

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

December 9, 2001

© FISCHER COMPUTER SYSTEMS, 2001

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

TABLE OF CONTENTS

OVERVIEW.....	2
EXTERNAL CONNECTIONS.....	3
HARDWARE DESCRIPTION.....	4
PROGRAMMER'S PERSPECTIVE.....	5
API LIBRARY.....	13
DRIVERS AND IOCTL CALLS.....	18
DEMONSTRATION PROGRAM.....	22
SAMPLE CONFIGURATIONS.....	23
VISUAL BASIC PROGRAMMING.....	25
SEC-PC TO SEC-PCI PORTING.....	26
CONTACT INFORMATION.....	29

OVERVIEW

The SEC-PCI provides real-time hardware-based quadrature decoding, counting and response facilities via a reconfigurable PCI form-factor PC card. SEC-PCI features include:

- 4 quadrature channels or axes
- 4 input, 12 output and 8 input/output external general purpose signals
- 4 input and 4 output external real-time signals
- 24-bit counters
- 4 MHz maximum count rate
- Digital filtering with error detection
- Up/Down, quadrature $\times 1$, $\times 2$ and $\times 4$ decoding
- Real-time compare, capture, preset, underflow and overflow functions
- Robust user notification system via polling or interrupt
- Windows 98/NT/2000 and Linux drivers
- Simplified API library
- Customizable FPGA-based design
- Customizable external card connector and front-end circuitry
- 33 MHz, 32-bit Universal PCI card
- User programmable EEPROM

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

EXTERNAL CONNECTIONS

The SEC-PCI hardware is organized around five 8-bit ports. These ports are named port A through E (PA, PB, PC, PD, PE) and correspond to the five 10-pin RJ48 jacks, (RJA, RJB, RJC, RJD, RJE). RJA is at the top of the card and RJE is at the bottom, closest to the PCI connector. The four quadrature axes map to PA, PB, PC and PD while PE is an 8-bit general purpose IO port. The figure below shows how port signals map to RJ48 pins. This view is looking directly into a jack. The signal 5VF is 5 volts fused. The SEC-PCI limits external current to 2.0 amps via a resettable fuse. Either a standard 8-pin RJ45 plug or 10-pin RJ48 plug will work with the SEC-PCI. When an 8-pin RJ45 plug is used, signal pairs (PA2,PA4), (PA5,PA6), (5VF,GND) and (PA0,PA3) correspond to standard twisted pair positions.

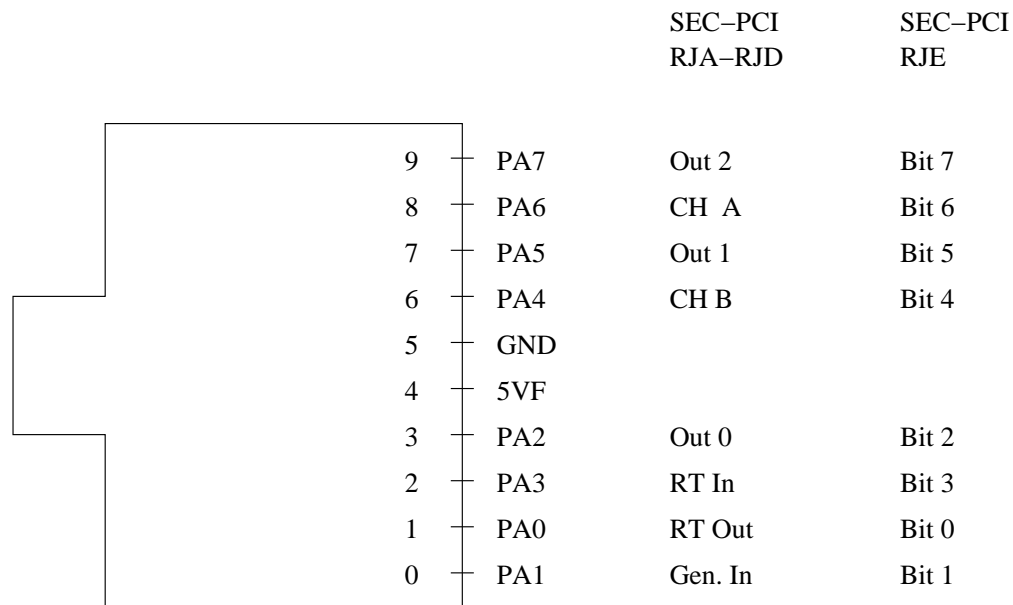


Figure 1: SEC-PCI RJ48 Connector

The SEC-PCI supports other types of external connectors. This requires a custom-made adapter or daughter board that connects to the 60-pin CN1, CN2 or CN3 positions on the SEC-PCI card. Differential quadrature input is also supported. Please contact Fischer Computer Systems if you have any custom requirements.

