Vigilant Technologies, Inc.

Performance ADC
(Analog Input) I/O Module

Hardware User's Manual

Version 1.00 Rev A Jan 1995

4700 SW 51st Street
Suite 204
Davie, FL 33314
Tel: (305) 797-9277 Fax: (305) 797-9065

Copyright

Warranties

Vigilant warrants its products to conform to current specifications in accordance with Vigilant's standard warranty. Testing is performed to the extent necessary as determined by Vigilant to support this warranty. Unless explicitly specified by customer order requirements not all device characteristics are necessarily tested.

Vigilant assumes no liability unless specifically agreed to in writing for customer's product design or infringement of patents or copyrights of third parties arising from use of Vigilant devices. No license, either express or implied, to Vigilant's patents, copyrights, or other intellectual property rights pertaining to any machine or combination of Vigilant's devices is hereby granted.

Life Support Policy

Vigilant Technologies' products are not authorized for use as critical components in life support devices or systems unless a specific written agreement pertaining to such intended use is executed between the manufacturer and an officer of Vigilant Technologies.

Life support devices or systems are devices or systems which (a) are intended for surgical implant into the body or (b) support or sustain life and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Table of Contents

Product Description

Vigilant's Performance ADC (PERF_ADC) High Density I/O modules are a low cost approach to add just the right amount of analog input to an I/O subsystem built around Vigilant's SlotSaver multifunction I/O coprocessor boards or any IndustryPack host board. The PERF_ADC I/O module converts at one million samples per second, up to 16 single-ended or 8 differential channels of analog input signals in either of two voltage ranges; unipolar 0 to +10 V or bipolar -5V to +5V, all in a single wide IndustryPack form factor. Placing four PERF_ADC I/O modules on a single SlotSaver board will provide for a maximum of 64 analog input I/O lines.

Two of the channels, channel one and two, may be jumpered to ground and +2.5V respectively. During initialization these may be sampled and stored on the carrier board for use during run-time when a gain and offset adjustment can be made in the software driver. This feature is ideal for modules installed on intelligent carrier boards such as Vigilant's SlotSaver boards, where the I/O coprocessor can apply the gain and offset adjustment prior to delivering the conversion values.

The module converts an input signal by first directing it through a 16/8 channel to 1 multiplexor, through an instrumentation op-amp circuit, and into a convertor. The convertor incorporates a track/hold acquiring the signal and converting it in less than 650 nanoseconds. Another 350 ns is needed for settling time in the multiplexor and op-amp circuit. Thus the net throughput rate is 1 million samples per second. The signals are resolved to 12 bits of resolution. All conversions are placed in a fifo 16 bits wide and 8K deep, in a two's complement form with the hardware doing the sign extension for simplification of programming. All conversions are done during a conversion sequence.

A conversion sequence consists of the sampling of sixteen channels specified in a sequencer. The channel selection is made during initialization when 16 channel numbers are written into a RAM storage. Channel numbers are then sampled from this sequence. For example; in a single-ended configuration, if all 16 channels are desired, then the sequence in the RAM is set to channel 1, 2, 3, .. thru 16. If only channel 1 is desired then the RAM is set to 1, 1, 1, .. 1. This setting would permit a single channel to be sampled at the full 1 million sample/sec rate. If each of the 16 channels are set as in the first example, then each channel would be sampled at 62.5Ksamples/sec. In a differential configuration, 16 channels are sampled even though only channels one through eight can be read from the sequencer. The sequencing logic insures a stable sampling interval minimizing jitter. The conversion cycles may be set up to execute once, writing just sixteen values in the fifo, or continuously, writing values into the fifo as long as there is room, until commanded to stop. The start of the conversion cycle can also be externally triggered. The IndustryPack supports two interrupt request signals to the host. If unmasked, one interrupt will assert upon completion of sixteen conversions in once mode or when the fifo is full in continuous mode. The other interrupt will assert when the fifo is half full in either mode. Commands and status are done via the IndustryPack's I/O accesses to various registers.

This module supports both a software settable interrupt vector and interrupt acknowledge with vector and the ID prom data. The ID information on the board can be used to for initialization configuration .

The circuit board is designed with a unique ground plane layer that is split between the digital and analog ground regions. The two are connected through an inductor type of EMI suppressor providing a low noise area for analog input and conversion.

The figure below shows a block diagram of the PERF_ADC module:

