ZARC HARDWARE USERS' GUIDE

Written by Merlin Skinner-Oakes, 4/8/2020

Last updated on 8/5/2022

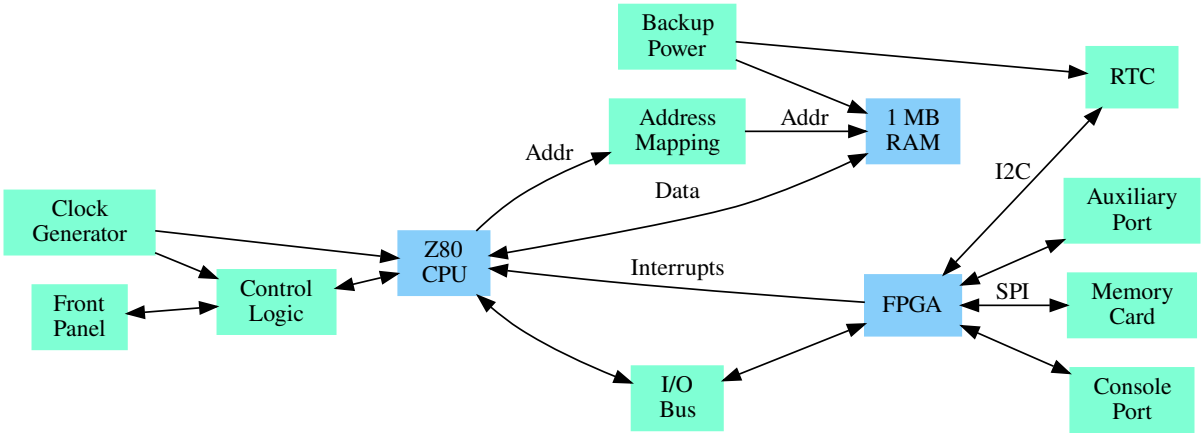
1 DESCRIPTION

The Z80 Anachronistic Retro Computer (ZARC) is a Zilog Z80-based 8-bit computer. Some features were inspired by a number of other machines, including the DEC PDP-8, BBC Microcomputer and Sinclair Spectrum. The most significant features are:

- Zilog Z80 CPU running at 8 MHz.
- Physical front panel with LEDs ("blinkenlights") and switches.
- 1 MB of battery-backed static RAM with write protection. This simulates core memory, allowing use of techniques previously used on such machines.
- A flexible memory mapping system is used to map the RAM into the 16-bit space directly addressable by the CPU.
- The core computer is a mixture of CMOS and TTL with no programmable logic.
- I/O is implemented in an Altera Cyclone II FPGA.

I/O includes:

- Two RS-232 ports.
- Real time clock (RTC).
- MMC memory card interface.

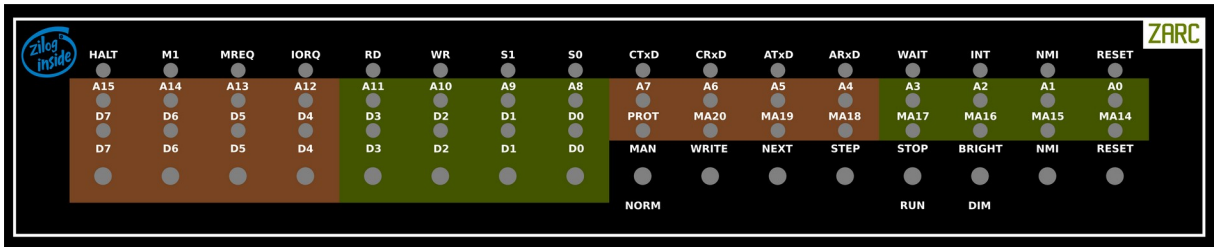


Hardware Block Diagram

The following sections describe these features in detail.

2 FRONT PANEL

The front panel allows control and monitoring of the CPU state. It may also be used to access memory and I/O. The memory access function is critical when starting the machine from a completely uninitialised state as there is no boot ROM from which to boot the machine. It is necessary to enter a short bootstrap programme from the front panel to begin normal operations.