HARDWARE DESCRIPTION

The SEC-PCI hardware is implemented as a soft Field Programmable Gate Array (FPGA) core. This core is organized around the four quadrature axes. Figure 2 shows how a single quadrature axis interfaces to the external ports and to software. The left side identifies external digital signals. CH A and CH B are the two quadrature clocks. RT In and RT Out are the real-time input and real-time output signals respectively. Gen. In, Out 2, Out 1 and Out 0 are one generic input and three generic output signals. Each quadrature axis duplicates the block shown in figure 2 and routes these 8 signals to the appropriate RJA, RJB, RJC or RJD jack. Each quadrature axis also governs two signals from the 8-bit RJE general purpose I/O port. For example, quadrature axis 0 governs PE I/O bits 0 and 1 ($2 \times 0 = 0$ and $2 \times 0 + 1 = 1$). Likewise, quadrature axis 3 governs PE I/O bits 6 and 7 ($2 \times 3 = 6$ and $2 \times 3 + 1 = 7$). Finally, the right side of figure 2 shows how a user communicates with the FPGA core at the lowest level. A configuration packet may be sent, a notification packet may be received and the count value may be read.

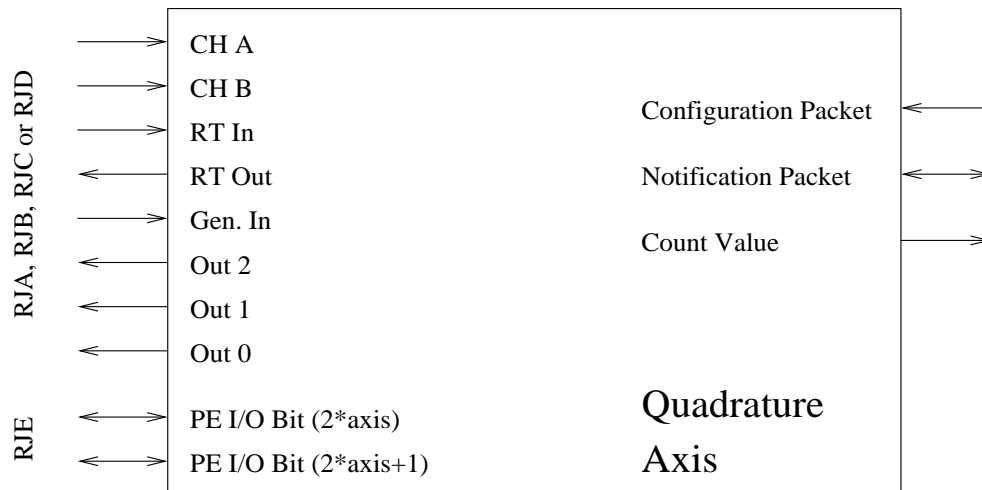


Figure 2: Conceptual Diagram of a Quadrature Axis

Since the SEC-PCI hardware is implemented as a soft FPGA-core, it is easily customized. SEC-PCI hardware updates may be sent as a single file (secpci.bin) and downloaded to the SEC-PCI even while the card is at location and in service. Please contact Fischer Computer Systems if you desire a customized SEC-PCI core or wish to target some other reconfigurable hardware application.

PROGRAMMER'S PERSPECTIVE

SEC-PCI programming may be divided into three tasks:

- Read a count packet
- Send a command packet
- Receive a notification packet

Count Packet

The first task, read a count packet, requires reading one of four memory-mapped locations. Each memory location corresponds to a single quadrature axis as shown in table 1. The value read at this location is as specified in table 2. Count packets are updated automatically as soon as any information changes. In typical applications, an API library call or specific driver IOCTL call is used to read and parse a count packet. If low-level control is desired, device memory read/write and device memory remapping to user space may be done via driver IOCTL calls.

Address	Description
0x4040	Axis 0 count packet
0x4044	Axis 1 count packet
0x4048	Axis 2 count packet
0x404c	Axis 3 count packet

Table 1: Count Packet Memory Locations

Bit	Description
31::29	Unused
28	Axis current real-time output value
27	Axis current real-time input value
26	Axis generic input value
25	Port E bit ($2 \times \text{axis} + 1$)
24	Port E bit ($2 \times \text{axis}$)
23::0	24 bit current count

Table 2: Count Packet Structure

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

Command Packet

The SEC-PCI core is configured by sending a command packet. A command packet is sent by writing to memory location 0x0040. This places the command packet in a hardware FIFO buffer that is read by the SEC-PCI core. A command packet is defined in table 3 and consists of a command code, axis code and data packet. The data packet is interpreted differently for each command code.

Bit	Description
31::30	Axis codes 00 = axis 0 01 = axis 1 10 = axis 2 11 = axis 3
29::24	Command codes 000000 = IO configure 000001 = filter configure 000010 = decode configure 000011 = set compare value 000100 = real-time response configure 000101 = set preset value 000110 = notify configure 000111 = set count value
23::0	Data packet

Table 3: Command Packet Structure

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

0x0 – IO Configure

The IO configure, command defined in table 4, sets the direction of the entire general purpose 8-bit IO port PE/RJE, sets output values for two bits of PE/RJE as well as sets output values for the three axis generic outputs out 2, out 1 and out 0. By factory default, all PE/RJE IO bits must be configured as either input or output. A factory modification is available so that some subset of bits may be defined as input and the rest as output. Two PE/RJE bits are assigned to each axis. If a PE/RJE output bit is configured, its value is stored in internal SEC-PCI memory. Consequently, if PE/RJE is changed to input and then to output, the stored output bit settings are restored. Setting the IO port direction for one axis is global and effects all other axes.