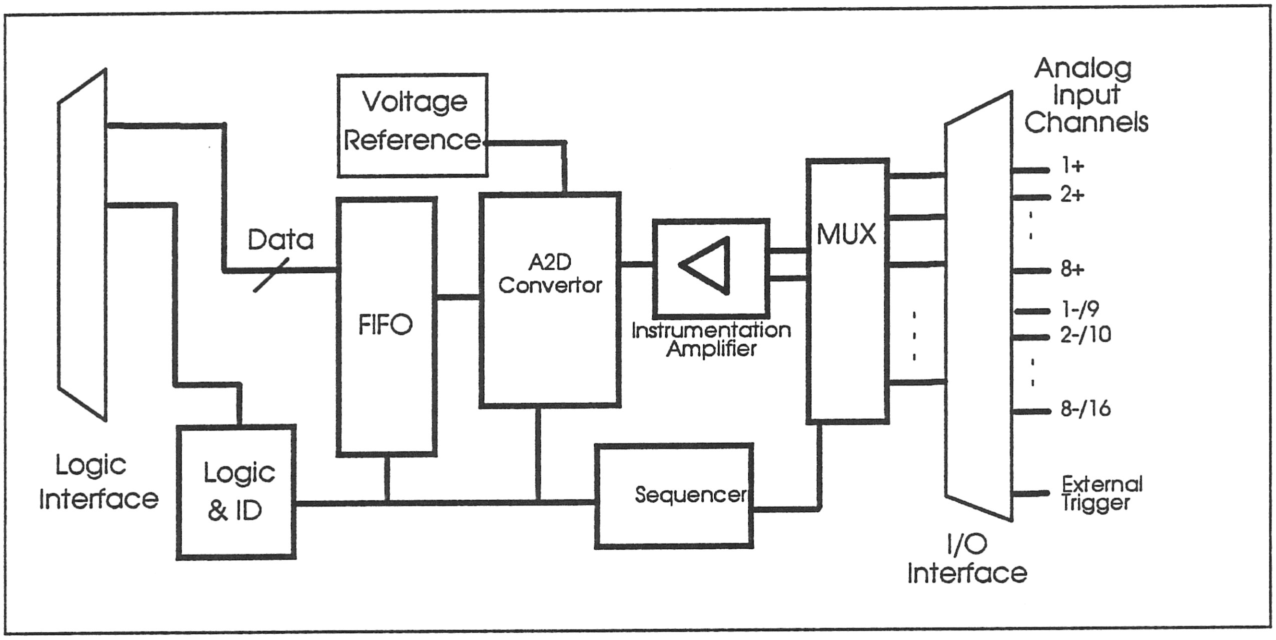


Figure 1: Block Diagram of PERF_ADC

A time line of the events in a conversion sequence in once mode is depicted below:

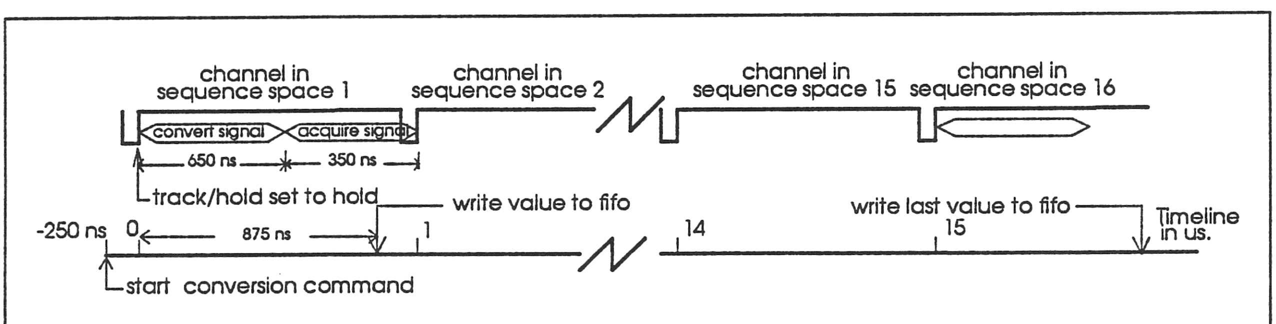


Figure 2: Timeline of Conversion Sequence

Address Mapping

The PERF_ADC registers are accessed using the IndustryPack I/O space. The ID information is accessed using the IndustryPack ID space, and the interrupt vector is accessed with the IndustryPack IntSel space.

The following table maps the address as an offset from the beginning of the modules I/O space:

Address Offset	Access width	Read/ Write	Port
00 hex	byte(do..d3)	write	channel number sampled in sequence space 1
02	byte "	write	channel number sampled in sequence space 2
04	byte "	write	channel number sampled in sequence space 3
06	byte "	write	" " " space 4
08	byte "	write	" " " space 5
0a	byte "	write	" " " space 6
0c	byte "	write	" " " space 7
0e	byte "	write	" " " space 8
10	byte "	write	" " " space 9
12	byte "	write	" " " space 10
14	byte "	write	" " " space 11
16	byte "	write	" " " space 12
18	byte "	write	" " " space 13
1a	byte "	write	" " " space 14
1c	byte "	write	" " " space 15
1e	byte "	write	" " " space 16
20	b/w	read	read status register
22	b/w	write	start conversion
24	b/w	write	stop conversion
26	byte	write	write commands
28	byte	write	set interrupt vector
2c	b/w	rd/wr	reset fifo
2e	b/w	rd/wr	retransmit fifo
30	word	read	read fifo

Table 1: I/O Address Mapping of PERF_ADC I/O

The following table maps the address as an offset from the beginning of the modules IntSel space:

Address Offset	Access width	Read/ Write	Cause	Vector Read
00 hex	byte(do..d7)	read	Int Req0* - in once mode: at end of 16th sample. - in continuous mode: when fifo full.	Int Vector & feh
02	byte "	read	Int Req1* when fifo half full.	Int Vector | 01h

Table 2: IntSel Address Mapping of PERF_ADC I/O

IntReq0 will stay asserted until an interrupt acknowledge vector is read. IntReq1 will stay asserted until a value is read from the fifo.

