ICS-500 FPDP II PMC BUFFER MODULE WITH PCI 64/66 INTERFACE

No. 43 Rev.D Interactive Circuits & Systems Ltd.

INTRODUCTION

The ICS-500 is an FPDP II to PCI buffer board with a PCI 32-bit/64-bit 33/66 MHz bus interface, in a PCI Mezzanine Card (PMC) format. FPDP II is a 32-bit 400Mbyte/s data interface that is backwards compatible with the 160Mbyte/s ANSI/VITA 17 Front Panel Data Port interface. For more information about FPDP and FPDP II, visit the website www.FPDP.com.

The ICS-500 may be ordered in either FPDP receive or transmit formats. The FPDP receive version provides an FPDP/R- interface, in FPDP terminology, and may be configured under software control to enable or disable FPDP bus terminations; this allows multiple ICS-500R boards to be connected to the same FPDP cable. The transmit format module provides an FPDP/TM (Transmit Master) interface. When connected on an FPDP cable to an FPDP interface (i.e. the original 160Mbyte/s interface) either model reverts to FPDP protocol. This is done by means of a sensing mechanism included with the FPDP II design.

A key feature of the ICS-500R is its ability to perform the corner turning function. This is the ability to reorder multi-channel sensor data from channel ordering by time to time ordering by channel. The feature is commonly required in sensor processing systems and is usually performed by digital signal processing (DSP) elements. By transferring this function to the dedicated hardware located on the ICS-500, substantial additional DSP power is made available for other activities.

Figure 1 – The ICS-500 PMC module

The ICS-500 includes a “swing buffer” (ping-pong buffer) for data storage. The standard product has a capacity of 2 Mbyte. Optionally, it may be ordered with 8 Mbytes capacity.

The ICS-500 is ideally suited for all sensor processing applications where a high data rate and/or a large number of channels is required.
The ICS-500 may be used to buffer high speed FPDP / FPDP II data for applications in :

- Radar
- Sonar
- Digital Receivers
- Transient Analysis
- Experiments in Atomic Physics
- Ultrasound
- Test and Measurement