Bit	Description
5	Port E direction 0 = input 1 = output
4	Port E bit ($2 \times \text{axis} + 1$) output value
3	Port E bit ($2 \times \text{axis}$) output value
2	Axis generic out 2
1	Axis generic out 1
0	Axis generic out 0

Table 4: IO Configure Data Packet Structure

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

0x1 – Filter Configure

The SEC-PCI digital filter samples axis inputs CH A, CH B and RT In. A specified number of samples must agree on average before this input value is considered valid. The sampling frequency is based on the PCI clock which is usually 33.333 MHz. Each axis is sampled in round-robin fashion. Hence, the axis sampling frequency is 4.167 MHz or roughly 240 ns. The maximum possible count rate is 4 MHz for $\times 4$ quadrature decoding with 2 filter samples. If only 1 filter sample is specified, then no filtering takes place. This filter setting is included for fast up/down count applications only. If a quadrature axis is not being used, its filter sampling rate should be set to maximum, 16 samples, to prevent noise counting.

Bit	Description	Minimum Stable Period	Maximum Quadrature Frequency	Max $\times 1$ Count Rate	Max $\times 2$ Count Rate	Max $\times 4$ Count Rate
3::0	0=1 sample	240 ns	Rate error	2 MHz ¹	Rate error	Rate error
	1=2 samples	480 ns	1 MHz	1 MHz	2 MHz	4 MHz
	2=3 samples	720 ns	694 kHz	694 kHz	1.39 MHz	2.77 MHz
	3=4 samples	960 ns	521 kHz	521 kHz	1.04 MHz	2.08 MHz
	4=5 samples	1.2 us	417 kHz	417 kHz	834 kHz	1.66 MHz
	5=6 samples	1.44 us	347 kHz	347 kHz	694 kHz	1.39 MHz
	6=7 samples	1.68 us	297 kHz	297 kHz	594 kHz	1.19 MHz
	7=8 samples	1.92 us	260 kHz	260 kHz	520 kHz	1.04 MHz
	8=9 samples	2.16 us	231 kHz	231 kHz	462 kHz	924 kHz
	9=10 samples	2.4 us	208 kHz	208 kHz	416 kHz	832 kHz
	10=11 samples	2.64 us	189 kHz	189 kHz	378 kHz	756 kHz
	11=12 samples	2.88 us	173 kHz	173 kHz	346 kHz	692 kHz
	12=13 samples	3.12 us	160 kHz	160 kHz	320 kHz	640 kHz
	13=14 samples	3.36 us	149 kHz	149 kHz	298 kHz	596 kHz
	14=15 samples	3.6 us	139 kHz	139 kHz	278 kHz	556 kHz
	15=16 samples	3.84 us	130 kHz	130 kHz	260 kHz	520 kHz

Table 5: Filter Configure Data Packet Structure

¹ up/down mode only

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

0x2 – Decode Configure

The decode configure command sets the quadrature decode mode as well as the real-time input edge sensitivity, RT In.

Bit	Description
2	Axis input event edge sensitivity 1 = rising 0 = falling
1::0	Decode mode 00 = up/down 01 = quadrature ×1 10 = quadrature ×2 11 = quadrature ×4

Table 6: Decode Configure Data Packet

0x3 – Set Compare Value

The set compare value command sets an axis compare register to the specified 24 bit value.

Bit	Description
23::0	Compare value

Table 7: Set Compare Value Data Packet

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

0x4 – Real-Time Response Configure

The SEC-PCI may be configured to respond in real-time to certain events. Setting the appropriate bit as described in table 8 will cause the corresponding real-time response for the listed event. The clear response takes precedence over the preset response if both are set. A real-time output refers to a signal change response on axis signal RT Out. Any real-time response configuration command will always set or clear RT Out via bit 4.

Bit	Description
11	Clear count value if count value equals compare value
10	Clear count value if real-time input event occurs
9	Preset count value if underflow occurs
8	Preset count value if overflow occurs
7	Preset count value if count value equals compare value
6	Preset count value if real-time input event occurs
5	Real-time output mode 1 = one shot toggle 0 = multiple toggle
4	Real-time output preset value 1 = Set 0 = Clear
3	Real-time output if underflow occurs
2	Real-time output if overflow occurs
1	Real-time output if count value equals compare value
0	Real-time output if real-time input event occurs

Table 8: Real-Time Response Configure Data Packet

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

0x5 – Set Preset Value

The set preset value command sets an axis preset register to the specified 24 bit value.

Bit	Description
23::0	Compare value

Table 9: Set Preset Value Data Packet

0x6 – Notify Configure

The notify configure command specifies for what reasons the SEC-PCI core should send a notification packet. Setting a bit as specified in table 10 implies notify request, clearing a bit implies notify disable. Once a notify packet has been sent in response to some event, the SEC-PCI clears the request bit. Hence, the user typically sets the notify request bit after handling a notify packet. Only real-time input events may trigger multiple notification events. Real-time input events (RT In) are also the only event that captures the count value. Allowing multiple real-time events allows multiple count captures. A rate error occurs when both quadrature signals change simultaneously. This implies noise on the channels or excessive quadrature frequencies. Finally, four of port E's IO bits may be configured to trigger a notification packet. This is helpful when switch input is desired.