The status and command register formats are described in the figures below:

	d7	d6	d5	d4	d3	d2	d1	d0
Offset 0x26	.	.	.	.	.	MK1	MK0	O/C

O/C = Once/Continuous; (once = 1, continuous = 0)
MK0 = Mask interrupt 0 (mask = 1, unmasked = 0)
MK1 = Mask interrupt 1 (mask = 1, unmasked = 0)
. = Not used

Figure 3: Command Register

	d7	d6	d5	d4	d3	d2	d1	d0
Offset 0x20	OTR	CE	FF	FHF	FE	O/C	R/S	DAV

DAV = data available output (0 = change in data.)
R/S = Running/Stop (Running = 0, Stop = 1)
O/C = Once/Continuous; (once = 1, continuous = 0)
FE = fifo empty (Fifo empty = 0)
FHF = fifo half full flag (Fifo half full = 0)
FF = fifo full flag (Fifo full = 0)
CE = Conversion error flag (Error = 0)
OTR = Out of range flag (out of range = 0)

Figure 4: Status Register

The channel numbers written into the sixteen sequencing spaces are four bits in d0 through d3 and are the inverse of the hex numbers 0 through f. For example to write channel 1 into the first sequencing space, write 0fh into the low byte at offset 0; to write channel 2 into the second space write 0eh into the low byte at offset 2.

The start conversion and stop conversion commands are writes with don't care data in byte or word transfers. The reset fifo and retransmit fifo can be reads with don't care data.

The CE, conversion error flag, if asserted will stay asserted until a stop conversion command is issued. An asserted CE may indicate a malfunctioning convertor or module. The DAV status should be high when not running. During conversion sequences DAV will be low from 150 to 300 ns during each microsecond. The OTR flag will be asserted if any conversion in a sequence is completed with an input signal that is out of range. The flag will stay asserted until a stop conversion command is issued. The FHF will go low when the fifo is half full and will remain low until the fifo is less than half full.

I/O Pin Assignments

The following table gives the pin assignments for the I/O connector:

Pin Number	Channel	Function
1	GND	---
2	Channel 1+	analog input or jumpered to gnd
3,4	GND	---
5	Channel 2+	analog input or jumpered to +2.5V
6,7	GND	---
8	Channel 3+	analog input
9,10	GND	---
11	Channel 4+	analog input
12,13	GND	---
14	Channel 5+	analog input
15,16	GND	---
17	Channel 6+	analog input
18,19	GND	---
20	Channel 7+	analog input
21,22	GND	---
23	Channel 8+	analog input
24,25	GND	---
26	Channel 9+/1-	analog input
27,28	GND	---
29	Channel 10+/2-	analog input
30,31	GND	---
32	Channel 11+/3-	analog input
33,34	GND	---
35	Channel 12+/4-	analog input
36,37	GND	---
38	Channel 13+/5-	analog input
39,40	GND	---
41	Channel 14+/6-	analog input
42,43	GND	---
44	Channel 15+/7-	analog input
45,46	GND	---
47	Channel 16+/8-	analog input
48	GND	---
49	VREF	+2.5V voltage reference
50	EXTRG	external trigger

Table 3: I/O Pin Assignments

IndustryPack Logic Interface Pin Assignments

Figure 2 below shows the pin assignments for the IndustryPack logic interface.Pins that are marked with an "nc" are defined by the specification but not used by the PERF_ADC module.

Signals		Pin Numbers	
GND	GND	1	26
CLK	+5V	2	27
Reset*	R/W*	3	28
D0	IDSel*	4	29
D1	nc	5	30
D2	nc	6	31
D3	nc	7	32
D4	IntSel*	8	33
D5	nc	9	34
D6	IOSel*	10	35
D7	nc	11	36
D8	A1	12	37
D9	nc	13	38
D10	A2	14	39
D11	nc	15	40
D12	A3	16	41
D13	IntReq0*	17	42
D14	A4	18	43
D15	IntReq1*	19	44
nc	A5	20	45
nc	nc	21	46
-12V	nc	22	47
+12V	ACK*	23	48
+5V	nc	24	49
GND	GND	25	50

Figure 5: IndustryPack Logic Interface Pins

Programming

The PERF_ADC I/O module will in general be programmed in the following manner:

1. During initialization the following should be done:
   - Write the channel numbers into the sequencer RAM. Invert the data bits as the RAM outputs data that is the complement of the stored data. For example, to write channel one in the first space store an 0x0f, for channel two store 0x0e, etc.
   - Write to the command register setting the once/continuous bit and the interrupt mask bits.
   - Write the interrupt vector if required.
   - Clear the fifo.
   - Read the status register to verify correct status.

2. During runtime the following should be done:
   To do a single conversion cycle (O/C set to once mode):
   - Write to the start conversion command. Sixteen samples will be converted and written to the fifo.
   - Either check the status of the fifo not empty flag or wait for the interrupt upon a sixteen channel sequence completion to begin reading the fifo. If an interrupt is serviced, the interrupt acknowledge vector must be read to deassert the interrupt request.
   - Read the sixteen values.
   - Verify that the conversion cycle is stopped.

   To do a continuous conversion cycle (O/C set to continuous mode):
   - Write to the start conversion command. Samples will be converted and written to the fifo until the stop command is issued. The stop command will terminate the conversion cycle at the end of the current channel conversion.
   - Either check the status of the fifo not empty flag or wait for the interrupt on fifo full or half full to begin reading the fifo.
   - Read the values.
   - If an interrupt on fifo full is serviced, the software must determine that data is not lost due to overrunning the fifo.

Other Operating Notes:
- When using the external trigger to start a conversion cycle, the external trigger signal must go low for at least one clock cycle or 125ns. The EXTRG I/O pin is pulled high through a 4.7K resistor.
- The programmable interrupt vector will store bits d1 thru d7. Bit d0 is not saved as the vector when read during and interrupt acknowledge cycle will place a 0 at d0 for interrupt 0 or a 1 when acknoweldging interrupt 1.