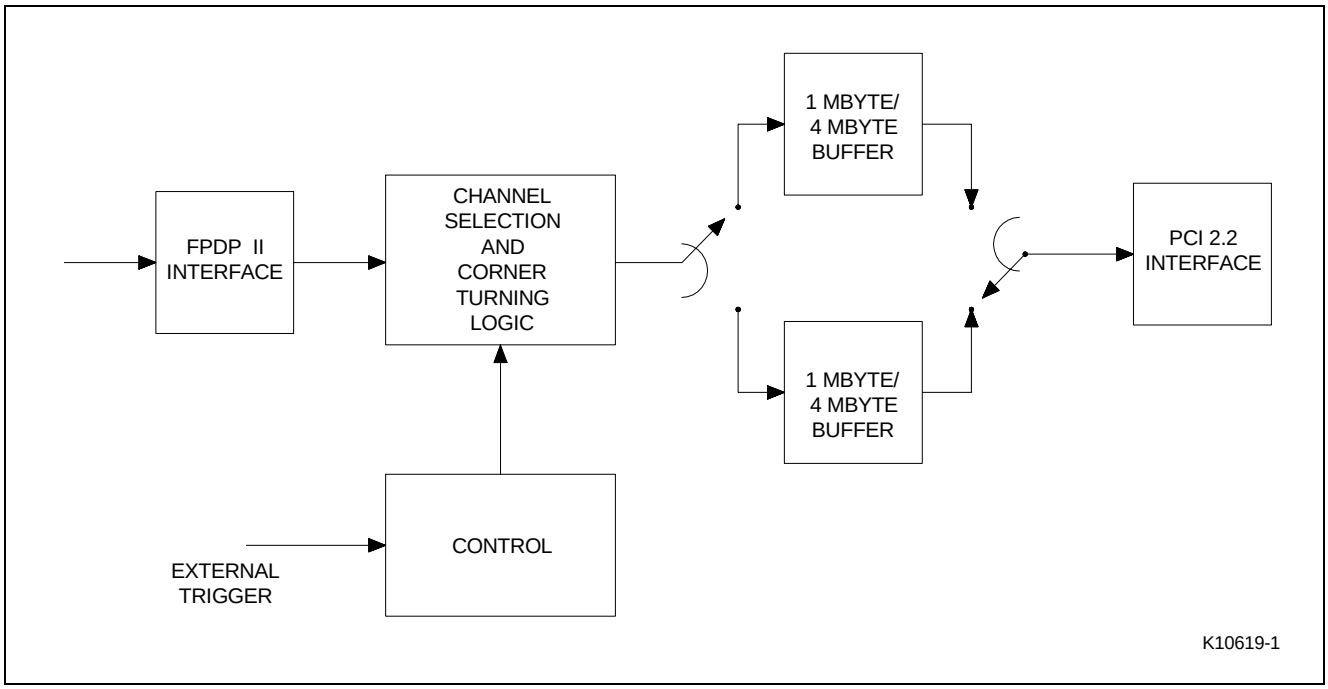

Figure 2 - ICS-500R FPDP Receiver Block Diagram

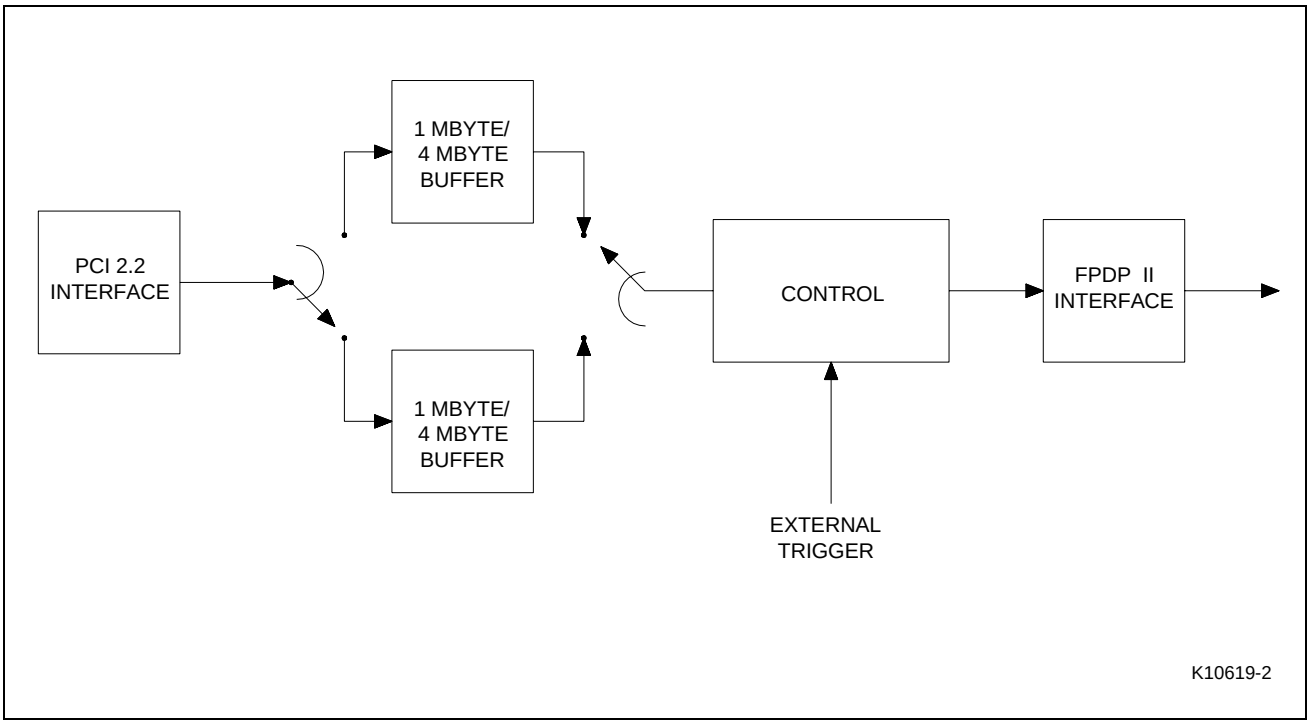

Figure 3 – ICS-500T Transmitter Block Diagram

The ICS-500 includes the following features:

- FPDP/TM or FPDP/R- versions;
- PCI 2.2 (64-bit, 66MHz) interface;
- 2 or 8 Mbyte on-board "swing buffer";
- Software-controlled channel selection and "corner turning" capabilities;
- Continuous and Capture buffer modes of data input (ICS-500R version);
- Continuous, One-shot and Loop modes of data output (ICS-500T version);
- Internal or external trigger.

