	GD Enable Time	0	-	0	-	ns	Write Cycle (Note 4)
t62	GD Valid Time	-	16	-	14	ns	Write Cycle
t63	$\overline{GIO}$ to GD Hold Time	12	-	12	-	ns	Write Cycle (Note 6)
t65	$\overline{GIO}$ to GD Disable Time	-	50	-	44	ns	Write Cycle (Notes 4, 5)
t67	PCLK to INTA High Time	-	25	-	25	ns	INTA Cycle
t68	INTA Hold Time	0	-	0	-	ns	INTA Cycle

NOTES:
1. High and low input levels for A.C. test: ICLK, NMI and RESET: 4.0V and 0.4V
   Other Inputs: 2.4V and 0.4V
2. Output load: 100pF.
3. 10MHz specifications are tested with **CYCEXT** set to 1.
   For guaranteed operation of User Instructions and Interrupt Acknowledge cycles, **CYCEXT** should be set to 1 for f(TCLK) > 8MHz.
4. Output enable and disable times are characterized only.
5. Tested with t1 at specified minimum and t2 = 0.5*t1.
   For t2 > 0.5*t1(min), add t2 - (0.51*t1(min)) to this specification.
6. Tested with t1 at specified minimum and t2 = 0.5* t1.
   For t2 < 0.5*t1(min), subtract (0.5*t1(min)) - t2 from this specification.

Timing Diagrams

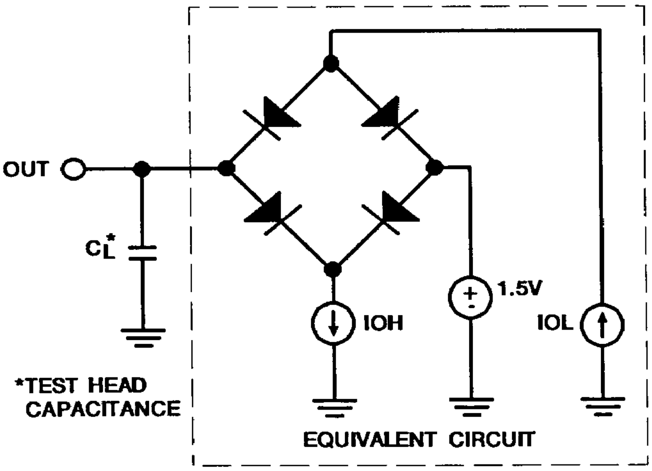


FIGURE 19. TEST CIRCUIT

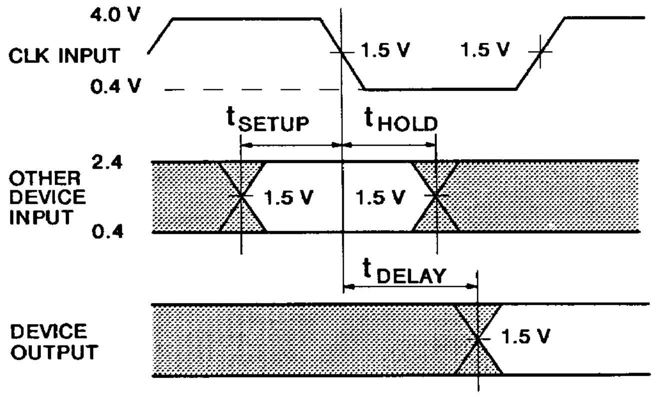


FIGURE 20. A.C. DRIVE AND MEASURE POINTS - CLK INPUT

NOTE: For A.C. testing, input rise and fall times are driven at 1ns/volt.