The digital representation of the analog input voltage levels is indicated below:

± 5V Input Voltage Range			0 to +10V Input Voltage Range		
Digital Value		± 5V	Digital Value		0 to +10V
(dec)	(hex)		(dec)	(hex)	
2047	7ff	+4.997	4095	fff	+9.997
1024	400	+2.500	3072	c00	+7.500
-1	fff	0.000	2048	800	+5.000
-1024	c00	-2.500	1024	400	+2.500
-2048	800	-5.000	0	000	0.000

Table 4: Analog to Digital Transfer Function

The following source code listing is an example of a driver for the PERF_ADC module.

The driver is made up of three routines; an initialization, a runtime routine, and an interrupt service routine. The driver is intended for a PERF_ADC configured as 16 channels of single-ended with an input range of 0 to +10V. The function of the driver is to take 1K samples total of the 16 single-ended analog input signals when triggered by software command. The host is interrupted by the fifo half full signal and the interrupt service routine places the samples in the dual-ported ram to be read by the PC. The initialization routine sets the sequencing for all channels 1 through 16 and the command settings to continuous mode and interrupt 1 unmasked. The runtime routine starts a conversion cycle. The interrupt service routine reads 1K samples and places them in the dual-port common memory interface with the PC. (This assumes that the PERF_ADC module is mounted on a Vigilant Technologies SlotSaver SS1000 intelligent I/O carrier board, and that the SlotSaver is inserted into the PC bus.)