ICS-500R VERSION DESCRIPTION

The ICS-500R provides an FPDP input function, also know as an FPDP/R- interface. Figure 2 shows a simplified block diagram of this version. The board can be configured for either the FPDP/RM (receiver master), or FPDP/R (receiver) functions under software control. When the data source is not an FPDP II interface, the behavior of the ICS-500R automatically reverts to FPDP specifications, with a maximum throughput of 160Mbytes/s. When connected to an FPDP II transmit interface, the full FPDP II throughput of 400Mbytes/s can be achieved. The product supports all defined operating modes of FPDP – Unframed Data, Single Frame Data, and Repeating Frame Data, Fixed and Dynamic. However, the Channel Selection and Corner Turning features are only available when using Fixed Length Repeating Frame Data mode.

Channel Selection and Corner Turning Features

Data received at the FPDP II interface is buffered in an 8Kbyte FIFO memory before application to the channel selection and corner turning logic. These features are designed to support sensor processing applications in which it is desirable to be able to select data for specific channels only, and to be able to reorder data from channel ordering into time series. To use the channel selection logic, the module must be programmed with the starting and ending channel numbers for the range of channels to be selected. Data for all channels within this range will be routed to the internal logic of the module while data for the other channels will be rejected. The corner turning logic may be enabled or disabled under software control. When corner turning is enabled, the selected channel data is reorganized from a channel sequence into a time series; this feature is illustrated in Figure 4. The length of the output sequence for each channel may be programmed, but it is limited by the size of buffer memory (1Mbyte or 4Mbytes maximum). Following channel selection and corner turning, data is stored in the swing buffer memory. From here it may be read over PCI Bus using DMA transfers.

Triggering

Processing of data starts when the board is triggered; until this occurs, all data presented at the FPDP II input will be ignored. Before the board is triggered, the FPDP Suspend Data signal will not be asserted, so the transmit interface will not be suspended from transmission of data. The trigger may be either internal (software enabled) or an external signal connected to the trigger input. When an external trigger signal is used, it may be configured by software to be either edge sensitive or level sensitive.

Operating Modes

The module has several modes of operation. In the Continuous mode, data is continuously routed to the PCI interface upon application of a trigger signal until the module is disabled, or until the level sensitive trigger is negated. In the Capture mode, a fixed number of words are acquired upon each application of the trigger. The number of words is determined by the value programmed to the buffer length register.

Sequence Number

The ICS-500R may be programmed to insert a sequence number in each buffer of output data. The sequence number occupies the first 64-bit word in the output buffer. The value weritten is a 32-bit number representing the number of 64-bit words that have been written to the buffer.

ICS-500T VERSION DESCRIPTION

The ICS-500T provides an FPDP output function, also know as an FPDP/TM interface. Figure 3 shows a simplified block diagram of this version. The features provided by this version are similar to those provided by the ICS-500R except that the flow of data is in the opposite direction. The channel selection and corner turning features are not provided. However, as an additional feature, the ICS-500T provides a Loop mode of operation, in which a fixed length of data equal to the programmed buffer length is written to both banks of the swing buffer. When triggered, this data is repetitively generated and transmitted by the FPDP output interface. The ICS-500T will respond to the assertion of the FPDP Suspend Data signal by suspending tranmission within 16 cycles of the Strobe signal. This is done by negating the Data Valid signal. When the Suspend Data signal is negated, transmission will be continued. The ICS-500T also allows the frame rate of transmitted data to be controlled, for applications where data must be transmitted at a specified sampling rate. This is done by programming of the FPDP Strobe (clock) frequency, which can be done over a wide frequency range and with fine resolution. The product supports all defined operating modes of FPDP except Dynamic Size Repeating Frame Data.

COMMON FEATURES

The following features are common to both ICS-500R and ICS-500T versions.