Bit	Description
7	Generic Port E input edge (port E bit 2 × axis when input) 1 = notify on rising edge 0 = notify on falling edge
6	Multiple real-time input event notifications
5	Notify on generic input (port E bit 2 × axis when input)
4	Notify on rate error
3	Notify on underflow
2	Notify on overflow
1	Notify if count value equals compare value
0	Notify on real-time input event

Table 10: Notify Configure Data Packet

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

0x7 – Set Count Value

The set count value command sets an axis count register to the specified 24 bit value.

Bit	Description
23::0	Count value

Table 11: Set Count Value Data Packet

Notification Packet

The SEC-PCI core communicates to the user through notification packets. A notification packet is read at address 0x0044. This location represents a 32-packet-deep FIFO buffer. If no packet is pending, 0xffffffff will be read. A user may check for notification packets by polling or by interrupt. The secpciapi.dll library and IOCTL have facilities for interrupt-driven notification. When decoding the notification packet, only notifications which are pending and have not occurred will have a bit set. Hence, detecting the actual notification event requires a logical XOR with the last known notification state. If multiple real-time input event notifications are enabled, the real-time input event pending flag will toggle.

Bit	Description
31::30	Axis 00 = axis 0 01 = axis 1 10 = axis 2 11 = axis 3
29	Generic port E input notification pending
28	Rate error notification pending
27	Underflow notification pending
26	Overflow notification pending
25	Count value equals compare value notification pending
24	Real-time input event notification
23::0	Last axis capture value (capture at every real-time input)

Table 12: Notification Packet Structure

API LIBRARY

The SEC-PCI API library provides a simplified and operating system independent programming interface to the SEC-PCI. The library's C source code is freely available and is considered the full reference documentation. Library function calls are listed and briefly described below. The SEC-PCI library supports up to four SEC-PCI cards.

```
void SecReset(int card);
```

Reset the SEC-PCI's PCI interface ASIC and FPGA. The SEC-PCI library supports up to four SEC-PCI cards. The argument `card` specifies the SEC-PCI card a library function should act on. Valid values for argument `card` are 0, 1, 2 or 3.

```
int SecIPLoad(char *filename, int card);
```

Download the appropriate soft core to the SEC-PCI's FPGA. Typical quadrature decode/count applications must always initialize the SEC-PCI by calling:

```
int status = SecIPLoad("\\systemroot\\system32\\drivers\\secpci.bin",0);
```

The file "\\systemroot\\system32\\drivers\\secpci.bin" is the default SEC-PCI soft core and is installed on your system during driver installation. The returned `int` will be 0 if an error occurred. A call to `SecIPLoad()` will also reset the SEC-PCI's PCI interface ASIC and FPGA before loading the specified core.

```
int SecDefaultIPLoad(int card);
```

Download the default driver soft core, "\\systemroot\\system32\\drivers\\secpci.bin", as described above in `SecIPLoad`.

```
int SecInit(int card);
```

Initializes the SEC-PCI by loading the default core, clearing all count values, setting decode to x4 quadrature mode, setting the filter to ~500 kHz and setting PortE to input. This emulates the SEC-PC default settings.

```
int SecPeek(int addr, int card);
```

Read a value from the specified SEC-PCI memory map address `addr`.

```
void SecPoke(int addr, int val, int card);
```

Write `val` to the SEC-PCI memory map address `addr`.

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

```
char SecReadEEPROM(char addr, int card);
```

Read a value from the specified SEC-PCI EEPROM address `addr`. The EEPROM is byte addressed and stores 256 bytes. The first 128 bytes (locations 0x00–0x7f) are read-only and contain factory-set information. Locations 0x00–0x1f contain 32 bytes of factory-set customer specific data or characters. Locations 0x20–0x7f are reserved. The last 128 bytes (locations 0x80–0xff) are user readable and writable.

```
void SecWriteEEPROM(char addr, char val, int card);
```

Write value `val` to SEC-PCI EEPROM address `addr`. Only locations 0x80 to 0xff are writable. A write to an address less than 0x80 will be interpreted as `addr = 0x80 + addr`.

```
int SecCards();
```

Return the number of SEC-PCI cards found in a system.

```
int SecCardStatus(int card);
```

Return the status code for SEC-PCI card. Status codes are defined in the library header file.

```
void SecCommand(int command, int data, int axis, int card);
```

Send a low-level command packet as described in the Programmer's Perspective section.

```
int SecGetCount(int axis, int card);
```

Read the current count value for a particular axis and card. The returned 24-bit count value is sign extended.

```
void SecSetCount(int val, int axis, int card);
```

Set the count value for axis to `val`.

```
int SecGetPreset(int axis, int card);
```

Return the current preset register value for axis.

```
void SecSetPreset(int val, int axis, int card);
```

Set the preset register value for axis to `val`.