The following is the source code listing:

```
/************************************************************************
                   PERF_ADC.c
       Copyright by Vigilant Technologies, Inc.1994
 ************************************************************************
Description: Performance ADC driver: sixteen channels of 12-bit analog input
             - this driver converts sixteen analog input channels and
               reads their digital values placing them in the dual-port
               ram interface with the pc.  The values are in 16 bit two's
               complement form.

             - There are three routines: init, exec, and isr
               - init:  initializes the sequencing channel numbers, command
                        register and interrupt vectoring.
               - exec:  starts the conversion.
               - isr:   reads the values from the fifo and copies them to the
                        dual-port ram, checks for errors.

Implementation:  It assumes the module is set for 16 channels of single-ended
                 input, unipolar 0 to +10V range.
                 Intended to be run on a Vigilant SlotSaver SS1000 186 cpu. in
                 Slot A.

Limitations:

Written by:
Modified  :
```

```
***************************************************************************/
#include <dos.h>

#define TRUE            1
#define FALSE           0
#define NOERROR         0
#define STS_IDLE        0xf3  /* fifo empty, cont mode, not running, no errors*/
#define STS_RUN_FFNE    0xf8  /* fifo not empty, cont mode, running, no errors*/
#define CMD_CONT_INT1   0xfa  /* set command reg to continuous, intl unmask */
#define FIFO_EMPTY_MSK 0x08   /* mask of fifo empty bit in status register */
#define PERF_STATUS_REG        0x20
#define PERF_START_CONVERSION 0x22
#define PERF_STOP_CONVERSION  0x24
#define PERF_WRITE_COMMANDS   0x26
#define PERF_WRITE_INT_VECTOR 0x28
#define PERF_RESET_FIFO       0x2c
#define PERF_RETRANSMIT_FIFO  0x2e
#define PERF_READ_FIFO        0x30

#define IP_A_IO         0x4000    /* module A I/O space in the SlotSaver SS1000 */
#define DP_MEM          0x8000    /* dual-port memory start in I/O space in the
                                     SlotSaver SS1000    */
/* interrupt registers in 186 I/O space*/
#define INTIN0      0xff38 /*  */
#define INTIN1      0xff3A /*  */
#define INTIN2      0xff3C /*  */
#define INTIN3      0xff3E /*  */
#define INTTMR      0xff32 /*  */
#define INTDM0      0xff34 /*  */
#define INTDM1      0xff36 /*  */
/*   8259 Int Controller Registers */
#define ICW1        0xc000
#define ICW2        0xc002
#define ICW4        0xc002
#define OCW1        0xc002
#define OCW2        0xc000


// ------- Function prototypes -------
int perfInit(short int ioBaseAddr, short int comMemStart);
int perfExec(short int ioBaseAddr, short int comMemStart);
void interrupt far perfHalfFullA();
void setupInterrupts(void);
void enableInterrupts(void);

/* driver SIS data structure from PC side memory space*/
//struct PERFADC_16Chl {
//            int slotsaver_status_flag;
//            int pc_status_flag;
//            short perfadc_chl[1024][16];
//              } perfadc_struct;
/* structure offsets in I/O space on SlotSaver */
#define SS_STATUS_FLAG_LOC  0    /* SS1000 side offset into dualport */
#define PC_STATUS_FLAG_LOC  2
#define PERF_CHL_LOC        4    /* offset of start of samples */


/******* Global variables *******/
int slot_loc[4];     /* 1 if occupied, 0 if not, must set to occupied
                        to set vector for slot slot A = 0, B = 1, etc.*/
typedef struct{      /* structure to hold the two int service rtns/slot */
    void interrupt far (*isr1_m) ( );
    void interrupt far (*isr2_m) ( );
}ISRS;

ISRS intSers[4] = {                  /* SlotSaver four IP slots */
    {perfSeqorFFFA, perfHalfFullA},
    {perfSeqorFFFB, perfHalfFullB},
    {perfSeqorFFFC, perfHalfFullC},
    {perfSeqorFFFD, perfHalfFullD},
};
unsigned char intrp_8259_msk;
```

```
/*--------------------------------------------------------------------
| Prototype  : perfInit(short int ioBaseAddr, short int comMemStart)
|--------------------------------------------------------------------
| Function   : initializes the PERF module
| Input      : SS1000 IP I/O space start address, dualport memory
| Output     : error status: 0 noerror, status if error
| Note       :
| Dependency :
---------------------------------------------------------------------*/
int perfInit(short int ioBaseAddr, short int comMemStart)
{
 int i;
 unsigned short ioloc;
 unsigned char tmpch, seq;
 ioloc = ioBaseAddr;

 /** set command to once mode and unmask interrupt 1 **/
 outport(ioBaseAddr+PERF_WRITE_COMMANDS,CMD_CONT_INT1);

/** issue a stop command to clear otr **/
outport(io_addr+PERF_STOP_CONVERSION,0xffff);

 /** reset fifo **/
 inpw(ioBaseAddr+PERF_RESET_FIFO);

 /** set interrupt vector to 0xff **/
 outport(ioBaseAddr+PERF_WRITE_INT_VECTOR,0xffff);

 /** set channel sequence to 1 through 16 in sequential order **/
 seq = 0x0f;
 for(i=0;i<16;i++)
 {
  outportb(ioloc,seq--);    /* remember channel number is inverted */
  ioloc = ioloc+2;
 }

 setupInterrupts();  /* but do not enable */
 enableInterrupts();

 /** check status register **/
 tmpch = inportb(ioBaseAddr+PERF_STATUS_REG);
 if(tmpch == STS_IDLE)
  return(NOERROR);
 else
  return((short)tmpch);
}

/*--------------------------------------------------------------------
| Prototype  : perfExec(short int ioBaseAddr, short int comMemStart)
|--------------------------------------------------------------------
| Function   : runtime routine for the PERF module; start conversions
| Input      : SS1000 IP I/O space start address, dualport memory
| Output     : error status: 0 noerror, status if error
| Note       :
| Dependency :
---------------------------------------------------------------------*/
int perfExec(short int ioBaseAddr, short int comMemStart)
{
 int ii,jj;
 short dp_loc;
 unsigned char tmpch;
 dploc = comMemStart + A2D_CHL_LOC;

 /** start conversion **/
 outport(ioBaseAddr+PERF_START_CONVERSION,0xffff);

 /** For now wait a couple of us until fifo gets a value **/
 jj = 2;
 while(jj--);

 /** Verify that status is normal before returning **/
 tmpch = inportb(ioBaseAddr+PERF_STATUS_REG);
 if((tmpch & 0xfe) == STS_RUN_FFNE)      /* don't care on DAV */
  return(NOERROR);
```

```
 else
  return((short)tmpch);
}

/*--------------------------------------------------------------------
| Prototype   : perfHalfFullA(void)
|--------------------------------------------------------------------
| Function    : interrupt service routine for the PERF module; copies
|             : values into dual-port memory structure.
| Input       : none
| Output      : error status: 0 noerror, status if error
--------------------------------------------------------------------*/
void interrupt far perfHalfFullA()
{
 short dploc;
 int jj;
 outport(IP_A_IO+VA2D_STOP_CONVERSION,0xffff);/** stop conversion **/
 dploc = DP_MEM + PERF_CHL_LOC;
 for(jj=0;jj<1024;jj++)              /* read 1024 values from fifo */
 {
  outport(dploc,inpw(IP_A_IO+VA2D_READ_FIFO));
  dploc = dploc +2;
 }
 inpw(IP_A_IO+VA2D_RESET_FIFO);         /* reset the fifo */

 /** Verify that status is normal before returning **/
 tmpch = inportb(ioBaseAddr+PERF_STATUS_REG);
 if(tmpch  == STS_IDLE)
  outport(DP_MEM+SS_STATUS_FLAG_LOC,NOERROR);
 else
  outport(DP_MEM+SS_STATUS_FLAG_LOC,tmpch);

 outportb(OCW2,0x20);        /* Non-specific EOI to 8259 */
 outport(INTEOI, 0x8000);    /* Non-specific EOI to 80C186 */
}

/*--------------------------------------------------------------------
| Prototype   : setupInterrupts(void)
|--------------------------------------------------------------------
| Function    : sets up interrupt vectors for 186, and initializes the
|             : 8259.
| Input       : none
| Output      : none
--------------------------------------------------------------------*/
void setupInterrupts(void)
{
 int i;
 unsigned int far *mem_off, *mem_seg;
 unsigned char msk_temp;
 msk_temp = 1;
 intrp_8259_msk = 0xff;     /* default is all ints masked.  */
 /* setup vectors for the int service routines */
 for(i=0;i<4;i++)    /* vectors for IP slots only */
 {
  if(slot_loc[i] == OCCUPIED)
  {
   mem_off = ( unsigned int far *) MK_FP(0x0000, 0x0080+i*8);
   mem_seg = ( unsigned int far *) MK_FP(0x0000, 0x0082+i*8);
   *mem_off = FP_OFF((void far *)intSers[i].isr1_m);
   *mem_seg = FP_SEG((void far *)intSers[i].isr1_m);
   intrp_8259_msk = intrp_8259_msk ^ (msk_temp << (i*2));

   mem_off = ( unsigned int far *) MK_FP(0x0000, 0x0084+i*8);
   mem_seg = ( unsigned int far *) MK_FP(0x0000, 0x0086+i*8);
   *mem_off = FP_OFF((void far *)intSers[i].isr2_m);
   *mem_seg = FP_SEG((void far *)intSers[i].isr2_m);
   intrp_8259_msk = intrp_8259_msk ^ (msk_temp << (i*2+1));
  }
 }
 /********  Initialize the 8259    *********/
 /****  - assumes a default priority     ****/
 outportb(ICW1,0x13); /* ICW1: LTM = edge, SNGL = 1, IC4 needed */
 asm { nop; nop ;}
 outportb(ICW2,0x20); /* ICW2: vector addresses start at 32dec*/
```



```
   asm { nop; nop ;}
   outportb(ICW4,0x01); /* ICW4: 8086 mode, Normal EOI, non-buffered*/
   asm { nop; nop ;}
   outportb(OCW1,0xff); /* OCW1: mask all interrupts for now */
   asm { nop; nop ;}
   /********  End of 8259 init   *********/

   /******** Initialize the 80C186 interrupt controller   ********/
   outport(INTIN0,0x0028);    /* Cascade mode, Priority 0, Mask set, edge */
   outport(INTIN1,0x000c);    /* Serial on board: Fully nested, Priority 4, Mask set, edge */
   outport(INTIN3,0x000d);    /* PC int : Fully nested, Priority 5, Mask set, edge */
   outport(INTDM0,0x000a);    /* Priority 2, Mask set*/
   outport(INTDM1,0x000b);    /* Priority 3, Mask set*/
   outport(INTTMR,0x0009);    /* Priority 1, Mask set*/
}

void enableInterrupts(void)
{
   outportb(OCW1,intrp_8259_msk); /* OCW1: unmask designated interrupts*/
   outport(INTIN0,0x0020);        /* unmask the 8259 to 186 int */
}
```

