Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Not For New Designs
Core Processor	Rabbit 2000
Number of Cores/Bus Width	1 Core, 8-Bit
Speed	30MHz
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	-55°C ~ 85°C (TA)
Security Features	-
Package / Case	100-BQFP
Supplier Device Package	100-PQFP (20x14)
Purchase URL	https://www.e-xfl.com/product-detail/digi-international/20-668-0003

- The built-in main clock oscillator uses an external crystal or more usually a ceramic resonator. Typical resonator frequencies are in the range of 1.8 MHz to 29.5 MHz. Since precision timing is available from the separate 32.768 kHz oscillator, a low-cost ceramic resonator with ½ percent error is generally satisfactory. The clock can be doubled or divided by 8 to modify speed and power dynamically. The I/O clock, which clocks the serial ports, is divided separately so as not to affect baud rates and timers when the processor clock is divided or multiplied. For ultra low power operation, the processor clock can be driven from the separate 32.768 kHz oscillator and the main oscillator can be powered down. This allows the processor to operate at approximately 100 µA and still execute instructions at the rate of approximately 10,000 instructions per second. This is a powerful alternative to sleep modes of operation used by other processors. The current is approximately 65 mA at 25 MHz and 5 V. The current is proportional to voltage and clock speed—at 3.3 V and 7.68 MHz the current would be 13 mA, and at 1 MHz the current is reduced to less than 2 mA. Flash memory with automatic power down (from AMD) should be used for operation at the lowest power.
- The excellent floating-point performance is due to a tightly coded library and powerful processing capability. For example, a 25 MHz clock takes 14 µs for a floating add, 13 µs for a multiply, and 40 µs for a square root. In comparison, a 386EX processor running with an 8-bit bus at 25 MHz and using Borland C is about 10 times slower.
- There is a built-in watchdog timer.
- The standard 10-pin programming port eliminates the need for in-circuit emulators. A very simple 10 pin connector can be used to download and debug software using Rabbit Semiconductor's Dynamic C and a simple connection to a PC serial port. The incremental cost of the programming port is extremely small.

- Input/output instructions are now accomplished by normal memory access instructions prefixed by an op code byte to indicate access to an I/O space. There are two I/O spaces, internal peripherals and external I/O devices.

Some Z80 and Z180 instructions have been deleted and are not supported by the Rabbit (see Chapter 19, "Differences Rabbit vs. Z80/Z180 Instructions,"). Most of the deleted instructions are obsolete or are little-used instructions that can be emulated by several Rabbit instructions. It was necessary to remove some instructions to free up 1-byte op codes needed to implement new instructions efficiently. The instructions were not re-implemented as 2-byte op codes so as not to waste on-chip resources on unimportant instructions. Except for the instruction **EX (SP),HL**, the original Z180 binary encoding of op codes is retained for all Z180 instructions that are retained.

3.3.1 Load Immediate Data To a Register

A constant that follows the op code in the instruction stream can generally be loaded to any register, except PC, AF, IP and F. (Load to the PC is a jump instruction.) This includes the alternate registers on the Rabbit, but not on the Z180. Some example instructions appear below.

```
LD A,3
LD HL,456
LD BC',3567  ; not possible on Z180
LD H',0x4A   ; not possible on Z180
LD IX,1234
LD C,54
```

Byte loads require four clocks, word loads require six clocks. Loads to IX, IY or the alternate registers generally require two extra clocks because the op code has a 1-byte prefix.

3.3.2 Load or Store Data from or to a Constant Address

```
LD A,(mn)    ; loads 8 bits from address mn
LD A',(mn)   ; not possible on Z180
LD (mn),A
LD HL,(mn)   ; load 16 bits from the address specified by mn
LD HL',(mn)  ; to alternate register, not possible Z180
LD (mn),HL
```

Similar 16-bit loads and stores exist for DE, BC, SP, IX and IY.

It is possible to load data to the alternate registers, but it is not possible to store the data in the alternate register directly to memory.

```
LD A',(mn)     ; allowed
** LD (mn),D'  ; **** not a legal instruction!
** LD (mn),DE' ; **** not a legal instruction!
```

3.3.3 Load or Store Data Using an Index Register

An index register is a 16-bit register, usually IX, IY, SP or HL, that is used for the address of a byte or word to be fetched from or stored to memory. Sometimes an 8-bit offset is added to the address either as a signed or unsigned number. The 8-bit offset is a byte in the instruction word. BC and DE can serve as index registers only for the special cases below.

```
LD A,(BC)
LD A',(BC)
LD (BC),A
LD A,(DE)
LD A',(DE)
LD (DE),A
```

Other 8-bit loads and stores are the following.

```
LD r,(HL)      ; r is any of 7 registers A, B, C, D, E, H, L
LD g,(HL)      ; same but alternate register destination
LD (HL),r      ; r is any of the 7 registers above
               ; or an immediate data byte
** LD (HL),g   ;**** not a legal instruction!
LD r,(IX+d)    ; r is any of 7 registers, d is -128 to +127 offset
LD g,(IX+d)    ; same but alternate destination
LD (IX+d),r    ; r is any of 7 registers or an immediate data byte
LD (IY+d),r    ; IX or IY can have offset d
```

