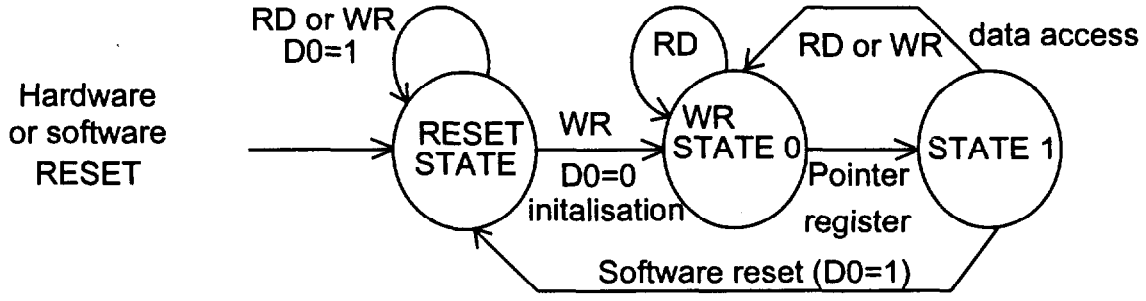
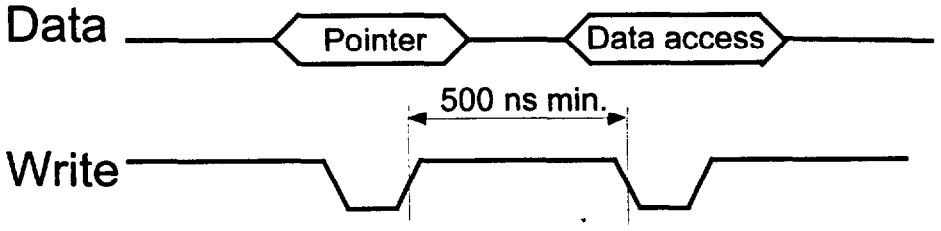
5.6 Z8536 registers access

Two cycles are necessary to address an internal register, as the internal CIO registers are accessed through an internal pointer register. The first cycle is the write pointer operation, into the control register (CR0) which select the internal register.
The next cycle is the data access operation.

The following state machine shows the internal register operation.



The CIO is on the reset state, after hardware or software reset. The software reset is accomplished by writing D0=1 in CR0 register. On the CIO controller, the hardware reset is accomplished when the reset signal is asserted on the IP logic interface. To leave this state, an initialisation cycle is necessary: write D0=0 in the control register, otherwise during a read cycle the CIO device returns always D0=1. The state 0 waits for the pointer writing (in the control register) for the internal register selection. After the data access (read or write cycle), the state machine returns to the state 0, waiting for the next pointer writing.
A **minimum time of 500 ns** is required between the pointer writing and the data access operation.



All internal CIO registers, including the internal data register, are accessed through this method. The pointer register is automatically cleared after each following read or write cycle.
A direct register access, without pointer writing cycle, is available for the data register with the PADR to PDDR registers (see I/O space section).

The following table shows the internal CIO registers.

CIO Internal addr.	Cycle	Registers
Main control registers		
$00	R/W	Master interrupt control
$01	R/W	Master configuration control
$02	R/W	Port A interrupt vector
$03	R/W	Port B interrupt vector
$04	R/W	Counter/timer interrupt vector
$05	R/W	Port C data path polarity
$06	R/W	Port C data direction
$07	R/W	Port C special I/O control
Most often accessed registers		
$08	R/partial W	Port A command and status
$09	R/partial W	Port B command and status
$0A	R/partial W	Counter/Timer 1 command and status
$0B	R/partial W	Counter/Timer 2 command and status
$0C	R/partial W	Counter/Timer 3 command and status
$0D	R/W	Port A data
$0E	R/W	Port B data
$0F	R/W	Port C data
Port A specification registers		
$20	R/W	Port A mode specification
$21	R/W	Port A handshake specification
$22	R/W	Port A data path polarity
$23	R/W	Port A data direction
$24	R/W	Port A special I/O control
$25	R/W	Port A pattern polarity
$26	R/W	Port A pattern transition
$27	R/W	Port A pattern mask
Port B specification registers		
$28	R/W	Port B mode specification
$29	R/W	Port B handshake specification
$2A	R/W	Port B data path polarity
$2B	R/W	Port B data direction
$2C	R/W	Port B special I/O control
$2D	R/W	Port B pattern polarity
$2E	R/W	Port B pattern transition
$2F	R/W	Port B pattern mask