PCI Interface

The PCI interface of the ICS-500 board conforms to the PCI 64-bit, 66MHz (64/66) specification and uses universal signalling (3.3Volt or 5V). When used with 32-bit or 33 MHz PCI, it will revert to the specification of the host backplane. It provides fast transfer of data to the host processor using burst mode (DMA) bus cycles. The interface supports chain (linked list, or scatter-gather) DMA so that the swing buffer contents may be transferred to/from multiple buffers in host memory. Transfer rates of up to 528 Mbytes/s may be achieved when using a 64/66 backplane, depending on the capabilities of the host. The PCI interface circuitry uses the QuickLogic Q5064 interface chip. The ICS-500 board can generate PCI interrupts when the buffer is full (ICS-500R) or the buffer is empty (ICS-500T). The interface includes programmable reordering of data to support operation with “Big-endian” host computer systems.

Power Supply and Cooling

All power requirements of the ICS-500 are satisfied with standard 3.3V and 5V PMC power; total consumption is 6W. The user should pay attention to the thermal environment of the module and ensure an adequate supply of forced air cooling.

ORDERING OPTIONS

The ICS-500 board is offered in two versions: as a FPDP II receive interface (model ICS-500R) and as a FPDP II transmit interface (model ICS-500T). Software device drivers are available for Windows NT, 98, 2000, Me and Linux operating systems, and for the VxWorks operating system.

ICS-500R	FPDP II Receive / PCI buffer Card with 2Mbyte buffer
ICS-500R-E	FPDP II Receive / PCI buffer Card with 8Mbyte buffer
ICS-500T	PCI / FPDP II Transmit buffer Card with 2Mbyte buffer
ICS-500T-E	PCI / FPDP II Transmit buffer Card with 8Mbyte buffer