The following are 16-bit indexed loads and stores. None of these instructions exists on the Z180 or Z80. The only source for a store is **HL**. The only destination for a load is **HL** or **HL**'.

```
LD HL,(SP+d)    ; d is an offset from 0 to 255.
                ; 16-bits are fetched to HL or HL'
LD (SP+d),HL    ; corresponding store
LD HL,(HL+d)    ; d is an offset from -128 to +127,
                ; uses original HL value for addressing
                ; l=(HL+d), h=(HL+d+1)
LD HL',(HL+d)
LD (HL+d),HL
LD (IX+d),HL    ; store HL at address pointed to
                ; by IX plus -128 to +127 offset
LD HL,(IX+d)
LD HL',(IX+d)
LD (IY+d),HL    ; store HL at address pointed to
                ; by IY plus -128 to +127 offset
LD HL,(IY+d)
LD HL',(IY+d)
```

the same priority, this introduces interrupt latency while the next routine is waiting for the previous routine to allow more interrupts to take place. If a number of devices have interrupt service routines, and all interrupts are of the same priority, then pending interrupts can not take place until at least the interrupt service routine in progress is finished, or at least until it changes the interrupt priority. As a rule of thumb, Rabbit Semiconductor usually suggests that 100 µs be allowed for interrupt latency on Z180-based controllers. This can result if, for example, there are five active interrupt routines, and each turns off the interrupts for at most 20 µs.

The intention in the Rabbit is that most interrupting devices will use priority 1 level interrupts. Devices that need extremely fast response to interrupts will use priority level 2 or 3 interrupts. Since code that runs at priority level 0 or 1 never disables level 2 and level 3 interrupts, these interrupts will take place within about 20 clocks, the length of the longest instruction or longest sensible sequence of privileged instructions followed by an unprivileged instruction. It is important that the user be careful not to overdisable interrupts in critical code sections. *The processor priority should not be raised above level 1 except in carefully considered situations.*

The effect of the processor priority on interrupts is shown in Table 3-1. The priority of the interrupt is usually established by bits in an I/O control register associated with the hardware that creates the interrupt. The 8-bit interrupt register (IP) holds the processor priority in the least significant 2 bits. When an interrupt takes place, the IP register is shifted left 2 positions and the lower 2 bits are set to equal the priority of the interrupt that just took place. This means that an interrupt service request (ISR) can only be interrupted by an interrupt of higher priority (unless the priority is explicitly set lower by the programmer). The IP register serves as a 4-word stack of 2-bit words to save and restore interrupt priorities. It can be shifted right, restoring the previous priority by a special instruction (**`IPRES`**). Since only the current processor priority and 3 previous priorities can be saved in the interrupt register, instructions are also provided to **`PUSH`** and **`POP IP`** using the regular stack. A new priority can be "pushed" into the IP register with special instructions (**`IPSET 0`**, **`IPSET 1`**, **`IPSET 2`**, **`IPSET 3`**).

Table 3-1. Effect of Processor Priorities on Interrupts

Processor Priority	Effect on Interrupts
0	All interrupts, priority 1,2 and 3 take place after execution of current non privileged instruction.
1	Only interrupts of priority 2 and 3 take place.
2	Only interrupts of priority 3 take place.
3	All interrupt are suppressed (except RST instruction).

```
push af          ;10
push hl
ld hl,(ptr)      ;11
ld a,(hl)        ;5
ioi ld (port),a ; 13 output data
inc hl
ld a,0x0f        ;4
and l            ; see if hl at end of cycle
jr z,step2
ld (ptr),hl
pop hl
pop af
reti
step2:
ld a,(beginptr)
ld l,a
ld (ptr),hl      ;13
pop hl           ;7
pop af
reti
; 103 clocks total
```

4.2 Open-Drain Outputs Used for Key Scan

The parallel port D outputs can be individually programmed to be open drain. This is useful for scanning a switch matrix, as shown in Figure 4-2. A row is driven low, then the columns are scanned for a low input line, which indicates a key is closed. This is repeated for each row. The advantage of using open-drain outputs is that if two keys in the same column are depressed, there will not be a fight between a driver driving the line high and another driver driving it low.