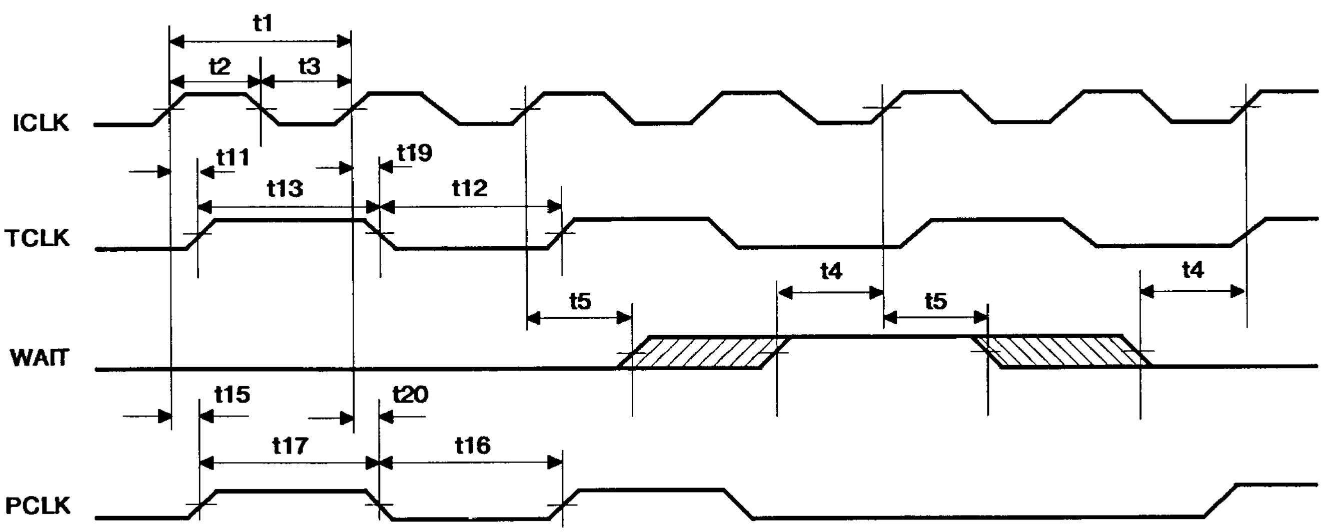


FIGURE 21. CLOCK AND WAIT TIMING

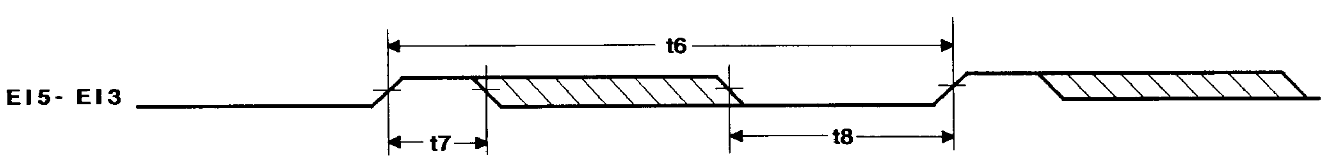


FIGURE 22. TIMER/COUNTER TIMING

Timing Diagrams (Continued)