Front Panel

Most front panel indicators show the state of the associated CPU pin. The exceptions are described in the following table:

Indicator	Function
S1	The memory card (MMC) interface is busy
S0	Memory mapping is enabled
CTxD	The serial port 1 transmitter is busy or the associated FIFO contains data to transmit.
CRxD	The serial port 1 receiver is busy or the associated FIFO contains data tor the CPU to read.
ATxD	The serial port 2 transmitter is busy or the associated FIFO contains data to transmit.
ARxD	The serial port 2 receiver is busy or the associated FIFO contains data for the CPU to read.
PROT	The currently addressed memory location is within a bank that is write protected.
SUPER (MA20)	The currently addressed memory location is within a bank that permits I/O instructions.
MA14 to MA19	The currently addressed memory location is within a bank that has this page value associated with it. This is also the memory address bits 14 to 19.

Switches functions are:

Switch	Action	Function
D0 to D7	Toggle	Data input. “0” = down, “1” = up.
Man / Norm	Toggle	When the CPU is stopped, selecting “Man” overrides data from memory or I/O with data from the D0 – D7 switches. “Norm” allows data to come from memory or I/O unaffected.
Write	Momentary	When the CPU is stopped, pressing write will store the data on the D0 – D7 switches to the currently addressed location.
Next	Momentary	Similar to Step, but asserts a NOP (0x00) on the data bus. This is inhibited for all but M1 (instruction fetch) cycles.
Step	Momentary	When the CPU is stopped, pressing step will advance the machine by one machine cycle.
Stop / Run	Toggle	“Run” enables normal operation of the CPU. “Stop” stops the CPU after the current cycle and enables some panel switch functions.
Bright / Dim	Toggle	Selects bright or dim modes for the front panel LEDs.
NMI	Momentary	Generates a non-maskable interrupt (NMI).
Reset	Momentary	

Operation of the panel is somewhat unusual in that addresses are always generated by the CPU itself. To write to an arbitrary memory address it is first necessary to make the CPU generate the required address, for example by jumping to that location. This is the main purpose of the "manual" mode as it is possible to feed the CPU a jump instruction (0xc3) followed by the required address.

Front panel memory write operations override the write protection mechanism, so it is not necessary to manipulate the mapping registers in order to modify the default page 0x3f memory. Mapping is necessary to address other pages, should this be required.

2.1 Normal Start

To start the machine normally, simply set the *Run / Stop* switch to *Run* and apply power to the computer. This assumes there is valid code in page 0x3f, which is the page selected when memory mapping is disabled.

The computer may be operated with the front panel disconnected. This results in a normal start.

2.2 Read Memory From Location 0x0000

Ensure *Run / Stop* is set to *Stop*.

Press *Reset*.

The indicators show an instruction fetch (M1) cycle from address 0x0000. The associated memory read data is shown on *D0 – D7*.

2.3 Advance (Increment) the Memory Address

To see the next location, it is necessary to supply a NOP (0x00) instruction to the CPU.

Ensure *Man / Norm* is set to *Man*.

Press *Next*.

The address displayed will increment, and the data changes to reflect the data at the new location. Repeat *Step* to advance through memory as required.

2.4 Set an Arbitrary Address

Ensure *Run / Stop* is set to *Stop*.

Ensure *Man / Norm* is set to *Man*.

Set D0 – D7 switches to a JMP instruction (0xc3). This is, from left to right, up, up, down, down, down, down, up, up.

Press *Reset* (optional, but it is an easy method to ensure that the CPU is expecting an instruction [an M1 cycle]).

Press *Step*.

Set Set D0 – D7 switches to the low byte of the desired address.

Press *Step*.

Set Set D0 – D7 switches to the high byte of the desired address.

Press *Step*.

Verify that the address bus indicators show the desired address.

2.5 Write Data To Memory

The write will occur at the current memory location.

Set D0 – D7 switches to the desired data.

Press *Write*.

Front panel writes are not inhibited by the memory write protection mechanism (PROT bit).

2.6 Single Step Through a Programme

Ensure *Man / Norm* is set to *Norm*, so the CPU receives data from memory and I/O rather than the D0 – D7 switches.

Press *Step* to advance the CPU by one machine cycle.

2.7 Begin Execution At the Current Location

Set *Run / Stop* to *Run*.

3 MEMORY SYSTEM

ZARC is equipped with a full 1 MB of battery-back CMOS static RAM. This is accessible by the CPU running at 8 MHz with no wait states.