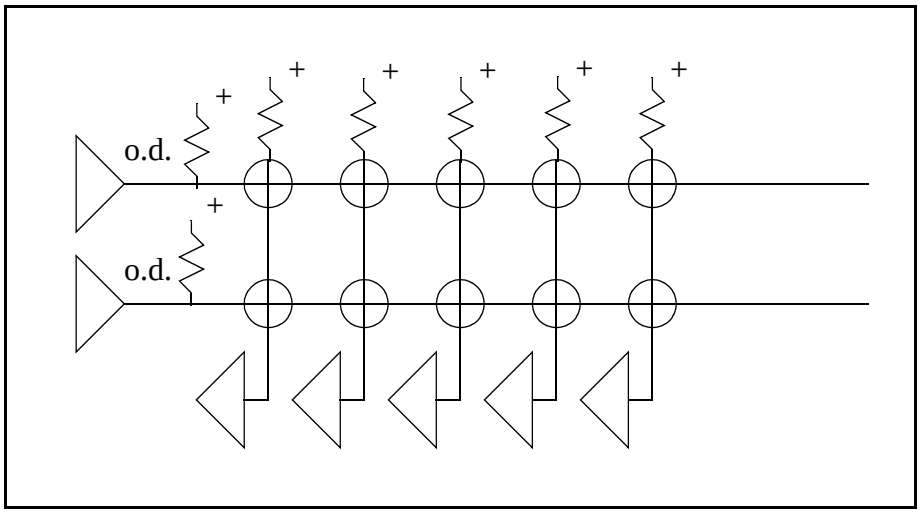

Figure 4-2. Using Open-Drain Outputs for Key Scan

5.5 Description of Pins with Alternate Functions

Table 5-2. Pins With Alternate Functions

Pin Name	Output Function	Input Function	Other Function
STATUS (38)	1. Low on first op code fetch. 2. Low on interrupt acknowledge		Programmable output port high/low
SMODE1 (35)		(SMODE0, SMODE1) Startup boot mode control.	1-bit input after boot complete.
SMODE0 (36)		(SMODE0, SMODE1) Startup boot mode control.	1-bit input after boot complete.
CLK (1)	1. Peripheral clock. 2. Peripheral clock/2.		Programmable output port high/low
/WDTOUT (34)	Outputs 30.5 µs pulse on watchdog timeout (processor is also reset).		Outputs a pulse between 30.5 and 61 µs under program control.
PA7 (88)	SD7	SD7	
PA6 (87)	SD6	SD6	
PA5 (86)	SD5	SD5	
PA4 (85)	SD4	SD4	
PA3 (84)	SD3	SD3	
PA2 (83)	SD2	SD2	
PA1 (82)	SD1	SD1	
PA0 (81)	SD0	SD0	
PB7 (100)	/SLAVEATTN (master needs attention from slave).		
PB5 (98)		SA1 (slave address).	
PB4 (97)		SA0	
PB3 (96)		/SRD (strobe for master to read a slave register).	
PB2 (95)		/SWR (strobe for master to write slave register).	

5.7 I/O Buffer Sourcing and Sinking Limit

Unless otherwise specified, the Rabbit I/O buffers are capable of sourcing and sinking 8 mA of current per pin at full AC switching speed. Full AC switching assumes 22.11 MHz CPU clock and capacitive loading on address and data lines of less than 100 pF per pin. Address pin A0 and Data pin D0 are rated at 16 mA each.

Table 5-6 shows the AC and DC output drive limits of the parallel I/O buffers.

Table 5-6. I/O Buffer Sourcing and Sinking Capability

Pin Name	Output Drive Sourcing*/Sinking† Limits (mA)	
Output Port Name	Full AC Switching SRC/SNK	Maximum‡ DC Output Drive SRC/SNK
PA [7:0]	8/8	12/12
PB [7:6]	8/8	12/12
PC [6, 4, 2, 0]	8/8	12/12
PD [7:4]	8/8	12/12
PD [3:0]**	16/16	25/25
PE [7:0]	8/8	12/12

* The maximum DC sourcing current for I/O buffers between V_{DD} pins is 112 mA.

† The maximum DC sinking current for I/O buffers between V_{SS} pins is 150 mA.

‡ The maximum DC output drive on I/O buffers must be adjusted to take into consideration the current demands made my AC switching outputs, capacitive loading on switching outputs, and switching voltage.

The current ascribed to AC switching is the average current that flows while AC switching is taking place. This can be computed using I = CVf, where f is the number of transitions per second, C is the capacitance switched, and V is the voltage swing. For example, if 12,000,000 transitions per second take place with a 5 V swing driving 100 pF, then I = 6 mA for one pin. The current attributable to all the pins between the power or ground pins must be summed to test the limits, including the current attributable to switching current or DC current.

The current drawn by all switching and non-switching I/O must not exceed the limits specified in the first two footnotes.

** The combined sourcing from Port D [7:0] may need to be adjusted so as not to exceed the 112 mA sourcing limit requirement specified above.

7.5 Output Pins CLK, STATUS, /WDTOUT, /BUFEN

Certain output pins can have alternate assignments as specified in Table 7-4.

Table 7-4. Global Output Control Register (GOCR = 0x0E)

Bit(s)	Value	Description
7:6	00	CLK pin is driven with peripheral clock.
	01	CLK pin is driven with peripheral clock divided by 2.
	10	CLK pin is low.
	11	CLK pin is high.
5:4	00	STATUS pin is active (low) during a first opcode byte fetch.
	01	STATUS pin is active (low) during an interrupt acknowledge.
	10	STATUS pin is low.
	11	STATUS pin is high.
3	1	WDTOUTB pin is low (1 cycle minimum, 2 cycles maximum, of 32 kHz).
	0	WDTOUTB pin follows watchdog function.
2	x	This bit is ignored.
1:0	00	/BUFEN pin is active (low) during external I/O cycles.
	01	/BUFEN pin is active (low) during data memory accesses.
	10	/BUFEN pin is low.
	11	/BUFEN pin is high.

Serial Port A is selected for bootstrap operation as a clocked serial port when SMODE = 10. In this case bit 7 of Parallel Port C is used for the serial data and bit 1 of Parallel Port B is used for the serial clock. Note that the serial clock must be externally supplied for bootstrap operation. This precludes the use of a serial EEPROM for bootstrap operation.

Serial Port A is selected for bootstrap operation as an asynchronous serial port when SMODE = 11. In this case bit 7 of Parallel Port C is used for the serial data and the 32 kHz oscillator is used to provide the serial clock. A dedicated divide circuit allows the use of the 32 kHz signal to provide the timing reference for the 2400 bps asynchronous transfer. Only 2400 bps is supported for bootstrap operation, and the serial data must be eight bits for proper operation.