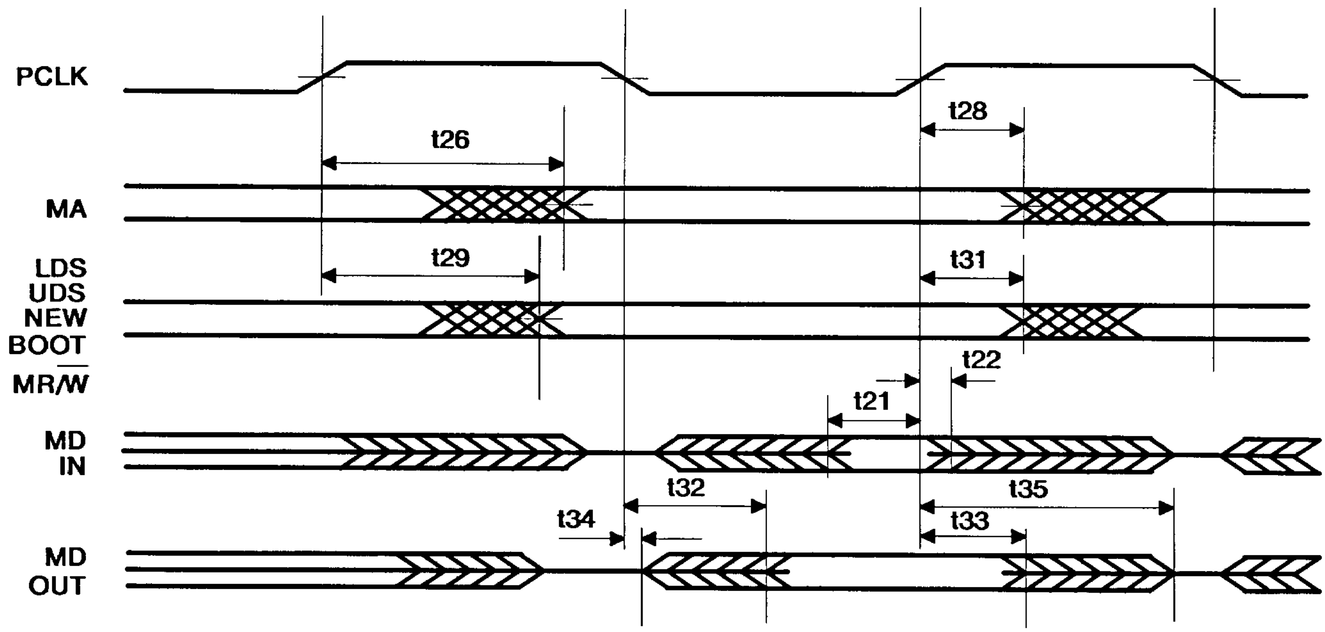


FIGURE 23. MEMORY BUS TIMING