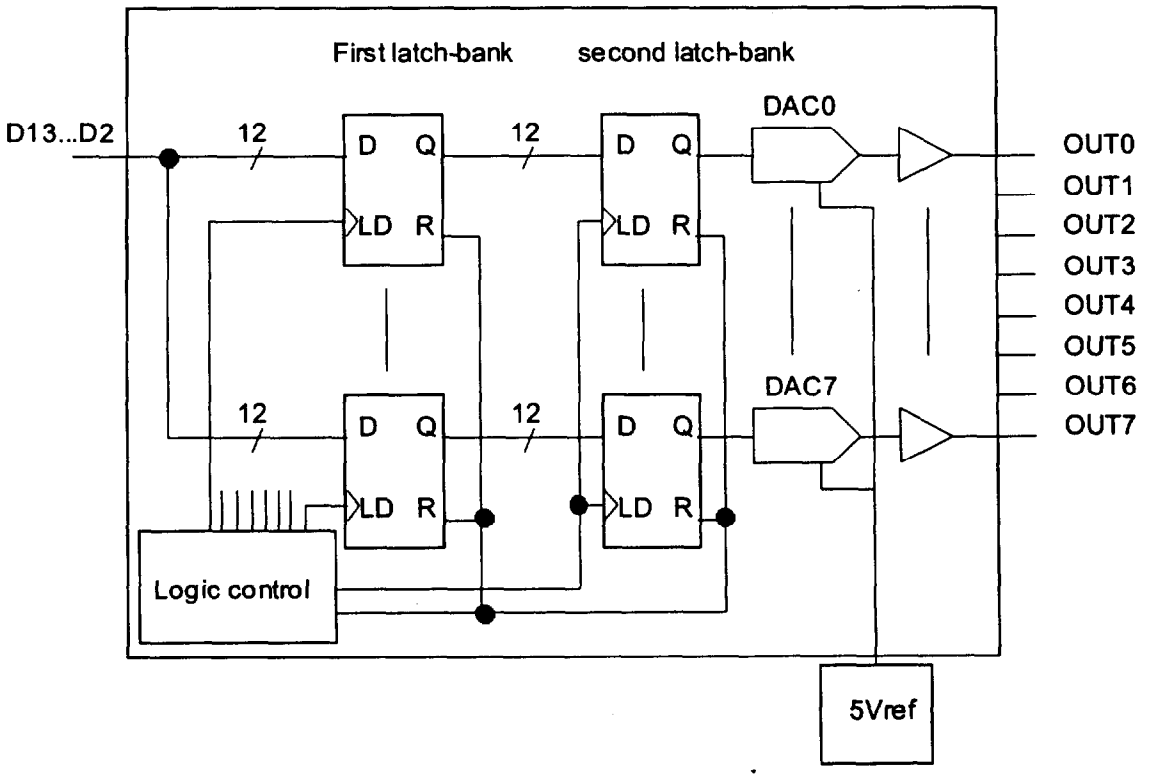
Counter/Timer related registers		
$10	R	Counter/Timer 1 current count MSB
$11	R	Counter/Timer 1 current count LSB
$12	R	Counter/Timer 2 current count MSB
$13	R	Counter/Timer 2 current count LSB
$14	R	Counter/Timer 3 current count MSB
$15	R	Counter/Timer 3 current count LSB
$16	R/W	Counter/Timer 1 time constant MSB
$17	R/W	Counter/Timer 1 time constant LSB
$18	R/W	Counter/Timer 2 time constant MSB
$19	R/W	Counter/Timer 2 time constant LSB
$1A	R/W	Counter/Timer 3 time constant MSB
$1B	R/W	Counter/Timer 3 time constant LSB
$1C	R/W	Counter/Timer 1 Mode Specification
$1D	R/W	Counter/Timer 2 Mode Specification
$1E	R/W	Counter/Timer 3 Mode Specification
$1F	R	Current vector

For more details on register programming and SCC functionality, please refer to the Zilog *"Z8036 Z-CIO/Z8536 CIO Counter/Timer and Parallel I/O Unit" technical manual.*

6 MPS MP7613

The IP modules work with octal DAC devices from Micro Power System to achieve to digital to analog conversion. One of these devices is used as per the conversion resolution required. The MP7613 converts in 12-bit resolution.