When the first phase of a bootstrap is performed using Serial Port A, the TXA signal is not needed since the bootstrap is a one-way communication. After the reset ends and the bootstrap mode begins, TXA will be low, reflecting its function as a parallel port output bit that is cleared by the reset. This may be interpreted as a break signal by some serial communication devices. TXA can be forced high by sending the triplet 0x80, 0x50, 0x40, which stores 0x40 in parallel port C. An alternate approach is to send the triplet 0x80, 0x55, 0x40, which will enable the TXA output from bit 6 of parallel port C by writing to the parallel port C function register (0x55).

> **NOTE:** Although the TXA signal is not needed during the first phase of the boot procedure, sending the “byte triplets,” two-way communication is required once the cold loader has been loaded.

The transfer rate in any bootstrap operation must not be too fast for the processor to execute the instruction stream. The Write Empty signal acts as an interlock when using the Slave Port for bootstrap operation, because the next byte should not be written to the Slave Port until the Write Empty signal is active. No such interlock exists for the clocked serial and asynchronous bootstrap operation. In these cases, remember that the processor clock starts out in divide-by-eight mode with four wait states, and limit the transfer rate accordingly. In asynchronous mode at 2400 bps it takes about 4 ms to send each character, so no problem is likely unless the system clock is extremely slow.

8.4 Allocation of Extended Code and Data

The Dynamic C compiler compiles code to root code space or to extended code space. Root code starts in low memory and compiles upward.

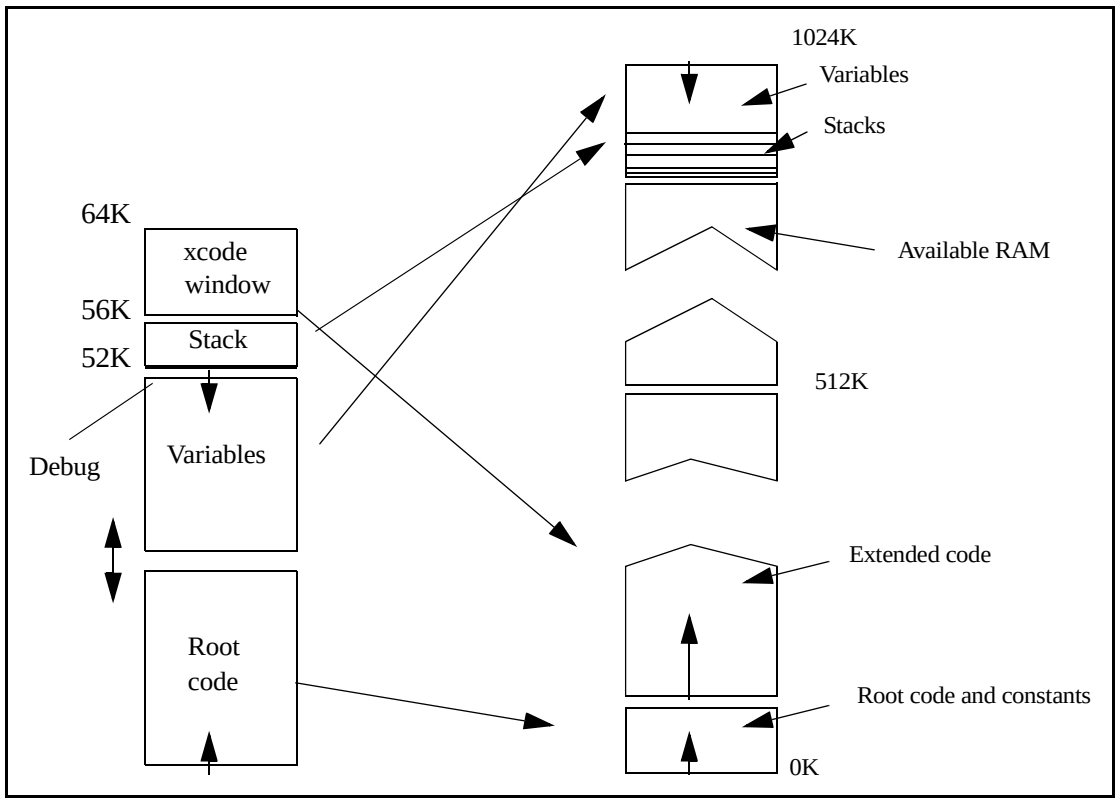

Figure 8-3. Example of Memory Mapping and Memory Usage

Allocation of extended code starts above the root code and data. Allocation normally continues to the end of the flash memory.

Data variables are allocated to RAM working backwards in memory. Allocation normally starts at 52K in the 64K D space and continues. The 52K space must be shared with the root code and data, and is allocated upward from zero.

Dynamic C also supports extended data constants. These are mixed in with the extended code in flash.