3.1 Memory Banks and Mapping

The CPU's 16-bit space is divided into four equal 16 KB banks. Each bank can be mapped to any 16 KB page in the RAM. The banks are arranged as follows:

Bank Number	***CPU Address Range***
0	0x0000 to 0x3fff
1	0x4000 to 0x7fff
2	0x8000 to 0xbfff
3	0xc000 to 0xffff

In addition, each bank can select write protection (read-only) and "supervisor" mode. Banks with the associated supervisor bit set may be used to execute code that addresses I/O. These features are intended to prevent inadvertent loss of data rather than to implement robust protection against malicious code. In particular, the supervisor mode prevents programs not designed for ZARC (such as those running under CP/M) from addressing ZARC I/O as this would almost certainly result in undesirable behaviour.

The memory mapping system can be turned on and off under software control. It is disabled by system reset. When disabled, every bank addresses page 0x3f (the highest page). Write protection and supervisor are enabled. As the Z80 begins executing code at location 0x0000 on reset, code in page 0x3f is automatically started. Such code must enable memory mapping and select at least one bank without write protection in order to write to memory.

3.2 NMI Processing

ZARC uses Non-Maskable Interrupts (NMIs) for a variety of purposes including tracing and write violation reporting. This presents a particular problem in cases where the handling code is paged out. For example, in a CP/M environment the NMI location (0x0066) is occupied by the CP/M system. The monitor's NMI handler is not accessible because it is paged out. Without further measures, generation of an NMI for any reason would almost certainly result in a crash.

To permit use of NMI in such circumstances, a special memory mapping mode is enabled when NMI is serviced. In this mode, the memory mapping system is disabled except for write operations. As a result, handling code in page 0x3f is executed and writes proceed to the previously selected bank. In practice, these writes will be the PC and some register data pushed onto the stack. Once this is performed, software triggers NMCLR to restore normal operation. See section 4.2 for details of this bit. The first few instructions of the monitor's NMI service routine are:

```
; Jumps here from the hardware NMI vector (0x0066).
; Save some registers (writes to to the user stack).
F5                push af
C5                push bc
;
; PC, AF and BC are on the user stack.
; Save memory mapping context before we change anything.
DB04              in IOA_MMAP_0
4F                ld c, a
DB05              in IOA_MMAP_1
47                ld b, a
; Set bank 0 (0x0000 to 0x3fff) as read-only with I/O access.
3EFF              ld a, MMAP_MON_B0_PAGE+MMAP_PROT_MASK+MMAP_SUPER_MASK
D304              out (IOA_MMAP_0), a
; Bank 1 (0x4000 to 0x7fff) writeable.
3E3E              ld a, MMAP_MON_B1_PAGE
D305              out (IOA_MMAP_1), a
; Enable normal memory mapping and disable mapping for writes only. From
; now on, reads and writes behave normally.
3E06              ld a, SYSCTRL_MMAP_MASK + SYSCTRL_NMCLR_MASK
D302              out (IOA_CONTROL), a
```

The above mechanism will fail for prefixed instructions (0xCB, 0xDD, 0xED or 0xFD). This is because the second byte of the instruction (also an M1 cycle) is indistinguishable from the NMI acknowledge cycle. This fools the logic into disabling mapped mode before it should, therefore causing reads associated with a traced instruction to be fetched from page 0x3f instead of the desired page. To prevent this, the monitor sets TRPRE in the control register appropriately when tracing such instructions. Note that some instructions can have two prefixes, such as:

```
DD CB 12 06           RLC   (ix+ 0x12)
```

Other sources of NMI such as write protect violations do not allow the possibility of setting TRPRE appropriately. In these cases, the last instruction will be fetched from page 0x3f instead of the desired page. In these cases, however, the priority is to prevent the system from continuing after an error. This undesirable behaviour only affects prefixed instructions, which are relatively uncommon in normal code.

Normally, memory mapping is enabled in user mode. The mechanism described cannot work in unmapped mode as in this case there is no writeable stack on which to store the PC and other registers. In this case, the stacked parameter values are lost.