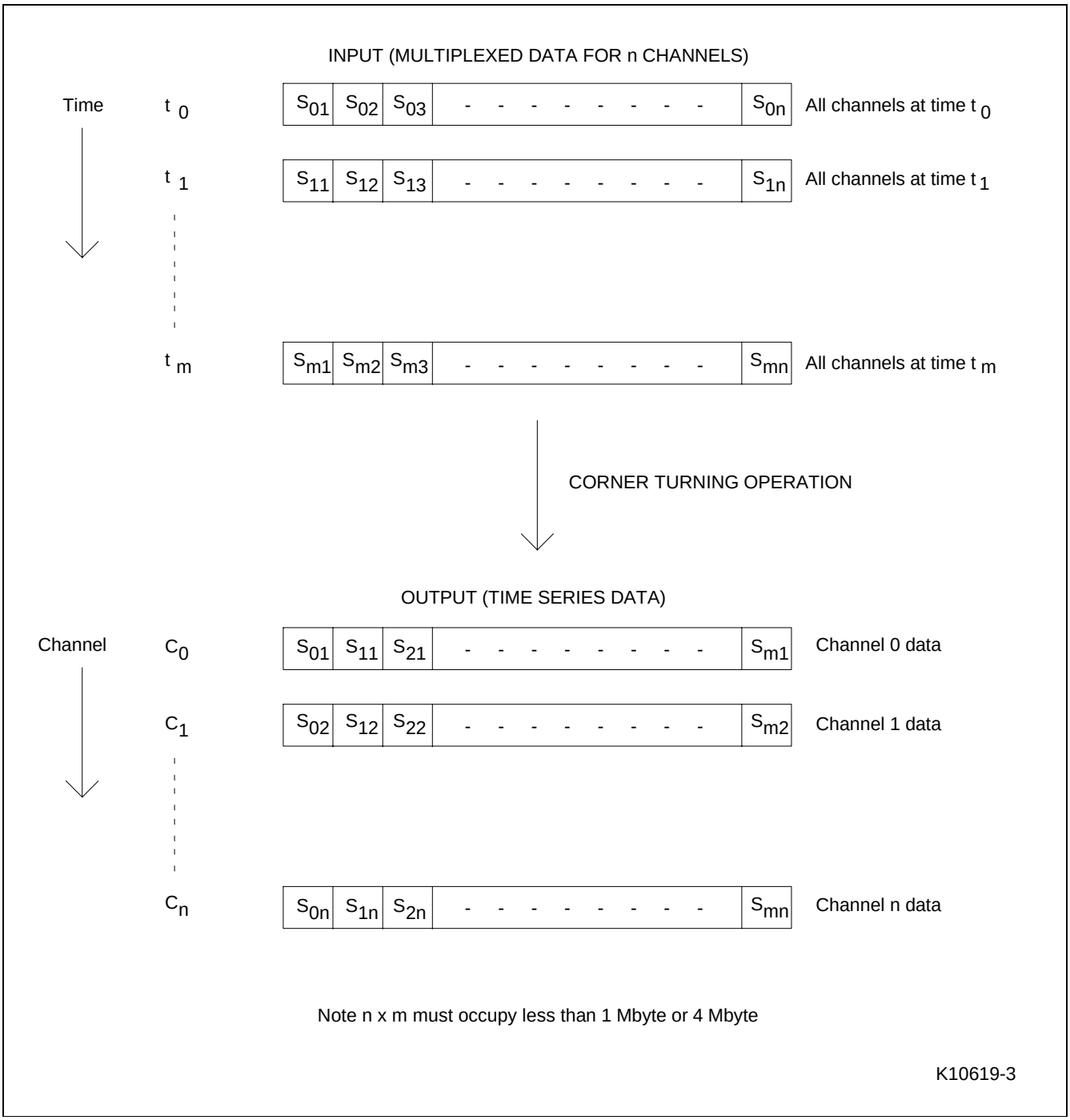

Figure 4 – Illustration of Data Reordering provided by Corner Turning Feature

ICS-500 SPECIFICATIONS

GENERAL

Buffer Memory	2 x 1Mbyte or 2 x 4Mbyte
Trigger	Internal or External edge triggered or level triggered

FPDP INTERFACE

Type	FPDP II Maximum strobe frequency 50MHz. Maximum throughput 400Mbytes/s. Reverts to ANSI/VITA 17 FPDP operation with non-FPDP II capable interfaces

PCI

PCI bus Interface:	PCI 2.2 compliant 64-bit, 66MHz Master/Target Capable of revertion to 64-bit, 33MHz or 32-bit, 33MHz 3.3V or 5V signalling (“universal”)
Environmental:	
Operating Temp:	0°- 50°C (at entry point of forced air)
Storage Temp:	-40° - +85°C
Humidity:	95% non-condensing
Cooling:	Approximately 490 LFM
Power:	+3.3V +5V Total power consumption 6W

PROGRAMMING MODEL

The ICS-500 PCI bus memory map is shown in Figure 5. All programming and control of the ICS-500 is accomplished through the PCI bus interface. All register bits that are not defined have no effect on the operation of the ICS-500, and may be read as zero or one.

SOFTWARE DEVICE DRIVERS

The ICS-500 board is supported by comprehensive software device drivers for the Windows NT, 98, 2000, Me and XP Professional operating systems, for Linux, and for the VxWorks real-time operating system.

The use of one of these drivers greatly simplifies control and operation of the ICS-500, and is strongly recommended. It will generally save programmers time in successfully implementing systems since they are relieved of the requirement to understand the complexities of the ICS-500 hardware model and of the Q5064 PCI bus interface device.

All drivers include the following elements:

(i) A device driver conforming to the I/O model for the target operating system, including support for open, close, read, write and ioctl functions, as well as device initialization and attachment.

(ii) A comprehensive application programming interface (API) containing approximately 20 “C” language functions. These provide, in most cases, access to individual device registers, thus providing control of all aspects of the ICS-500.

(iii) Sample application code in “C” showing how to program the ICS-500. The can be used as a template for development of the user’s application.

The API functions allow the user to quickly program applications using meaningful function names rather than the low-level "ioctl" functions. For users who wish to customize or expand the library routines, and for those who wish to port the driver to other operating systems, complete source code is available.