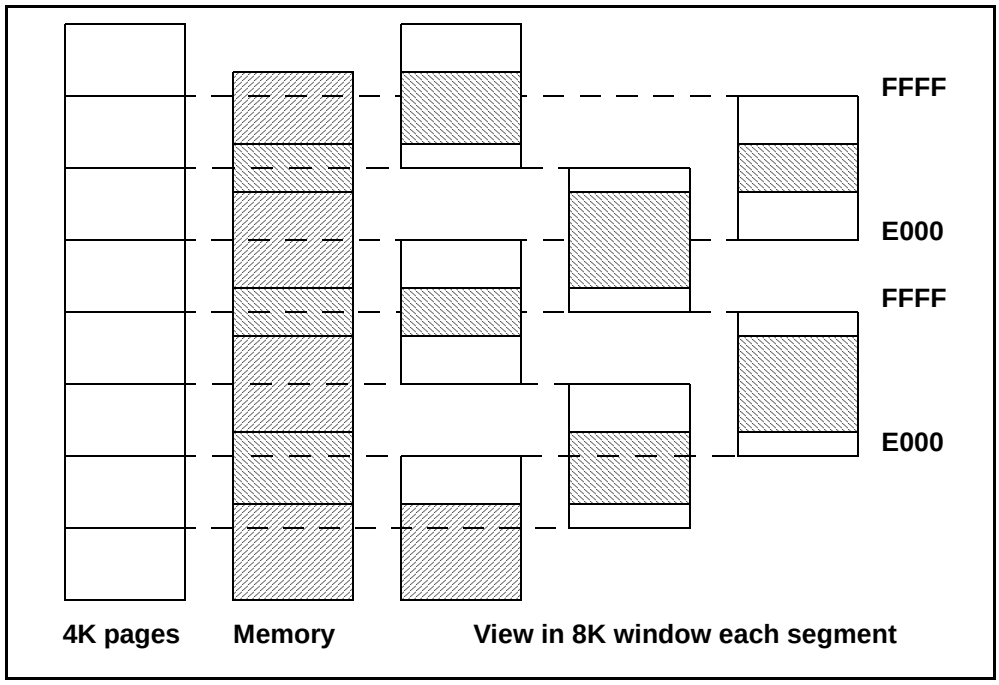

Figure 8-4. Compilation of Code Segments in Extended Memory

9. PARALLEL PORTS

The Rabbit has five 8-bit parallel ports designated A, B, C, D and E. The pins used for the parallel ports are also shared with numerous other functions as shown in Table 5-2. The important properties of the ports are summarized below.

- Port A—Shared with the slave port data interface.
- Port B—Shared with control lines for slave port and clock I/O for clocked serial mode option for serial ports A and B.
- Port C—Shared with serial port serial data I/O.
- Port D—4 bits shared with alternate I/O pins for serial ports A and B. 4 bits not shared. Port D has the ability to configure its outputs as open drain outputs. Port D has output preload registers that can be clocked into the output registers under timer control for pulse generation. Port D bits 0–3 have a higher current drive capability.
- Port E—All bits of Port E can be configured as I/O strobes. 4 bits of port E can be used as external interrupt inputs. One bit of port E is shared with the slave port chip select. Port E has output preload registers that can be clocked into the output registers under timer control for pulse generation.

9.3 Parallel Port C

Parallel port C, shown in Table 9-6, has four inputs and four outputs. The even-numbered ports, PC0, PC2, PC4, and PC6, are outputs. The odd-numbered ports, PC1, PC3, PC5, and PC7, are inputs. When the data register is read, bits 1,3,5,7 return the value of the voltage on the pin. Bits 0,2,4,6 return the value of the signal driving the output buffers. The signal driving the output buffers and the value of the output pin are normally the same. Either the Port C data register is driving these pins or one of the serial port transmit lines is driving the pin. The bits set in the PCFR Parallel Port C Function Register identify whether the data register or the serial port transmit lines were driving the pins.

Table 9-5. Parallel Port C Registers

Register Name	Mnemonic	I/O address	R/W	Reset
Port C Data Register	PCDR	0x50	R/W	x0x0x0x0
Port C Function Register	PCFR	0x55	W	x0x0x0x0

Table 9-6. Parallel Port C Data Register and Function Register

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PCDR (r) adr = 0x050	PC7 in	Echo drive	PC5 in	Echo drive	PC3 in	Echo drive	PC1 in	Echo drive
PCDR (w) adr = 0x050	x	PC6	x	PC4	x	PC2	x	PC0
PCFR (w) adr = 0x055	x	Drive TXA	x	Drive TXB	x	Drive TXC	x	Drive TXD

Parallel port C shares its pins with the four serial ports. The parallel port input pins may also serve as serial port inputs. (Serial ports A and B can alternately use bits 7 and 5 respectively in Port D as inputs, and the source of the serial port inputs for these serial ports depends on the setup of the corresponding serial port control register.) When serving as serial inputs, the data lines can still be read from the parallel port C data register. The parallel port outputs can be selected to be serial port outputs by storing bits in the corresponding positions of the Port C Function register (PCFR). When a parallel port output pin is selected to be a serial port output, the value stored in the data register is ignored. On reset the active (even-numbered) function register bits and data register bits are zeroed. This causes the port to output zeros on the four output bits.

- PDCR—Parallel port D control register. This register is used to control the clocking of the upper and lower nibble of the final output register of the port. On reset, bits 0, 1, 4, and 5 are reset to zero.