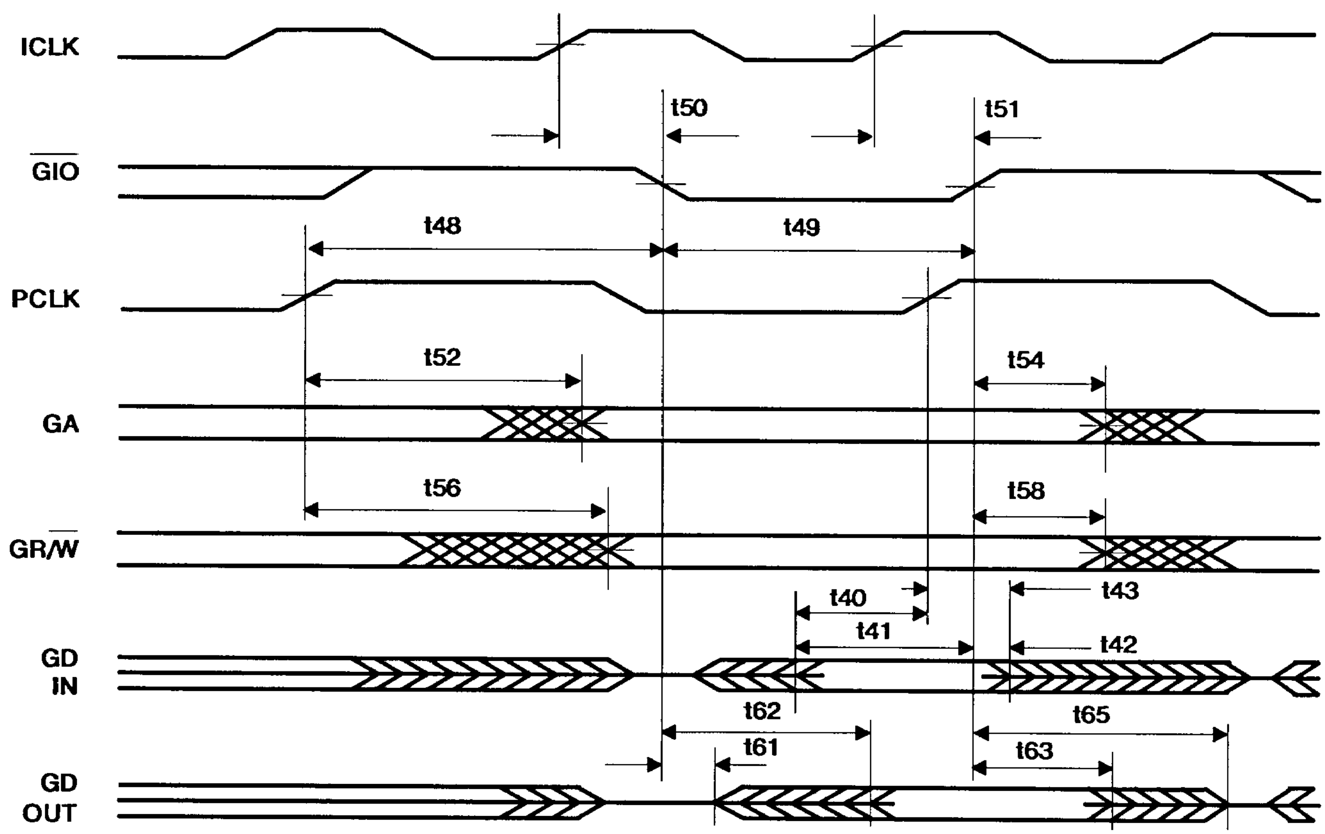


FIGURE 24. ASIC BUS TIMING

Timing Diagrams (Continued)