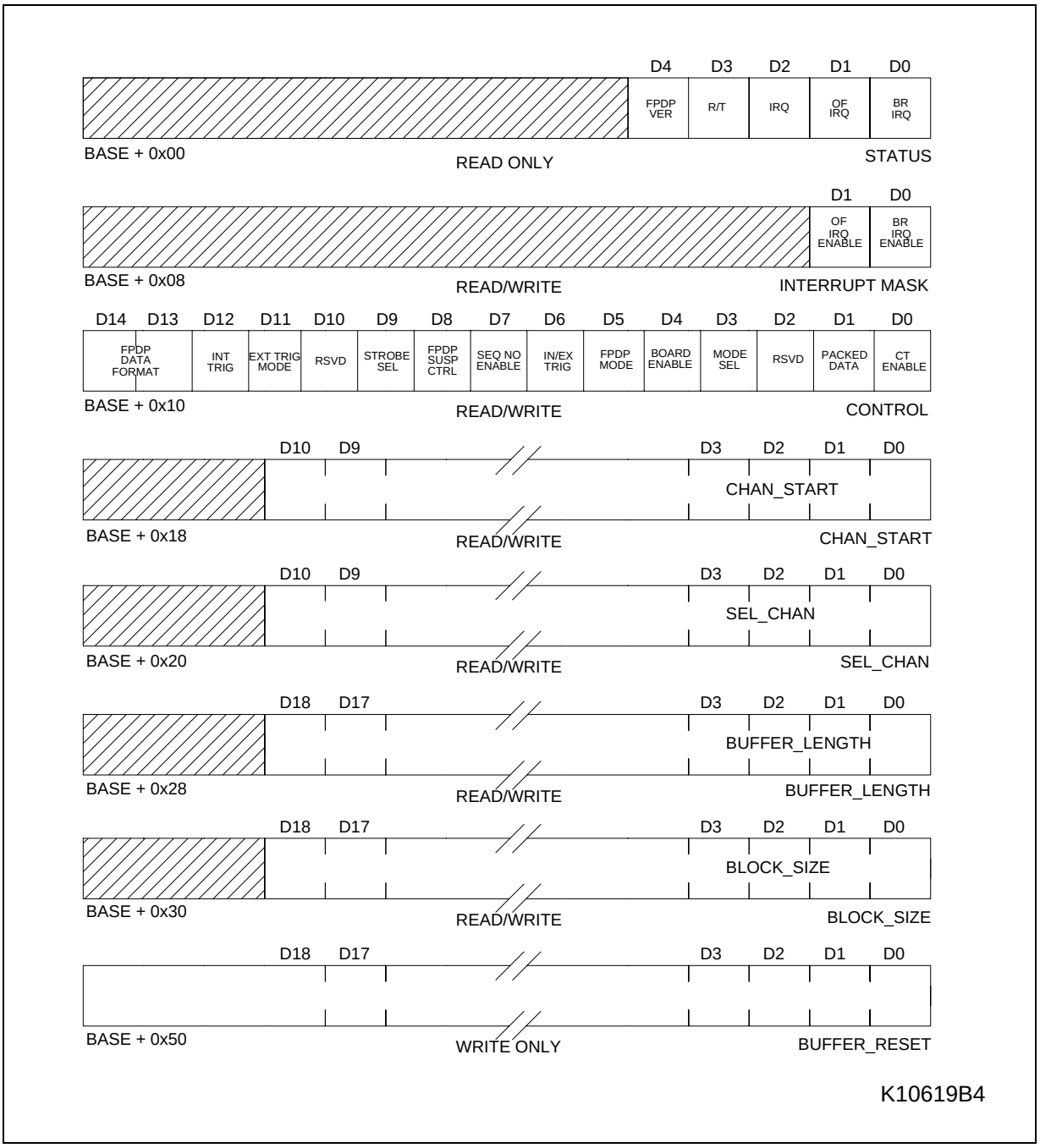

Figure 5 - ICS-500R Register Detail

The following are examples of some of the library routines:

API Function Name	Purpose
ics500RStatusGet	Get the contents of the Status Register
ics500RControlGet	Get the contents of the Control Register
ics500RControlSet	Set the contents of the Control Register
ics500RBufferLengthGet	Get the contents of Buffer Length Register
ics500RBufferLengthSet	Set the contents of Buffer Length Register
ics500REnable	Enable ICS500R Board
ics500RDisable	Disable ICS500R Board
ics500RTrigger	Trigger ICS500R Board
ics500RStartChannelSet	Get the contents of Channel Start Word Register
ics500RStartChannelGet	Set the contents of Channel Start Word Register
ics500RSelectChannelGet	Get the contents of Selected Channel Word Register
ics500RSelectChannelSet	Set the contents of Selected Channel Word Register
ics500RBlockSizeGet	Get the contents of Block Size Register
ics500RBlockSizeSet	Set the contents of Block Size Register
ics500RBufferReset	Issue Buffer Reset
ics500RWaitBufferReadInt	Wait for Buffer Read Interrupt or Timeout