The internal architecture of these devices is divided in two latch sections. The first contains eight 16-bit data registers (first latch-bank, LB0 to LB7 registers), which store the DAC conversion codes. These registers may be individually accessed for each DAC output. However, the relevant DAC analog output is not updated when a new value is writing. The DAC Output Update register (DACREG, bit D1) allows the transfer, in a single cycle, of the content in the eight first latches (LB0 to LB7 registers) into the second latch-bank (LB0' to LB7') and update the analog output.



The reset DAC (DACREG register) is used to execute a general reset cycle into the converter (see the 6.2. section).

An on-board precision voltage reference of +5 V provides a very high temperature stability of 25 ppm/°C drift to the DAC converter.

6.1 DAC code registers (offset $40 to $4E)

The eight independent, read or write, DAC code registers contain the conversion codes for each analog output. The data bits are aligned to the bit D2 (12 bit, D2 to D13). When a new value is stored, the corresponding output is not affected. To update the analog output , it is necessary to make a write cycle in the DACREG register (see the following section).

DAC code format

IP-BUS 16-bit	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
DAC 12-bit	-	-	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	-	-

- : Undefined.

Note that the bits D[15 :14 ; 1-0] are unused.

6.2 DACREG: Output Update and reset register (offset $79)

This register transfers the first latch-bank content into the second latch-bank, which are directly connected to the MP7613 internal output amplifiers. In general application, the eight DAC code registers are set in a first step and then the DACREG operation is executed. At that time, only the required DAC outputs are changed, without alteration for the other not concerned outputs.

To execute this operation, the D1 bit of DACREG register (Update bit) must be set to 1.

To execute a reset operation, the D0 bit of DACREG register (Reset bit) must be set to 1.

IP-Bus	D7	D6	D5	D4	D3	D2	D1	D0
DACREG	-	-	-	-	-	-	Update	Reset

- : Undefined.

The following table shows the DAC states after a reset cycle.

	12-bits du DAC
First latch value	$0800
Second latch value	$0800
Analog output voltage ±10 V range	**0 Volt**

Note that the DAC function execute also a hardware reset at each system reset.

6.3 Conversion codes

The following equation gives the analog output voltage according to the conversion code (D parameter) for a 12-bit conversion in ±10V analog output ranges.

DAC 12-bit converter	
Output range	**Conversion code**
±10 V	$10 * (-1 + 2 * D / 2^{12})$

6.4 Precision reference voltage

The ADA-08 IP module provides a voltage reference for digital to analog conversion. It offers a very good temperature stability, less than 25 ppm/°C rift. It is factory calibrated, with a potentiometer, to give to DAC0 a minimum total error, in full scale conditions.

7 LM12H458 acquisition system

The ADC function works with the enhanced LM12H458 converter. This highly integrated data acquisition system is a 12-bit + sign with integrated sample-and-hold. This acquisition system operates in three different modes: 12-bit + sign, 8-bit + sign, and watchdog. The watchdog function generates an interrupt request when the input goes out from the range defined by the two-limit references. The acquisition and conversion times are software programmable with the on-chip timers.

An internal 32-word FIFO reduces the CPU overhead. When this FIFO is used, it is possible to stores up to 32 consecutive conversions before reading the FIFO values.

The LM12H458 has an auto-calibration mode, which executes an auto-zero offset voltage correction and an ADC linearity calibration with eight samples of the offset voltage. A short auto-zero correction is also provided for every conversion.

A synchronisation signal SYNC is present to operate either as an input or an output. When the SYNC signal is programmed as an input, a rising edge applied to this pin starts a conversion cycle or a watchdog comparison. The LM12458 internal architecture includes a sequencer, which executes programmable instructions. The instruction RAM holds up to eight executable instructions. The conversion modes, sequencer operation (continue or single), synchronisation capability, conversion resolution and the timer functions are defined into this instruction RAM.