Hardware Configuration (Factory set)

The PERF_ADC module is factory configured for either unipolar, 0 to +10V, or bipolar, -5V to +5V, analog inputs, and for either single-ended or differential signals. The options are selected by placing shunts or jumpers. The following table describes the position of the shunts:

	Unipolar(0 to +10V)		Bipolar (-5V to +5V)	
	Jumper	Position	Jumper	Position
Jumpers:	J3	1 - 2	J3	2 - 3
	J4	1 - 2	J4	2 - 3
	J5	1 - 2	J5	2 - 3

	Single-ended Inputs		Differential Inputs	
	Jumper	Position	Jumper	Position
Jumpers:	J9	2 - 3	J9	1 - 2
	J8	1-2	J8	Do Not Install

	Other Jumper Functions		
	Jumper	Position	Function
Jumpers:	J6	Installed	Connects Channel 1+ to ground
	J7	Installed	Connects Channel 2+ to 2.5V reference

Table 5: Jumper Settings

The Unipolar/Bipolar configuration may be changed by changing the jumper locations. The Single-ended/Differential configuration cannot be changed without removal of the multiplexor. **(WARNING! Take precautions to avoid electrostatic discharge that may damage the components.)** Contact the factory if a change in the Single-ended/Differential configuration is desired.

The analog input circuit from the multiplexor to the convertor is a low power instrumentation amplifier with a gain of one half. The convertor thus sees a 0 to +5V input for a 0 to +10V differential input signal at the I/O pin, and a $\pm$2.5V input for a -5 to +5V differential input signal at the I/O pin. In single-ended mode, Vin- is jumpered to ground. The circuit is depicted below:

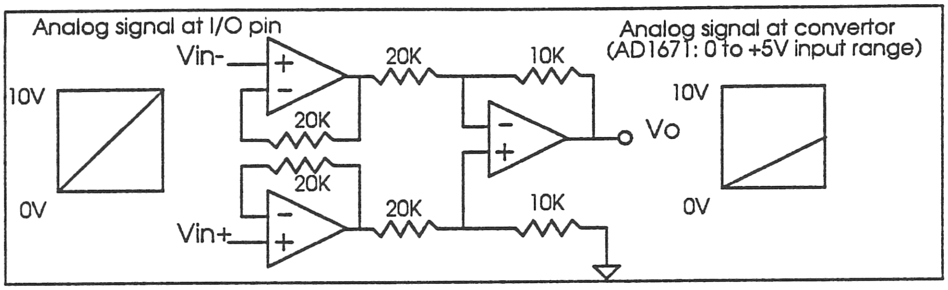


Figure 6: Analog Input Circuit for Unipolar configuration

Analog input channels 1 and 2 may be jumpered to a fixed reference value of ground and +2.5V respectively. Install jumpers at J6 or J7 to connect ground or +2.5V respectively to these channels. Disconnect any inputs to the corresponding I/O pins at the I/O connector. If the module is configured for differential mode, connect channels 1- and 2-I/O pins on the connector to any of the ground pins on the I/O connector.

ID Prom

The PERF_ADC I/O module is IndustryPack compatible and includes ID information. The ID data holds 12 bytes of significant information. The ID information permits software configuration management. The user's software, or a supplied driver, may verify that the device it expects is actually installed at the location it expects, and is nominally functional.

Standard data in the ID PROM on the PERF_ADC module is shown in the table below. For more information on the IndustryPack ID, refer to the IndustryPack Logic Interface Specification, available from GreenSpring Computers.