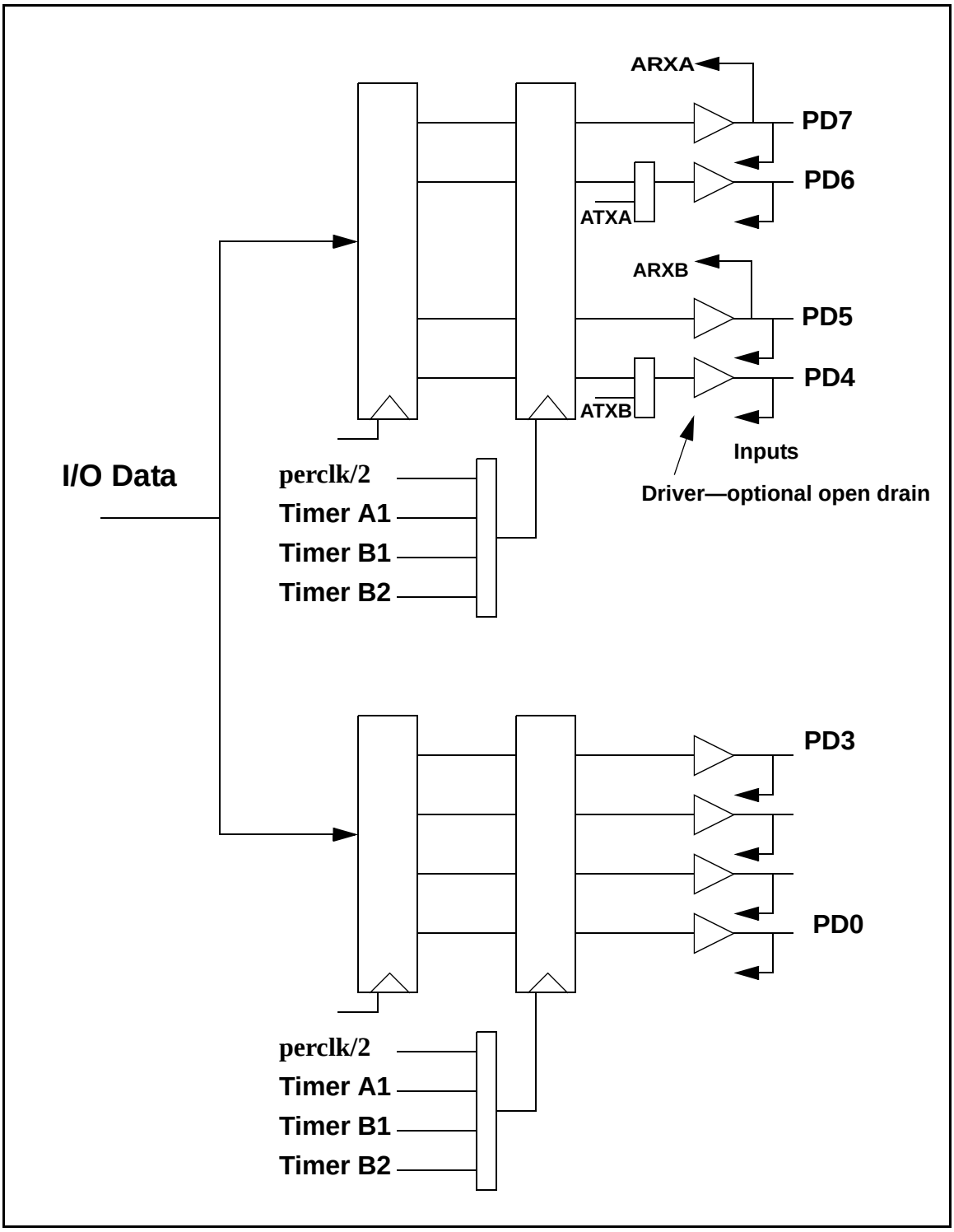

Figure 9-1. Parallel Port D Block Diagram

Table 9-10. Parallel Port E Registers

Register Name	Mnemonic	I/O address	R/W	Reset
Port E Data Register	PEDR	0x70	R/W	xxxxxxxx
Port E Control Register	PECR	0x74	W	xx00xx00
Port E Function Register	PEFR	0x75	W	00000000
Port E Data Direction Register	PEDDR	0x77	W	00000000
Port E Bit 0 Register	PEB0R	0x78	W	xxxxxxxx
Port E Bit 1 Register	PEB1R	0x79	W	xxxxxxxx
Port E Bit 2 Register	PEB2R	0x7A	W	xxxxxxxx
Port E Bit 3 Register	PEB3R	0x7B	W	xxxxxxxx
Port E Bit 4 Register	PEB4R	0x7C	W	xxxxxxxx
Port E Bit 5 Register	PEB5R	0x7D	W	xxxxxxxx
Port E Bit 6 Register	PEB6R	0x7E	W	xxxxxxxx
Port E Bit 7 Register	PEB7R	0x7F	W	xxxxxxxx

The following registers are described in Table 9-11 and in Table 9-12.

- PEDR—Port E data register. Reads value at pins. Writes to port E preload register.
- PEDDR—Port E data direction register. Set to "1" to make corresponding pin an output. This register is zeroed on reset.
- PEFR—Port E function register. Set bit to "1" to make corresponding output an I/O strobe. The nature of the I/O strobe is controlled by the I/O bank control registers (IBxCR). The data direction must be set to output for the I/O strobe to work.
- PEBxR—These are individual registers to set individual output bits on or off.
- PECR—Parallel port E control register. This register is used to control the clocking of the upper and lower nibble of the final output register of the port. On reset, bits 0, 1, 4, and 5 are reset to zero.

15.1 Memory Access and I/O Read/Write Times

The memory access time requirements are listed in Table 15-2. It is important that wait states should not be used for any memory that holds code that is being executed. Memory wait states are only intended for use with data accesses. For code memory the clock should be matched to the memory requirements, or one of the clock dividers should be enabled to accommodate slow memory. As a rough guide, each data memory wait state in main RAM that is introduced will reduce the average compute performance by approximately 8%. The data memory read access is slowed by 50% for 1 wait state and is slowed by 100% for 2 wait states. However, since only a small proportion of accesses are data accesses rather than code accesses or instruction fetch cycles, the overall affect on performance is slight. If data memory wait states are introduced, it is important to use the macros specified in the BIOS so that the compiler will be aware of the wait states.