One option with the "Sync" signal wired to the bus "Strobe" line is also available.

Recommended initialisation:
The following steps must be achieved to insure correct analog conversions.
Execute a full LM12H458 auto-calibration through the CONF register.

For more details on register programming and ADC functionality, please refer to the National Semiconductor *"LM12454/ LM12H454/ LM12458/ LM12H458 12-bit + sign data acquisition system with self-calibration"* data sheets.

7.1 Interrupt vector (ADIV) register (offset $61)

This read and write ADIV register includes the 8-bit vector for the interrupt acknowledge cycle, generates in response of a LM12458 interrupt request.

IP-Bus	D7	D6	D5	D4	D3	D2	D1	D0
ADIV	ADIV7	ADIV6	ADIV5	ADIV4	ADIV3	ADIV2	ADIV1	ADIV0

7.2 Enable ADC interrupt (EADI) register (offset $71)

The LM12H458 interrupt is enable when the D0 bit (enable bit) of read and write EAI register must be set to 1.

IP-Bus	D7	D6	D5	D4	D3	D2	D1	D0
EADI	-	-	-	-	-	-	-	Enable

- : Undefined

7.3 External synchronisation

The ADA-08 provides, on the 50-pin I/O connector interface, an external synchronisation signal (input or output). The SYNC signal controls the LM12H458 start conversion, and allows the synchronisation with other converter modules in multiple acquisitions system.

The synchronisation signal from the I/O connector is directly connected from the LM12H458 device, and detects a rising edge.
(One option with the "Sync" signal wired to the bus "Strobe" line is also available.)

7.4 Conversion codes

The following table gives the conversion codes for the 12-bit + sign and 8-bit + sign resolutions.

Voltage range	Conversion codes
±10 V	$\frac{V_{IN}+10}{20}\cdot N-1$
±5 V *	$\frac{V_{IN}+5}{10}\cdot N-1$
0 to +5 V *	$\frac{V_{IN}}{5}\cdot N-1$
0 to +10 V *	$\frac{V_{IN}}{10}\cdot N-1$

* Factory options

The N parameter defines the acquisition resolution for the analog to digital conversion (4096 for 12-bit, 256 for 8-bit conversions).

7.5 Precision reference voltage

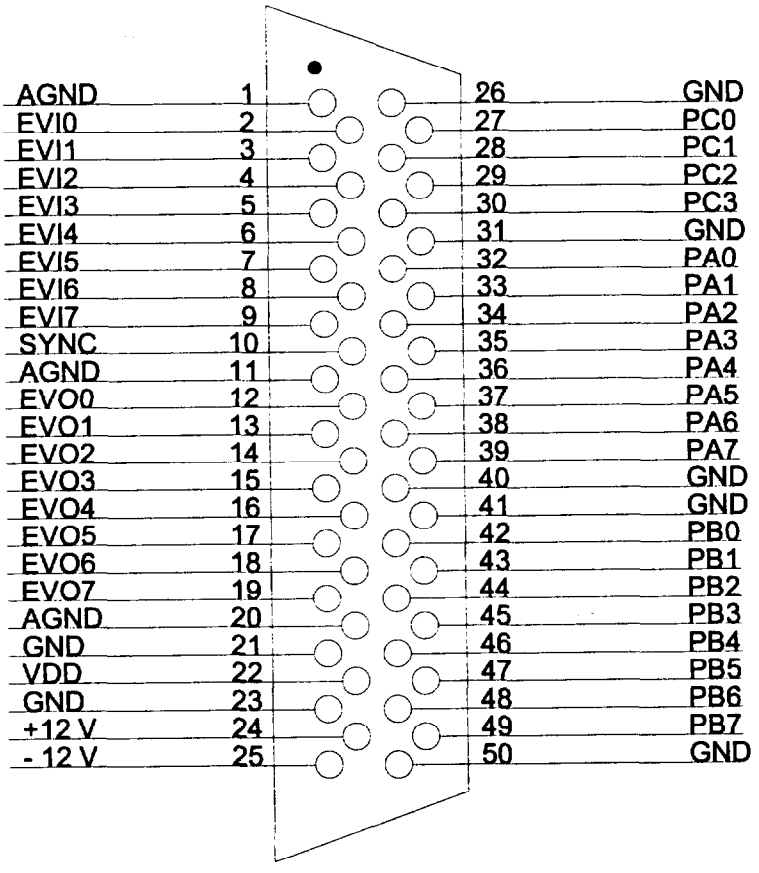
The ADA-08 module provides a second voltage reference for the analog to digital conversion. It offers a very good temperature stability, less than 25 ppm/°C drift. It is factory calibrated, with a potentiometer, in order to give the minimum full scale error.