3.3 Switch from Monitor to User Code

In the case of non-recoverable errors such as NMI, no return from the NMI is appropriate. However, monitor commands such as GO and TRACE require that the saved context be restored. Setting MAPARM enables memory mapping after a predetermined number of machine cycles. This allows execution of the monitor's code to be completed before switching to the user code with the appropriate mapping state. See section 4.2 for details of MAPARM. The last few instructions of the context switch are:

```
F1                pop af                        ; Restore AF
; A will be overwritten. Code between now and exit must not disturb the
; flags!
ED7B6B01          ld sp, (mstate + MST_SP)     ; Load user stack pointer
3A6000            ld a, (run_ctrl)             ; Set pre-calculated value
D302              out (IOA_CONTROL), a         ; Start hardware timers if
selected
; Memory mapping is now off, so we have just read-only access to page 0x3f.
79                ld a, c
D304              out IOA_MMAP_0
78                ld a, b
D305              out IOA_MMAP_1
; Memory mapping is now turned off so we only have page 0x3f. Set user
; banks 0 and 1. The following code will be modified before execution.
010000            ld bc, 0
3E00              ld a, 0
FB                ei                       ; Set to EI or NOP as required
C30000            jp 0                     ; Jump to desired PC
```

The final instructions are modified as required before execution. This is required as the machine state variables are no longer accessible once mapping is disabled. The MAPARM delay is such that memory mapping is enabled once the jump to the destination code is complete.

4 I/O SPACE

I/O is implemented on a separate plug-in board. The FPGA implements the I/O registers described in the following sub-sections.

4.1 FPGA Revision

Two bytes are allocated to report the FPGA revision to the software.

I/O Addr	Symbol	Mode	Description
0x00	IOA_FPGA_MajRev	Read only	FPGA major revision number byte
0x01	IOA_FPGA_MajRev	Read only	FPGA minor revision number byte

4.2 System Control

This register contains the memory mapping enable bit.

I/O	Symbol	Mode	Description

Addr			
0x02	IOA_Control	Read / write	System control register

The following table describes the bit mapping.

7	6	5	4	3	2	1	0
Reserved	R/W		R/O	W/O	W/O	R/W	W/O
	TRPRE		MMAPN	MAPARM	NMCLR	MMAP	TRARM

TRARM – set to generate an NMI after a delay. This is used by the monitor's TRACE command to allow execution of a single instruction. Cleared on reset.

MMAP – set if memory mapping is enabled. Changing this bit has an effect starting with the next instruction. Cleared on reset.

NMCLR – "NMI Mapping Clear". Following an NMI, memory mapping is enabled for write cycles only. Write "1" to this bit to revert to the memory mapping mode defined by the MMAP bit. See section 3.2.

MAPARM – "MMAP Arm". Write "1" to this bit to enable memory mapping after a delay. This is used for monitor to user code switches. See section 3.2. Cleared on reset.

MMAPN – Indicates the state of MMAP when an NMI occurred.

TRPRE – "Trace Prefix". Set to the number of prefix bytes (0xCB, 0xDD, 0xED or 0xFD) for the traced instruction. In most cases, this will be zero, but it can be up to two, such as 0xDD 0xCB. This affects the memory mapping behaviour during NMI servicing. Cleared on reset.

The MMAP bit is indicated on the front panel by LED "S0". When lit, memory mapping is enabled.

4.3 Front Panel Data

This register returns state of the front panel data switches.

I/O Addr	Symbol	Mode	Description
0x03	IOA_FP_SW	Read only	Returns state of the front panel data switches.

4.4 Memory Mapping Bank Registers

These registers define the characteristics of each 16 KB bank directly addressable by the CPU.

I/O Addr	Symbol	Mode	Description
0x04	IOA_MMAP_0	Read / write	Page, supervisor and protection for bank 0 (0x0000 to 0x3fff)
0x05	IOA_MMAP_1	Read / write	Page, supervisor and protection for bank 0 (0x4000 to 0x7fff)
0x06	IOA_MMAP_2	Read / write	Page, supervisor and protection for bank 0 (0x8000 to 0xbfff)
0x07	IOA_MMAP_3	Read / write	Page, supervisor and protection for bank 0 (0xc000 to 0xffff)

The following table describes the bit mapping for all bank registers.

7	6	5	4	3	2	1	0
PROT	SUPER	PAGE					

PROT – “1” enables write protection for the corresponding bank.