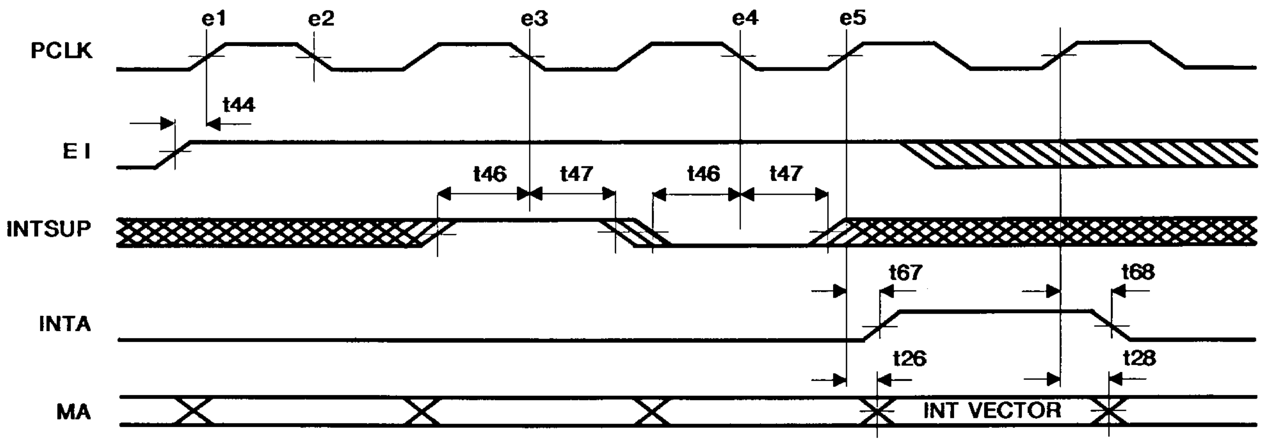


FIGURE 25. INTERRUPT TIMING: WITH INTERRUPT SUPPRESSION

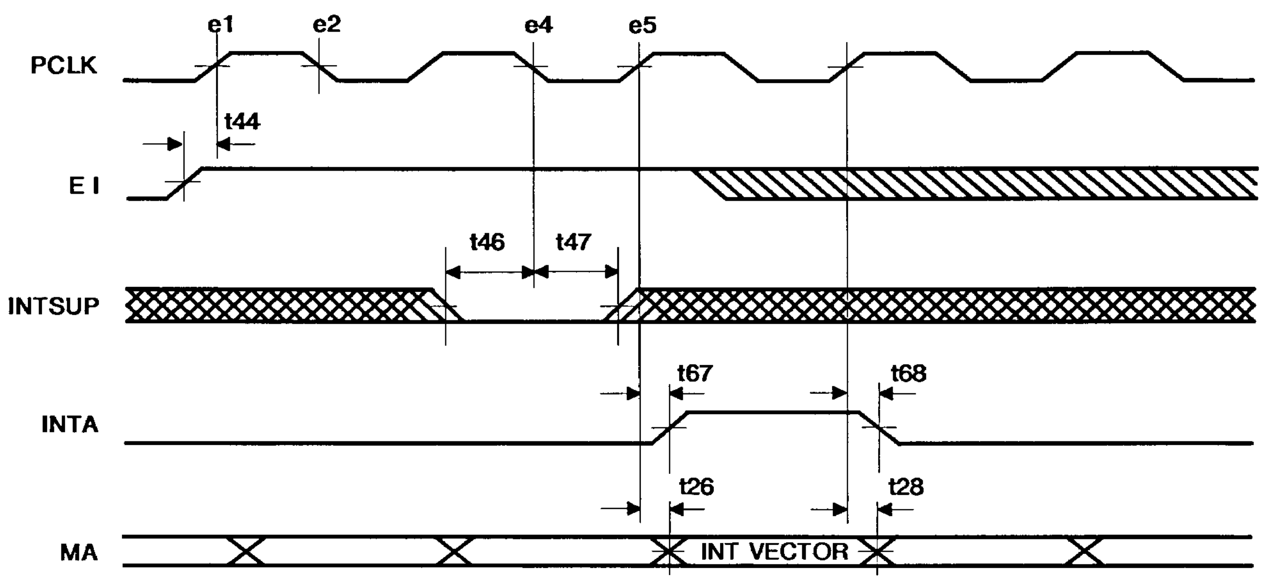


FIGURE 26. INTERRUPT TIMING: WITH NO INTERRUPT SUPPRESSION

Timing Diagrams (Continued)