The ID information is accessed via the dedicated ID space. The data is available on the low 8 bits only. Thus sequential addressable locations on the PROM are in increments of two at even locations for little endian CPUs and at odd locations for big endian CPUs. The following table assumes little endian access modes.

Address(hex)	Data	Value(hex)
3E ll 18	Available for User ll Available for User	
16	Checksum	(00)
14	Number of bytes used = 11dec	(0b)
12	Driver ID, high byte	(00)
10	Driver ID, low byte	(00)
0E	Reserved	(00)
0C	Revision Number A	(41)
0A	Model No Digital I/O	(50)*
08	Manufacturer ID Vigilant	(A7)
06	ASCII "C"	(43)
04	ASCII "A"	(41)
02	ASCII "P"	(50)
00	ASCII "I"	(49)

Table 6: ID Data

* Note: Model No: 50hex for single-ended, 51hex for differential.

IndustryPack / VITA-4 Module Compliance

The PERF_ADC I/O module conforms to the following IP / VITA-4 module features.

1.	**Size**	Single-size Type I.
2.	**Select spaces**	ID space, I/O space, and Vector space.
3.	**Data Width**	16-bit.
4.	**Byte Strobes**	Byte strobes are ignored.
5.	**Response**	ID accesses: no wait states. I/O accesses: read fifo, write channel sequence - one wait state, all other I/O accesses- one wait state. Vector accesses: no wait states. Carrier board holds supported.
6.	**Clock Speed**	8 MHz.
7.	**DMA**	DMA not supported.
8.	**Reset**	The module command register will be set to once mode, interrupts masked.
9.	**Signal Definition**	Conforms to specification.
10.	**Timing**	Conforms to specification.
11.	**Mechanical**	Conforms to specification.

Board Outline

Figure 7 below shows a layout of the module. Resistor and jumper locations are detailed for hardware configuration purposes.

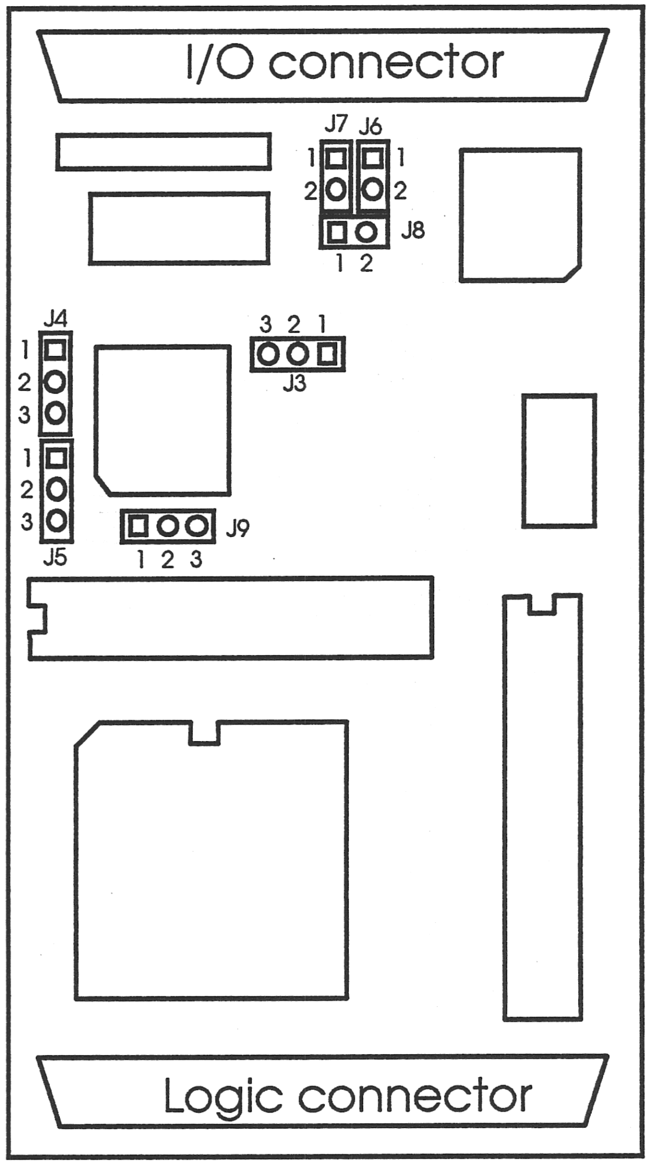


Figure 7: Board Outline (Not to scale)

Specifications

A2D Converter
AD1671

No. of analog input lines
16 single-ended without reference inputs, or 14 plus gnd plus 2.5V reference, or 8 differential without reference inputs, or 6 plus gnd plus 2.5V reference

Analog Input Range
0 to +10V, or -5 to +5V factory configurable.

Converter accuracy
12 bits ± 1.5 lsb typ

Temperature drift
± 50 ppm/°C

Throughput
1Million samples per second; one channel in 1us, 16 channels in 16 us.

Logic Interface
Programmable sequencing, 8K deep FIFO buffer for sampled data, status register, command register, interrupt vector register.

Data Format
16 bit two's complement hardware sign extended

Interrupts
Vectored, programmable.
- IntReq0: 16 channel sequence completion in once mode or fifo full in continuous mode.
- IntReq1: fifo half full.