Generally, the maximum operating speed is proportional to the power supply voltage. The operating current is proportional to the voltage, and so the operating power is proportional to the square of the voltage. The operating power is also proportional to the clock speed. Higher temperatures reduce the maximum operating speed by approximately 1% for each 5°C. In addition, higher operating speeds increase the die temperature because of the heat generated and therefore slightly compound the adverse effects of higher temperature.

The MMIDR register shown in Table B-9 is used to enable and configure separate I & D space support in addition to the /CS1 enable option used to improve the access time of battery-backable SRAM.

NOTE: Bits [7:5] and [3:0] were always written with zero in the original Rabbit 2000 chip.

Table B-9. MMU Instruction/Data Register (MMIDR = 0x010)

MMU Instruction/Data Register (MMIDR) (Address = 0x10)		
Bit(s)	**Value**	**Description**
7:6	00	These bits are ignored and always return zeros when read.
5	0	Enable A16 and A19 inversion independent of instruction/data.
	1	Enable A16 and A19 inversion (controlled by bits 0-3) for data accesses only. This enables the instruction/data split. This is separate I and D space.
4	0	Normal /CS1 operation.
	1	Force /CS1 always active. This will not cause any conflicts as long as the memory using /CS1 does not also share an Output Enable or Write Enable with another memory.
3	0	Normal operation.
	1	For a DATASEG access, invert A19 before MBxCR (bank select) decision.
2	0	Normal operation.
	1	For a DATASEG access: invert A16
1	0	Normal operation.
	1	For root access, invert A19 before MBxCR (bank select) decision.
0	0	Normal operation.
	1	For root access, invert A16

B.2.10 DDCB/FDCB Instruction Page and Wait State Bug Fixes

Four-byte instructions starting with DD-CB or FD-CB didn't work when attempted with wait states.

The fetch of the byte immediately following the instruction did not have the correct number of wait states inserted for the following instructions only when using wait states. Rather than the programmed number of wait states, the fetch was short by one wait state.

```
DJNZ (branch not taken only)
JR cc (branch not taken only)
JP cc (branch not taken only)
```

A similar thing happens for the block move instructions. In these cases, the read cycle is short by one wait state.

```
LDDR
LDIR
```

For the multiply instruction, the fetch of the first byte after the MUL instruction had no wait states, independent of the number programmed.

These problems were corrected in revisions A–C of the Rabbit 2000.

New Bug with LDIR/LDDR

A new LDIR/LDDR bug was discovered in September, 2002. The problem has to do with wait states and the block move operations. With this problem, the first iteration of **LDIR/LDDR** uses the correct number of wait states for both the read and the write. However, all subsequent iterations use the number of waits programmed for the memory located at the write address for both the read and write cycles. This becomes a problem when moving a block of data from a slow memory device requiring wait states to a fast memory device requiring no wait states. With respect to external I/O operations, the **LDIR** or **LDDR** performs reads with zero wait states independent of the waits programmed for the I/O for all but the first iteration. The first iteration is correct. This bug is automatically corrected by Dynamic C.

B.2.11 LDIR/LDDR Instruction/Data Split Bug Fix

The bug with **LDIR**/**LDDR** and separate I & D space discovered in the Rabbit 2000A had to do with the way the memory control unit treated the move from and the move to addresses of the block move operation. With the instruction/data split enabled, data access in the ROOT and/or DATASEG regions would result in addresses A16 and/or A19 being inverted, depending on how the MMIDR was configured. This would allow the data space to be moved up or down by 64K or 512 K.

With this problem, the first iteration of **LDIR**/**LDDR** resulted in the correct address inversion for data accesses in the ROOT and/or DATASEG regions. However, all subsequent iterations took place in the code region (without any address inversion).

This problem was fixed in revisions B and C of the Rabbit 2000.

B.2.12 Clock Spectrum Spreader Module

This is a feature introduced on the Rabbit 3000 and migrated to revisions B and C of the Rabbit 2000. The clock spectrum spreader and early memory output enable are turned on by default for the Rabbit 2000C in Dynamic C version 7.32 and higher. The spectrum spreader is very powerful for reducing EMI because it will reduce all sources of EMI above 100 MHz that are related to the clock by about 15 dB. This is a very large reduction since it is common to struggle to reduce EMI by 5 dB in order to pass government tests.