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

```
int SecGetCompare(int axis, int card);
```

Return the current compare register value for axis.

```
void SecSetCompare(int val, int axis, int card);
```

Set the compare register value for axis to val.

```
int SecGetFilterSamples(int axis, int card);
```

Return the current filter samples setting for axis.

```
void SecSetFilterSamples(int val, int axis, int card);
```

Set the filter samples for axis to val. The argument val may range from 1 to 16. See table 5 for more details.

```
int SecGetDecode(int axis, int card);
```

Return the current decode configuration for axis.

```
void SecSetDecode(int val, int axis, int card);
```

Configure decode for axis to val. See table 6 for appropriate val.

```
int SecGetResponse(int axis, int card);
```

Return the current response configuration for axis.

```
void SecSetResponse(int val, int axis, int card);
```

Configure response for axis to val. See table 8 for appropriate val.

```
int SecGetNotify(int axis, int card);
```

Return the current notify configuration for axis.

```
void SecSetNotify(int val, int axis, int card);
```

Configure notify for axis to val. See table 10 for appropriate val.

```
void SecGetPortE(int val, int card);
```

Set PortE to output if it is currently set to input. Drive PortE with val.

```
int SecGetPortE(int card);
```

Set PortE to input if it is currently set to output. Return the value read on PortE.

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

```
int SecGetIO(int axis, int card);
```

Return the current IO value for *axis*. The returned value is a byte as shown below.

Bit	Description
7	Axis current real-time output value
6	Axis current real-time input value
5	Axis generic input value
4	Port E bit ($2 \times \text{axis} + 1$)
3	Port E bit ($2 \times \text{axis}$)
2	Axis generic out 2
1	Axis generic out 1
0	Axis generic out 0

```
void SecSetIO(int val, int axis, int card);
```

Set the lowest 5 bits as described in table 4 for *axis* to *val*. Port E direction is not altered.

```
int SecGetPortEDir(int card);
```

Return the current port E direction setting.

```
void SecSetPortEDir(int val, int card);
```

Set port E direction to *val*. The argument *val* must be 0 for input and 0x20 for output.

```
void SecSetPortABCD8(int val, int card);
```

Combines generic out bits 1 and 0 from Ports A,B,C,D into a single 8-bit port. This port is ordered most significant bit first PD5, PD2, PC5, PC2, PB5, PB2, PA5, PA2. This port is driven with *val*.

```
void SecSetPortABCD4(int val, int card);
```

Combines generic out bit 2 from Ports A,B,C,D into a single 4-bit port. This port is ordered most significant bit first PD7, PC7, PB7, PA7. This port is driven with *val*.

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

```
int SecNotificationPoll(unsigned long interval,
                       unsigned int requests, int card);
```

Poll for a notification packet. Returns a notification packet as described in table 12. The argument `requests` specifies the requested number of polls. Set `requests` to `SEC_BLOCKINGPOLL` if this function call should block until a notification packet is received. The argument `interval` specifies the time in milliseconds between polls. For example, `SecNotificationPoll(0,0,0);` will poll the notification packet FIFO immediately and return the packet if successful or `0xffffffff` if unsuccessful. `SecNotificationPoll(10,20,0);` will poll the notification packet FIFO 20 times at intervals of 10 milliseconds. If a poll is successful, the notification packet will be returned immediately. If all requested polls have been made and no notification packet is received, `0xffffffff` is returned. Finally, `SecNotificationPoll(0,SEC_BLOCKINGPOLL,0);` will not return until a notification packet is received. `SEC_BLOCKINGPOLL` is interrupt-driven and hence the `interval` argument has no effect.

```
void SecNotificationHandler(void (*handler)(int),
                           int card);
```

The passed function handle is called when a notification packet is received. `SecNotificationHandler()` creates a new thread and returns immediately. The new thread calls `handler` when the notification packet is received via interrupt notification. At the end of a successful handler call, another call to `SecNotificationHandler()` must be made to reinstate handler. Only one handler may be registered and active at any given time. The programmer is responsible for maintaining thread safety. A handler function must return `void` and accept a single `int` argument. The notification packet is passed to the handler via this `int` argument. Examples appears below.

```
void testhandler(int packet) {
    printf("Test handler called with packet %x\n",packet);
}

SecNotificationHandler(*testhandler, 0);
```

DRIVERS AND IOCTL CALLS

The SEC-PCI has drivers for Windows 98/NT/2000 and Linux. Refer to the README.TXT file included on the distribution disk for detailed driver installation instructions. On Windows 98/2000, the PNP operating system will detect new hardware. Let Windows 98/2000 search for a suitable driver. Make sure the driver floppy is present and the search path includes the floppy.