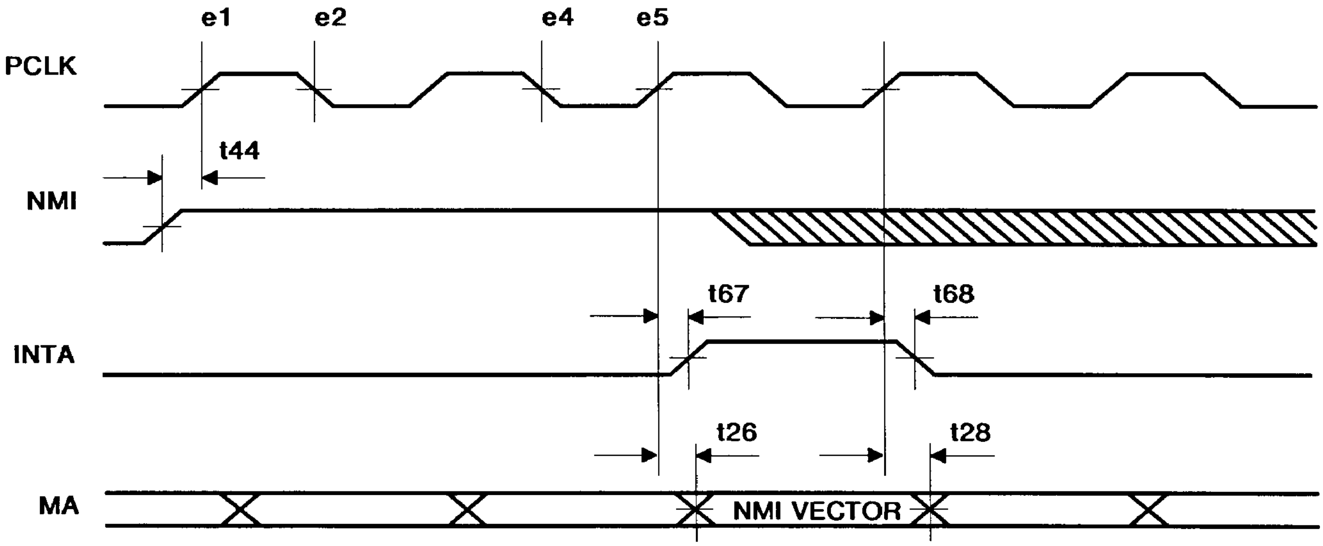


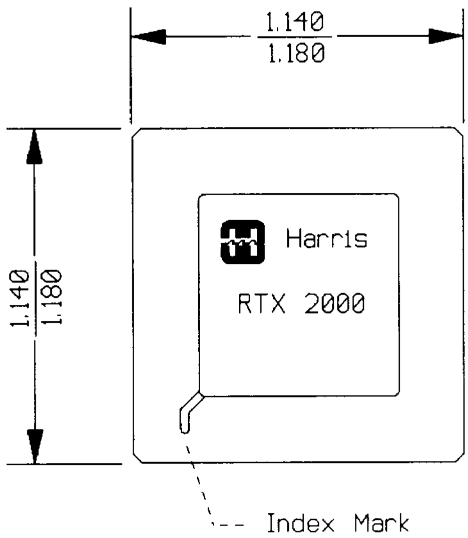
FIGURE 27. NON-MASKABLE INTERRUPT TIMING

NOTES:

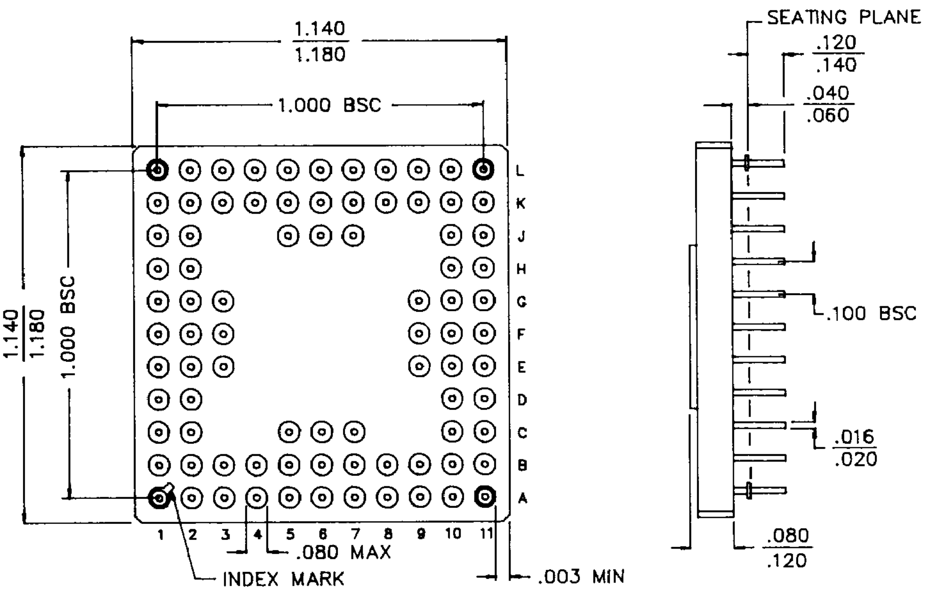
1. Events in an interrupt sequence are as follows:
   - e1. The Interrupt Controller samples the interrupt request inputs on the rising edge of PCLK. If NMI rises between e1 and e5, the interrupt vector will be for NMI.
   - e2. If any interrupt requests were sampled, the Interrupt Controller issues an interrupt request to the core on the falling edge of PCLK.
   - e3. The core samples the state of the interrupt requests from the Interrupt Controller on the falling edge of PCLK. If INTSUP is high, maskable interrupts will not be detected at this time.
   - e4. When the core samples an interrupt request on the falling edge of PCLK, an Interrupt Acknowledge cycle will begin on the next rising edge of PCLK.
   - e5. Following the detection of an interrupt request by the core an Interrupt Acknowledge cycle begins. The interrupt vector will be based on the highest priority interrupt request active at this time.
2. t44 is only required to determine the cycle in which Interrupt Acknowledgment will occur.
3. Interrupt requests should be held active until the Interrupt Acknowledge cycle for that interrupt occurs.