7.6 Voltage range selection

The input voltage is ±10 V on the standard version.
The bipolar ±5 V, unipolar 0 to 5 V and 0 to 10 V are also available as a factory option or special order with minimum quantity(please contact ACTIS Computer marketing dpt).

8 Connection

The ADA-08 signals of the two general-purpose 8-bit ports and the special-purpose 4-bit ports, the eight analog output, and the eight single-ended or the four differential analog input are present on the 50-pin IP I/O connector. The power supply (+5V and ±12V) is also provided for external signal conditioning, and it's short-circuit protected.

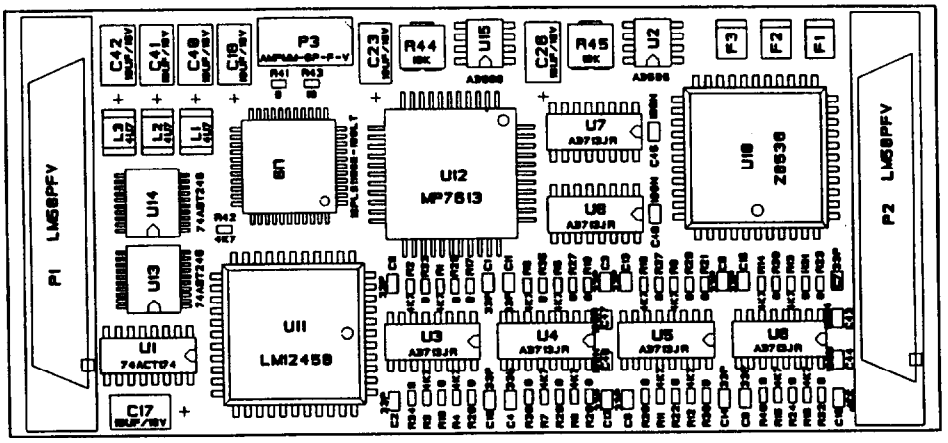


Signals	Description
PA[7:0]	Port A general-purpose 8-bit port
PB[7:0]	Port B general-purpose 8-bit port
PC[3:0]	Port C special-purpose 4-bit port
EVO[7 :0]	DAC output [7 :0]
EVI[7 :0]	ADC input[7 :0]
SYNC	ADC External synchro.

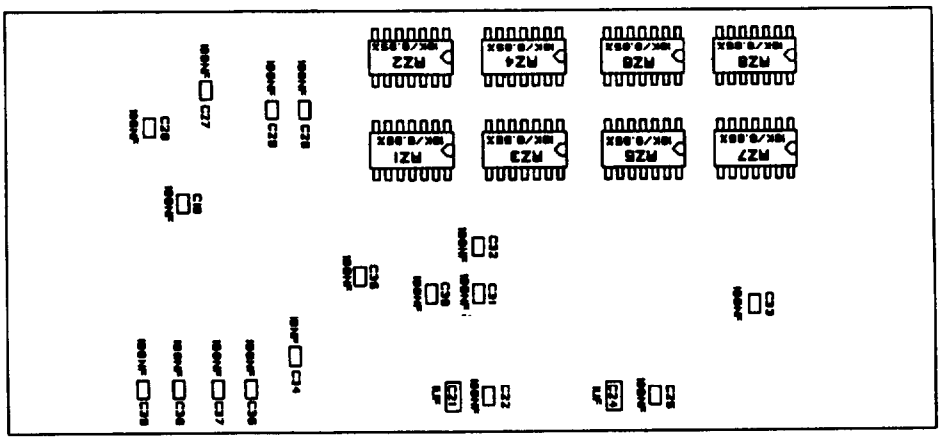
POWER	
Signals	**Description**
+12 V	+12 V, 500 mA max. fuse protected.
-12 V	-12 V, 500 mA max. fuse protected.
VDD	+ 5 V, 500 mA max. Fuse protected.
GND	Digital ground
AGND	Analog ground

9 Component location

Component side:



Back side :



Although the information contained in this document is believed to be correct, ACTIS Computer S.A. can not be held responsible for any error or for any resulting consequential losses. ACTIS Computer S.A. may change or improve the specifications of its products at any time without prior notification.