SUPER – “1” enables I/O operations for the corresponding bank.

PAGE – the page value defines the upper 6 bits of the RAM address for the corresponding bank.

An attempt to write to a bank with the PROT bit set will result in the write being ignored and an NMI generated.

An attempt to execute an I/O instruction from a bank with the SUPER bit reset will result in the I/O cycle being ignored and an NMI generated.

The physical hardware mapping registers are not readable or cleared on reset. In order that software can read these registers, a shadow copy is maintained in the FPGA. It is the shadow register value that is returned when the bank registers are read. In general, this implementation is transparent to software. It does mean that the value read is not necessarily the value in the hardware register until an initial write operation is performed.

4.5 Interrupt Control

There are a number of sources of maskable interrupt to the CPU.

I/O Addr	Symbol	Mode	Description
0x08	IOA_INT_EN	Read / write	Interrupt enable bits
0x0a	IOA_INT_PEND	Read / write	Interrupt pending bits. Read only for all but TICK (see below).

Note: addresses 0x09 and 0x0b reserved for use if more than eight interrupt sources are required.

The following table describes the bit mapping for both the IOA_INT_EN and IOA_INT_PEND registers.

7	6	5	4	3	2	1	0
Reserved			TICK	AUXRX	AUXTX	CONRX	CONTX

CONTX – serial port 1 transmitter buffer is half full or less.
CONRX – serial port 1 receiver has data.
AUXTX – serial port 2 transmitter buffer is half full or less.
AUXRX - serial port 2 receiver has data.
TICK- 10 Hz timekeeping interrupt.

If a bit is set in the IOA_INT_EN (enables) register, an interrupt request will be sent to the CPU if the corresponding IOA_INT_PEND (pending) register bit is set.

A bit will be set in the pending register if the corresponding reason is true. This happens regardless of the enable state.

The TICK bit (only) may be cleared by writing a “1” to this bit. This bit is also cleared automatically when the TICK interrupt is serviced.

Z80 interrupt mode 2 (“IM 2”) is anticipated. The required fixed vector is supplied by the FPGA logic when the interrupt is serviced. The vector supplied is 0xe0 + 2 * interrupt number (0 to 7). The interrupt numbers are identical to the bit numbers used the interrupt controller registers. For example, the tick interrupt is number 4 (hence bit 4 in the enable and pending registers). The vector is therefore 0xe0 + 2 * 4 = 0xe8. If the CPU “I” register were set to 0xff, this would result in the CPU jumping to the location stored in the vector at 0xffe8. This method allows separate service routines for each interrupt, rather than having the CPU poll to find which event caused the interrupt.

4.6 Non-Maskable Interrupt (NMI) Control

There are a number of sources of non-maskable interrupt to the CPU. These represent urgent conditions and cannot be disabled. The IOA_NMI_REASONS register allows software to determine which event caused the interrupt.

I/O Addr	Symbol	Mode	Description
0x0c	IOA_NMI_REASONS	Read / write	NMI reasons flags. A bit will be set when the corresponding even occurs. Write “1” to a bit to clear it.

The following table describes the bit mapping.

7	6	5	4	3	2	1	0
Reserved				TRACE	SUPV	PROTV	SWCH

SWCH – the front panel NMI switch was pressed.
PROTV – an attempted write to protected memory was blocked.
SUPV – an attempted I/O operation from a bank with SUPER reset was blocked.
TRACE – a trace interrupt has occurred.

The TRACE function is intended for use by a debugger. It will cause an NMI 15 bus cycles after being armed by a write to IOA_Control with the TRARM bit set.

4.7 Serial (RS-232) I/O

There are two identical serial ports, referred to console and auxiliary. Each contains a transmitter and receiver with associated 32-byte FIFOs.

I/O Addr	Symbol	Mode	Description
0x10	IOA_SER1_CSR	Read only	Serial port 1 control / status
0x11	IOA_SER1_DATA	Read / write	Serial port 1 data
0x12	IOA_SER2_CSR	Read only	Serial port 2 control / status
0x13	IOA_SER2_DATA	Read / write	Serial port 2 data

The following table describes the bit mapping for the control and status registers (CSRs).