Packaging

84 PIN GRID ARRAY
TOP VIEW



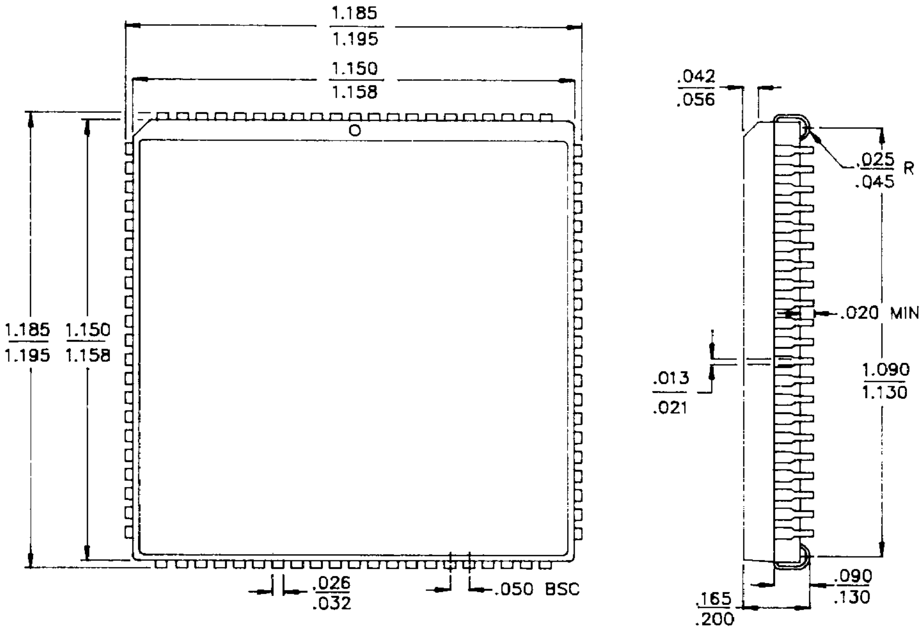
84 PIN GRID ARRAY
BOTTOM VIEW



NOTE: All Dimensions are $\frac{\text{Min}}{\text{Max}}$, Dimensions are in inches.

Packaging (Continued)

84 LEAD PLCC



Ordering Information

COMMERCIAL/INDUSTRIAL

RTX 2000 G I -10

FAMILY
RTX (Real Time Express)

PART NUMBER

PACKAGE TYPE
G: PGA
J: PLCC

TEMPERATURE RANGE
I: Industrial
-40°C to +85°C

SPEED/PERFORMANCE
10: 10MHz
8: 8MHz

Index

NOTICE: Harris Semiconductor's products are sold by description only. Harris Semiconductor reserves the right to make changes in circuit design and/or specifications at any time without notice. Accordingly, the reader is cautioned to verify that data sheets are current before placing orders.

Sales Offices

U.S. HEADQUARTERS
Harris Semiconductor
1301 Woody Burke Road
Melbourne, Florida 32902
TEL: (407) 724-3739

EUROPEAN HEADQUARTERS
Harris Semiconductor
Mercure Centre
Rue de la Fusse 100
Brussels, Belgium 1130
TEL: (32) 246-2201

SOUTH ASIA
Harris Semiconductor H.K. Ltd
13/F Fourseas Building
208-212 Nathan Road
Tsimshatsui, Kowloon
Hong Kong
TEL: (852) 3-723-6339

NORTH ASIA
Harris K.K.
Shinjuku NS Bldg. Box 6153
2-4-1 Nishi-Shinjuku
Shinjuku-Ku, Tokyo 163 Japan
TEL: 81-3-345-8911

DISTRIBUTORS IN U.S.A.
Almac Electronics
Anthem Electronics
Electronics Marketing Corporation
Falcon Electronics
Gerber Electronics
Hall-Mark Electronics
Hamilton/Avnet Corporation
Newark Electronics
Schweber Electronics
Wyle Laboratories

DISTRIBUTORS IN CANADA
Hamilton/Avnet Corporation
ITT Multicomponents

HARRIS

COMMERCIAL PRODUCTS GROUP

016890 ✓