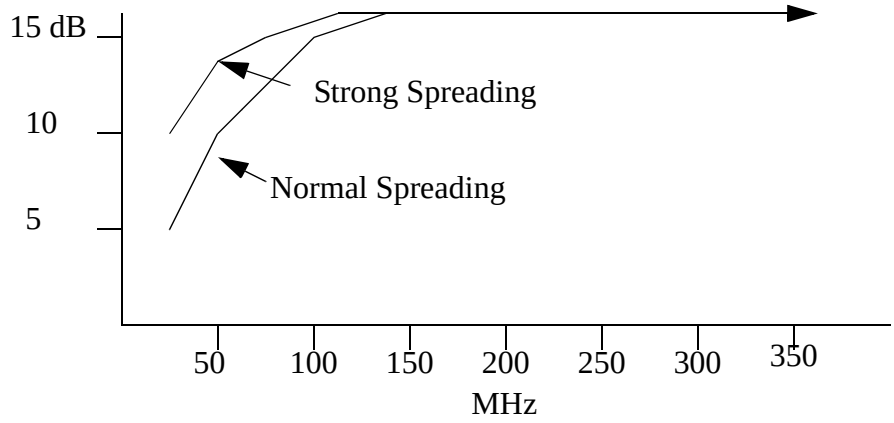

Figure B-2. Peak Spectral Amplitude Reduction from Spectrum Spreader

The spectrum spreader modulates the clock so as to spread out the spectrum of the clock and its harmonics. Since the government tests use a 120 kHz bandwidth to measure EMI, spreading the energy of a given harmonic over a wider bandwidth will decrease the amount of EMI measured for a given harmonic. The spectrum spreader not only reduces the EMI measured in government tests, but it will also often reduce the interference created for radio and television reception.

The spectrum spreader has three settings under software control: off, normal spreading, and strong spreading.

Two registers control the clock spectrum spreader. These registers must be loaded in a specific manner with proper time delays. GCM0R is only read by the spectrum spreader at the moment when the spectrum spreader is enabled by storing 0x080 in GCM1R. If GCM1R is cleared (when disabling the spectrum spreader), there is up to a 500-clock delay before the spectrum spreader is actually disabled. The proper procedure is to clear GCM1R, wait for 500 clocks, set GCM0R, and then enable the spreader by storing 0x080 in GCM1R.

unless the station is very weak, in which case the interference will be seen as noise distributed over the screen.

A more important change in timing is that the memory access time will be shortened. The shortening with the clock doubler enabled and zero wait states is a maximum of 6 ns in the normal mode and 9 ns in the strong mode. Only one of the 2 clocks in a memory cycle will be shortened.

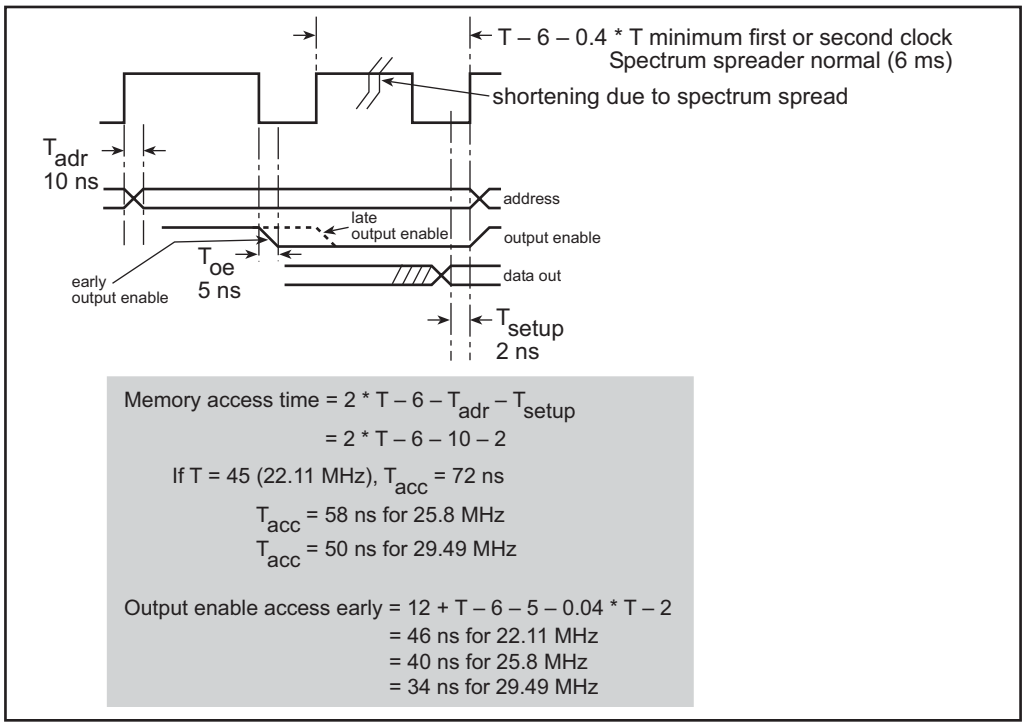

Figure B-3. Clock Spectrum Spreader Example

If the clock doubler is not enabled, then the maximum shortening will be 9 ns in the normal mode and 18 ns in the strong mode. Figure B-3 assumes that the combined address out and data setup in is 12 ns. The time from clock to output enable is assumed to be 5 ns. The maximum asymmetry of the clock is assumed to be 52-48%, which shortens one clock by 4% and lengthens the other by 4% if the clock is doubled.

Early output enable is enabled by default on the Rabbit 2000C, but may be disabled. The clock low time is controlled by the clock doubler control register, and is assumed to be a minimum of 14 ns in the above example. Also the maximum clock speed from the example with the spreader enabled and 55 ns memory with 25 ns output enable is 25.8 MHz. At 29.49 MHz the memory access must be 50 ns, and the spectrum spreader must be turned off, or a wait state must be added. Operation with a doubled clock and the spreader enabled at 29.49 MHz is only allowed for T $< 70°C$ and V > 4.75 V since the instantaneous clock frequency bursts to 38.5 MHz when the spectrum spreader and clock asymmetry together produce maximum shortening of a clock cycle.