Data Transfer Modes
8 or 16 bit accesses

Wait States
for fifo read and channel sequence writes = one
all others = zero

Dimensions
1.8 x 3.9 x 0.340 inches

Power requirements
200 mA @ 5V typ
40 mA @ +12V
40 mA @ -12V

Environmental

Operating Temperature:	0 to 70°C
Humidity:	5 to 95% non-condensing
Storage Temperature:	-20 to +85°C

Order Information

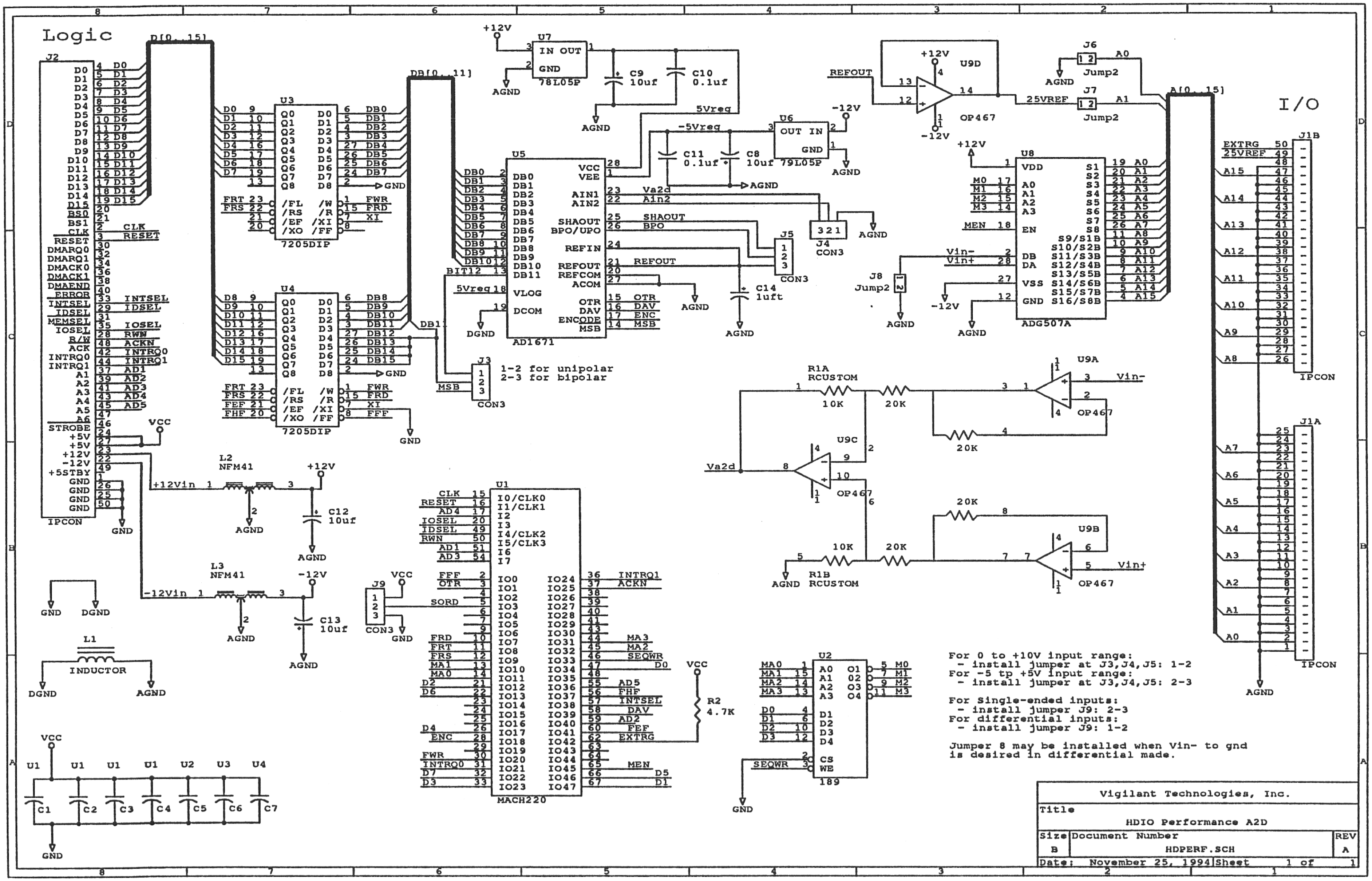
HD-ADC-PERF-x	High Performance16 channel High Density I/O Module
x takes the values:	a = ±5V, single-ended, or
	b = ±5V, differential, or
	c = 0 to +10V, single-ended,or
	d = 0 to +10V, differential

For Orders Contact:

Vigilant Technologies, Inc.
4700 SW 51st Street, Suite 204
Davie, FL 33314
tel: (305) 797-9277
fax: (305)797-9065

Schematics

Schematics are provided here for customer *reference only*. The information was current at the time the printed circuit board was last revised. This information is not necessarily current or complete manufacturing data, nor is it part of the product specification. All information following is Copyright Vigilant Technologies, Inc.



Logic
I/O
For 0 to +10V input range:
- install jumper at J3,J4,J5: 1-2
For -5 tp +5V input range:
- install jumper at J3,J4,J5: 2-3
For Single-ended inputs:
- install jumper J9: 2-3
For differential inputs:
- install jumper J9: 1-2
Jumper 8 may be installed when Vin- to gnd
is desired in differential made.
1-2 for unipolar
2-3 for bipolar
Vigilant Technologies, Inc.
Title
HDIO Performance A2D
Size B
Document Number HDPERF.SCH
REV A
Date: November 25, 1994
Sheet 1 of 1