If desired, driver IO control (IOCTL) calls may be used to interface to the SEC-PCI. In Windows, a SEC-PCI device may be opened with,

```
Handle = CreateFile("\\\\.\\secpci-0",
                   GENERIC_READ | GENERIC_WRITE,
                   FILE_SHARE_READ | FILE_SHARE_WRITE,
                   0,
                   OPEN_EXISTING,
                   FILE_FLAG_OVERLAPPED,
                   0);
```

The device names `secpci-0`, `secpci-1`, `secpci-2`, `secpci-3` refer to up to four SEC-PCI cards in the system. Once open, Win32 API function `DeviceIoControl()` is used to perform IOCTL operations on a device.

```
result = DeviceIoControl(Handle,
                          Code,
                          InputData,
                          InputLength,
                          OutputData,
                          OutputLength,
                          &Feedback,
                          &Overlapped);
```

If a device is opened with `FILE_FLAG_OVERLAPPED` for asynchronous operation, then a valid `Overlapped` structure must be passed by reference with every `DeviceIoControl()` call. `FILE_FLAG_OVERLAPPED` must be used if interrupt-driven notification is desired. Listed next are defined IOCTL codes and `InputData`, `OutputData` structures.

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

```
IOCTL_SEC_PEEK
ULONG addr; // InputData
ULONG val; // OutputData
```

Read value at SEC-PCI memory location.

```
IOCTL_SEC_POKE
typedef struct {
    ULONG addr;
    ULONG val;
} sec_poke; // InputData
// OutputData undefined
```

Write value at SEC-PCI memory location.

```
IOCTL_SEC_COMMAND
typedef struct {
    ULONG command;
    ULONG data;
    ULONG axis;
} sec_command; // InputData
// OutputData undefined
```

Send a low-level command packet as described in the Programmer's Perspective section.

```
IOCTL_SEC_NOTIFY
// InputData undefined
// OutputData undefined
```

Creates an IRP which pends until a notification packet is received. This is interrupt driven. Refer to the secpciapi.c source file for an example use of this IOCTL.

```
IOCTL_SEC_IPLOAD
typedef struct {
    WCHAR filename[64];
} sec_ipload; // InputData
// OutputData undefined
```

Download the appropriate soft core to the SEC-PCI's FPGA. Typical quadrature decode/count applications must always initialize the SEC-PCI with file "\\systemroot\\system32\\drivers\\secpci.bin." This file is the default SEC-PCI soft

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

core and is installed on your system during driver installation. This IOCTL call will also reset the SEC-PCI's PCI interface ASIC and FPGA before loading the specified core.

```
IOCTL_SEC_MMAP
// InputData undefined
ULONG memorymap; // OutputData
```

Return a user mode pointer to the SEC-PCI's memory map. Refer to the `secpciapi.c` source file for an example use of this IOCTL.

```
IOCTL_SEC_MUNMAP
ULONG memorymap; // InputData
// Outputdata undefined
```

Release a user mode pointer to the SEC-PCI's memory map. This should be called as program exit clean-up when `IOCTL_SEC_MMAP` has been used. Refer to the `secpciapi.c` source file for an example use of this IOCTL.

```
IOCTL_SEC_GETPACKET
// InputData undefined
typedef struct {
    ULONG axisa;
    ULONG axisb;
    ULONG axisc;
    ULONG axisd;
} sec_packet; // OutputData
```

Read count packets for all four axes. Refer to table 2 for count packet structure.

```
IOCTL_SEC_POLL
// InputData undefined
ULONG pollresult; // OutputData
```

Performs a single poll of the notification packet FIFO. Returns `0xffffffff` if no notification packet is present. Refer to table 12 for notification packet structure.

```
IOCTL_SEC_RESET
// InputData undefined
// OutputData undefined
```

Reset the SEC-PCI's PCI interface ASIC and FPGA.

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

```
IOCTL_SEC_RDEEPROM
char addr; // InputData
char val; // OutputData
```

Read a single byte from the SEC-PCI's EEPROM. Refer to library call description `SecReadEEPROM()` for more EEPROM details.

```
IOCTL_SEC_WREEPROM
typedef struct {
    char addr;
    char val;
} sec_wreeprom; // InputData
// OutputData undefined
```

Write a single byte to the SEC-PCI's EEPROM. Refer to library call description `SecWriteEEPROM()` for more EEPROM details.

DEMONSTRATION PROGRAM

A comprehensive GUI SEC-PCI demonstration program is available. For Windows installation, copy the entire directory `a:\sepcidemo` on the distribution disk to a suitable location on your hard drive. Execute the program `sepcidemo.exe` found in that directory. The directory `a:\src\sepcidemo` contains a Microsoft Visual C++ 6.0 project and source code for the `sepcidemo` executable. A few project include and library paths will have to be modified for your local environment. To compile this project, you will need the wxWindows cross-platform GUI library available open source at: www.wxwindows.org.

The `sepcidemo` program demonstrates all functionality described in the Programmer's Perspective section. When a particular configuration is made, the appropriate library call is displayed in the log window. This text output may be copied and pasted into other files. To set particular count, compare or preset value, double-click on the particular value. The Programmer's Perspective section aids understanding and use of the `sepcidemo` program.