7	6	5	4	3	2	1	0
Res	TxF	TxH	TxE	FE	RxF	RxH	RxE

RxE – receiver buffer is empty.
RxH – receiver buffer is at least half full.
RxF – receiver buffer is full.
FE – framing error in last character.
TxE – transmitter buffer is empty.
TxH – transmitter buffer is at least half full.
TxF – transmitter buffer is full.
Res – reserved.

Notes:

The serial port configuration (Baud rate, parity etc.) is configured in the FPGA and is not adjustable under software control.

Reads from the data register will be extended indefinitely by means of the hardware WAIT mechanism if the receiver FIFO is empty.

Writes to the data register will be extended indefinitely by means of the hardware WAIT mechanism if the transmitter FIFO is full.

These unusual features makes it possible to write very short programs that send and receive data using the serial ports. There is no need to poll the CSR to determine whether or not there is data to read. This is is useful when booting the machine from cold using a bootstrap entered using the front panel. In normal operation, the CSR registers or interrupt mechanisms may be used to avoid extended wait conditions.

4.8 I^2C (Inter-Integrated Circuit) Master Interface

The I^2C interface is used to communicate with the Real Time Clock (RTC) (DS1672).

I/O Addr	Symbol	Mode	Description
0x18	IOA_I2C_CSR	Read / write	RTC (DS1672) I^2C interface control and status
0x19	IOA_I2C_DATA	Read / write	I^2C RTC (DS1672) data

The following table describes the bit mapping for the control and status register (CSR).

7	6	5	4	3	2	1	0
BUSY	Reserved				STOP	START	ACKn

ACKn – (Read / write) acknowledge from receiving device.
START – (Write only) trigger start sequence.
STOP – (Write only) trigger stop sequence.
BUSY – (Read only) I^2C transceiver is busy.

Notes:

I^2C is an open-collector (or open-drain) bus, so transmitting a logic “1” is identical to turning off the transmitter.

Writing to the data register triggers a byte transfer. This consists of 8 data bits plus the acknowledge bit. To receive a byte, write 0xff to the data register.

When read, the ACKn bit returns the acknowledge bit received during the last transfer.

Writing to the ACKn bit sets the acknowledge state to be transmitted. This should be set to “1” to receive the acknowledge from a slave device.

4.9 SPI (Serial Peripheral Interface) Master Interface

The SPI interface is used to communicate with the memory card. The register behaviour is similar to that for the I^2C interface.

I/O Addr	Symbol	Mode	Description
0x20	IOA_SPI_CSR	Read / write	MMC card SPI interface control and status
0x21	IOA_SPI_DATA	Read / write	MMC card SPI data

The following table describes the bit mapping for the control and status register (CSR).

7	6	5	4	3	2	1	0
BUSY	Reserved					SS	FAST

FAST – (Read / write) SPI speed select.
SS – (Read / write) memory card slave select (set to enable card).
BUSY – (Read only) SPI transceiver is busy.

Notes:

The SPI interface runs at one of two speeds. When FAST is zero, the interface runs at 400 kHz for memory card identification mode. The low frequency is required for Multi Media Card (MMC) compatibility. When FAST is high, the SPI clock runs at half of the CPU clock speed. For ZARC, this is 4 MHz, and is usable for some memory cards or other peripherals.

4.10 CRC16 / XMODEM and MultiMediaCard CRC16

The CRC algorithm used for XMODEM and MultiMediaCard data transfers is identical. The polynomial is $X^{16}+ X^{12}+ X^{5}+ 1$ (0x1021). The initial value is zero for XMODEM and MultiMediaCard.

These registers provide an interface to a hardware CRC generator suitable for either of these applications.

I/O Addr	Symbol	Mode	Description
0x24	IOA_CRC_0	Read / write	CRC register (low byte)
0x25	IOA_CRC_1	Read / write	CRC register (high byte)
0x26	IOA_CRC_DATA	Write only	Data input

The CRC register must be initialised by writing to CRC_0 and CRC_1. Bytes to be included in the CRC are then simply written in turn to CRC_DATA. The hardware requires approximately eleven clock cycles to process each byte. If an attempt is made to supply another byte or read the result while the CRC hardware is busy, the CPU is blocked using the WAIT mechanism until processing is complete. In practise, it is unlikely that software will be fast enough to incur this penalty.