Appendix : Program examples

To use the DAC MP7613 (page 43) converter:
This program shows how to use the DAC converter. The value is placed in the DACx register (defined by the _dac structure) and the output is enable writing a 0x2 on the DACREG register.

```
void    main (void)
{
        short int               value, code;
        struct _dac             *dac_ptr;
        unsigned char   *dacreg_ptr;

        dac_ptr         = (struct _dac*) 0x4200040;     /* 0x420 0000 = ad base Ada08 */
        dacreg_ptr      = (unsigned char*) 0x4200079;

        code    = 4*0x000;                      /* 12 bits converter: LSB = bit2 */
        dac_ptr->dac_0 = code;          /* channel 0 : output = -10 volts */

        code = 4*0x800;
        dac_ptr->dac_1 = code;          /* channel 1 : output = 0 volts   */

        code = 4*0xfff;
        dac_ptr->dac_2 = code;          /* channel 2 : output = +10 volts */

        *dacreg_ptr = 0x2;              /* enable output */
}
```

To use the ADC LM12458 (page 45) converter:
This program shows how to use the ADC converter. First, calibrate the ADC with functions auto-zero and full scale. Second, initialize the RAMx register with the conversion mode. Third, start conversion and forth, read values in the FIFO register. The _adc structure define ADC registers.

```
void    main (void)
{
        volatile int            i;
        struct _adc             *adc_ptr = (struct _adc*) 0x4200000;
        short int               adc_value[8];

        adc_ptr->conf_reg = 0x2;                        /* reset */
        adc_ptr->conf_reg = 0x4;                        /* auto-zero */
        while ((adc_ptr->interr_stat & 0x8) != 0x8);    /* check if auto-zero has  been completed */
        adc_ptr->conf_reg = 0x8;                        /* Full scale */
        while ((adc_ptr->interr_stat & 0x10) != 0x10);  /* check if full scale has been completed */

        adc_ptr->ram_0 = 0x2000 + 2;                    /* acquisition time=2 / pause bit */
        adc_ptr->ram_1 = 0x2000;                        /* 8 acquisitions sequence preparation */
        adc_ptr->ram_2 = 0x2000;
        adc_ptr->ram_3 = 0x2000;
        adc_ptr->ram_4 = 0x2000;
        adc_ptr->ram_5 = 0x2000;
        adc_ptr->ram_6 = 0x2000;
        adc_ptr->ram_7 = 0x2000;
        adc_ptr->conf_reg = 0x2;                                        /* reset */
        adc_ptr->conf_reg = 0x1;                                        /* start */
        while ((adc_ptr->interr_stat & 0x4000) != 0x4000);              /* check end conversion */

    for(i=0; i<7; i++)    adc_value[i] = adc_ptr->fifo_reg;
}
```

Structures:

```
struct _dac
        {
        short int dac_0;
        short int dac_1;
        short int dac_2;
        short int dac_3;
        short int dac_4;
        short int dac_5;
        short int dac_6;
        short int dac_7;
        };
```

```
struct _adc
{
short int ram_0;
short int rsvd8;
short int ram_1;
short int rsvd9;
short int ram_2;
short int rsvd10;
short int ram_3;
short int rsvd11;
short int ram_4;
short int rsvd12;
short int ram_5;
short int rsvd13;
short int ram_6;
short int rsvd14;
short int ram_7;
short int rsvd15;
short int conf_reg;
short int rsvd16;
short int ena_int;
short int rsvd17;
short int interr_stat;
short int rsvd18;
short int tim_reg;
short int rsvd19;
short int fifo_reg;
short int rsvd20;
short int lim_stat;
};
```