SAMPLE CONFIGURATIONS

Configuration 1: Clear via Index Signal

A quadrature encoder with index signal is connected to the SEC-PCI's axis 2. The axis 2 count must be cleared on a falling edge of the index signal. Quadrature ×4 decoding is desired. To do this, connect the index signal to the axis 2 real-time input signal. Execute the code,

```
SecSetDecode(SEC_DECODE_INEVENT_FALL | SEC_DECODE_X4,  
             2, 0);  
SecSetResponse(SEC_RESPONSE_CLEAR_ON_INEVENT, 2, 0);
```

Configuration 2: Real-Time Count Capture by Switch

Count values for axes 1 and 2 must be captured in real-time on the falling edge of a foot switch signal. Quadrature ×4 decoding is desired. To do this, connect the pulled-high foot switch signal to real-time input signals for axes 1 and 2. (This modification may be done at the factory with appropriate jumpers on CN2.) Execute the code,

```
SecSetDecode(SEC_DECODE_INEVENT_FALL | SEC_DECODE_X4,  
             1, 0);  
SecSetDecode(SEC_DECODE_INEVENT_FALL | SEC_DECODE_X4,  
             2, 0);  
SecSetNotify(SecGetNotify(1, 0) | SEC_NOTIFY_INEVENT,  
             1, 0);  
SecSetNotify(SecGetNotify(2, 0) | SEC_NOTIFY_INEVENT,  
             2, 0);
```

Notification packets for both axes will be sent when the foot switch is pressed. These may be read via polling or an interrupt-driven notification handler. As foot-switch events are often noisy and occur relatively infrequently, the multiple input event notification flag should not be used. Rather, the user should reinstate the desired notification with repeated calls to SecSetNotify(). Notice that SecSetNotify() sets all notification events. Consequently, it is desirable to read the current notify configuration and set the desired flags through logical OR.

Configuration 3: Notification by Port E Switch Event

A user desires notification when a rising edge occurs on port E bit 2. Execute the code,

```
SecSetNotify(SecGetNotify(1, 0) | SEC_NOTIFY_GIN_RISE  
             SEC_NOTIFY_GIN,  
             1, 0);
```

Configuration 4: Limit Count Range from 0 to 4789

A user desires axis 0 to only count in the range of 0 to 4789. To do this, set the compare register to 4790. Set a response to clear the count value when the compare is true. Set the preset register to 4789. Set a response preset on underflow. The following code implements this.

```
SecSetCompare(4790, 0, 0);  
SecSetPreset(4789, 0, 0);  
SecSetResponse( SEC_RESPONSE_PRESET_ON_UFLOW |  
                SEC_RESPONSE_CLEAR_ON_MATCH,  
                0, 0);
```

VISUAL BASIC PROGRAMMING

The "Software Source Code and Examples Disk", available at www.sepc.com, contains a Visual Basic/SEC-PCI example project in subdirectory `src\secpci.vb`. Use this example project as a guide for your own Visual Basic applications.

The file `secpciapi.bas` contains the required declarations to permit access to `secpciapi.dll` C functions from within Visual Basic. You must include this file as a module in your Visual Basic application. All function calls described in the API Library section on page 13 are available in Visual Basic except for `SecIPLoad`, `SecPeek`, `SecPoke` and `SecNotificationHandler`. Use `SecDefaultIPLoad` to initialize the SEC-PCI. Check for notification packets via polling only with `SecNotificationPoll`.

```
'Example SEC-PCI Initialization code
Private Sub Form_Load()

'Load FPGA Core
Call SecDefaultIPLoad(0)

'Set mode to x4 decode for all axis
Call SecSetDecode(3, 0, 0)
Call SecSetDecode(3, 1, 0)
Call SecSetDecode(3, 2, 0)
Call SecSetDecode(3, 3, 0)

'Set filter to 500 kHz like SEC-PC
Call SecSetFilterSamples(3, 0, 0)
Call SecSetFilterSamples(3, 1, 0)
Call SecSetFilterSamples(3, 2, 0)
Call SecSetFilterSamples(3, 3, 0)

'Set all count values to 0
Call SecSetCount(0, 0, 0)
Call SecSetCount(0, 1, 0)
Call SecSetCount(0, 2, 0)
Call SecSetCount(0, 3, 0)

End Sub
```

SEC-PC TO SEC-PCI PORTING

Porting applications from the SEC-PC (ISA card) to the SEC-PCI is not a one-to-one mapping because of the SEC-PCI's additional features. Although recommendations for porting your applications and wiring from the SEC-PC to the SEC-PCI are given below, you are encouraged to make use of the advanced SEC-PCI features such as real time inputs/outputs, notification via interrupt, count capture, count compare, count preset, etc. The most significant differences are with IO bits (foot switches, mouse buttons, etc.) The SEC-PCI's 16 output, 8 input and 8 input/output bits are distributed across all 5 RJ-48 jacks and their functionality should be defined by the end user. To emulate the SEC-PC IO with the SEC-PC, use the following suggestions:

- The SEC-PC's 8-bit parallel input port (PI) OR parallel output port (PO) may be emulated with the SEC-PCI's bidirectional PortE. The SEC-API's `SecGetPortE` and `SecSetPortE` functions are used in place of the SEC-PC's `SecPortIn` and `SecPortOut` functions respectively. If your SEC-PC application uses both the parallel input and output ports, then use the SEC-PCI's PortE as your parallel input port and all four axes generic output bits 0 and 1 as your parallel output port. The SEC-PCI's `SecGetPortE` and `SecSetPortABCD8` functions are used.
- The SEC-PC's 4-bit LED output port (LED) may be emulated with the SEC-PCI's bidirectional PortE. The SEC-API's `SecPortEOut` function is used in place of the SEC-PC's `SecLed` function. If the SEC-PCI's PortE is already in use, use all four axes generic output bits 2 as your LED output port. The SEC-PCI's `SecSetPortABCD4` function is used. Any ground signal will suffice as an LED return signal.
- The SEC-PC's switch inputs (Switch Common, Switch NC, Switch NO) may be emulated with the an SEC-PCI's axis real-time input bit (RT In). For example, use Port A's RT In as a switch input (either to Switch NC or Switch NO) and ground (Switch Common). For simple switch polling, use the SEC-PCI's `SecGetIO` instead of the SEC-PC's `SecPoll`. Mask the value from `SecGetIO` with 0x40 to read the status of the RT Input. Unlike the SEC-PC, the SEC-PCI does not latch the value of the switch. Therefore there is no need to clear or set the switch flag as done with the SEC-PC's `SecPoll` and `SecFlag`. Instead use the SEC-PCI's `SecSetNotify` to setup a notification event when the switch changes. Finally, applications may use PortE bits (4 bits on PortE will generate notification events) and axis generic input bits as alternate switch inputs.

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

<i>SEC-PC Signal</i>	<i>SEC-PCI Signal</i>
Switch Common	GND
Switch NC	PB3 Port B's RT In
Switch NO	PA3 Port A's RT In alternatives: PD3, PB3, PC3, PA1, PB1, PC1, PD1, PE all
PI-0 to PI-7	PE0-PE7 PortE bidirectional port
PO-0 to PO-7	PE0-PE7 PortE bidirectional port alternative: PD5,PD2,PC5,PC2,PB5,PB2,PA5,PA2 (Outs 1,0)
LED-3 to LED-6	PE1, PE3, PE5, PE7 alternatives: PD7, PC7, PB7, PA7 (Outs 2)
Return	GND
Encoder XA, XB	PA6, PA4 PortA CH A and CH B
Encoder YA, YB	PBC, PB4 PortB CH A and CH B
Encoder ZA, ZB	PC6, PC4 PortC CH A and CH B
Encoder WA, WB	PD6, PD4 PortD CH A and CH B
+5 Volts	5VF
Ground	GND

Table 13: SEC-PC to SEC-PCI Signal Mapping

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

To initialize the SEC-PCI to match the decoding and filtering of the SEC-ISA, execute the `SecInit` function. Refer to the source code for initialization details.

<i>SEC-PC DLL Function</i>	<i>SEC-PCI API Function</i>
<code>SecPortIn</code> , <code>SecPortOut</code>	<code>SecGetPortE</code> , <code>SecSetPortE</code> , <code>SecSetPortABCD8</code>
<code>SecClear</code>	for each axis: <code>SecSetCount</code>
<code>SecPoll</code>	(<code>SecGetIO(3,0)</code> & 0x40) true if PortD RT In used as switch
<code>SecGetReading</code>	<code>SecGetCount</code>
<code>SecFlag</code>	Use SEC-PCI notification to latch switch values
<code>SecFlagRd</code>	Combination of above two rows
<code>SecBase</code>	Hardware specific to SEC-PC, no SEC-PCI equivalent
<code>SecPresent</code>	<code>SecCards</code>
<code>SecLed</code>	<code>SecSetPortABCD4</code>
<code>SecRdRom</code> , <code>SecWrRam</code>	<code>SecReadEEPROM</code> , <code>SecWriteEEPROM</code>
<code>SecInByte</code> , <code>SecOutByte</code>	Hardware specific to SEC-PC, no SEC-PCI equivalent
<code>SecDllDiag</code>	<code>SecCardStatus</code>
No SEC-PC equivalent	<code>SecInit</code> , <code>SecReset</code> , <code>SecIPLoad</code> , <code>SecDefaultIPLoad</code> , <code>SecPeek</code> , <code>SecPoke</code> , <code>SecCommand</code> , <code>SecGetPreset</code> , <code>SecSetPreset</code> , <code>SecGetCompare</code> , <code>SecSetCompare</code> , <code>SecGetFilterSamples</code> , <code>SecSetFilterSamples</code> , <code>SecGetDecode</code> , <code>SecSetDecode</code> , <code>SecGetResponse</code> , <code>SecSetResponse</code> , <code>SecGetNotify</code> , <code>SecSetNotify</code> , <code>SecGetIO</code> , <code>SecSetIO</code> , <code>SecGetPortEDir</code> , <code>SecSetPortEDir</code> , <code>SecNotificationPoll</code>

Table 14: SEC-PC to SEC-PCI Library Function Mapping

SEC-PCI QUADRATURE COUNTER/DECODER

USER'S MANUAL

CONTACT INFORMATION

General Information:

Fischer Computer Systems
445 Bay Street
Angwin, CA 94508

Web: <http://www.secpc.com>

E-mail: info@secpc.com or support@fcs.net

Tel: 707-965-2414

Fax: 707-965-3687

Programming and Technical Information:

E-mail: steve@fcs.net