The following is an example application written in ‘C’ for use under Windows. The ICS-500R is configured for continuous mode operation with internal trigger.

```
/*****************************************************************************
/* This is a fragment of code which shows a simple programming sequence
for the ICS-500R. CPU-specific and operating system-specific code have been
omitted.

Copyright 2001, 2002 Interactive Circuits & System Ltd.
 *****************************************************************************/

#include <windows.h>
#include <stdio.h>

#include "ICS500Api.h"


HANDLE               hDevice = INVALID_HANDLE_VALUE;
ULONGLONG           *buffer = NULL;


int
main ()
{
  int                WaitSeconds = 5;
  DWORD              i;
  FILE              *fd;

  ICS500R_CONTROL    BoardControl;
  ULONGLONG          BufferLength, StartChannel, ActiveChannels;
  ULONGLONG          temp;


  /* StartChannel and ActiveChannels could be changed. Refer to Operating Manual. */
  StartChannel    = 0x0;
  ActiveChannels  = 0x20;
  BufferLength    = 0x100;


  hDevice = CreateFile ("\\\\.\\ICS500-1",
                        GENERIC_READ | GENERIC_WRITE,
                        FILE_SHARE_READ | FILE_SHARE_WRITE,
                        NULL, OPEN_EXISTING, 0, (HANDLE) NULL);

  if (hDevice == INVALID_HANDLE_VALUE) {
    printf ("Could not open device!\n");
    goto ErrorExit;
  }
```

```
  if (NULL == (buffer = (ULONGLONG *) malloc ((DWORD)(BufferLength * sizeof (LONGLONG))))) {
    printf ("Not enough memory!\n");
    goto ErrorExit;
  }

  /* Reset everything on the board */
  if (ICS500R_OK != ics500RBoardReset (hDevice)) {
    printf ("ICS500R Board Reset failed!\n");
    goto ErrorExit;
  }

  memset (&BoardControl, 0, sizeof (BoardControl));

  BoardControl.corner_turning   = ICS500_DISABLE;      /* ICS500_ENABLE or ICS500_DISABLE */
  BoardControl.data_format      = ICS500_FPDP_UNPACKED;/* ICS500_FPDP_PACKED or ICS500_FPDP_UNPACKED
*/
  BoardControl.acquisition_mode = ICS500R_CONTINUOUS;  /* ICS500_CONTINUOUS or ICS500_ONESHOT */
  BoardControl.board_enable     = ICS500_DISABLE;      /* ICS500_DISABLE or ICS500_ENABLE */
  BoardControl.receiver_mode    = ICS500_FPDP_MASTER;  /* ICS500_FPDP_RECEIVER or ICS500_FPDP_MASTER
*/
  BoardControl.trigger_source   = ICS500_INTERNAL;     /* ICS500_INTERNAL or ICS500_EXTERNAL */
  BoardControl.sequence_num_en  = ICS500_DISABLE;      /* ICS500_DISABLE or ICS500_ENABLE */
  BoardControl.fpdp_suspend     = ICS500R_SUSPEND_ON;  /* ICS500R_SUSPEND_ON or CS500R_SUSPEND_OFF */
  BoardControl.strobe_selecting = ICS500_TTLCLK;       /* ICS500_TTLCLK or ICS500_PECLCLK */
  BoardControl.trigger_edge     = ICS500_EDGE;         /* ICS500_EDGE or ICS500_LEVEL */
  BoardControl.internal_trigger = ICS500_DISABLE;      /* ICS500_DISABLE or ICS500_ENABLE */
  BoardControl.frame_format     = ICS500R_FRAME_FIXED; /* ICS500R_FRAME_FIXED, ... */

  if (ICS500R_OK != ics500RControlSet (hDevice, &BoardControl)) {
    printf ("ControlSet failed!\n");
    goto ErrorExit;
  }

  if (ICS500R_OK != ics500RStartChannelSet (hDevice, &StartChannel)) {
    printf ("Write Channel Start Word Register Failed!\n");
    goto ErrorExit;
  }
  else
    printf ("Set FPDP Start Channel to 0x%16.16I64x.\n", StartChannel);

  if (ICS500R_OK != ics500RSelectChannelSet (hDevice, &ActiveChannels)) {
    printf ("Write Number of Seleted Channel Word Register Failed!\n");
    goto ErrorExit;
  }
  else
    printf ("Set FPDP Active Channel to 0x%16.16I64x.\n", ActiveChannels);

  temp = BufferLength - 1;
  if (ICS500R_OK != ics500RBufferLengthSet (hDevice, &temp)) {
    printf ("Buffer Length Set failed!\n");
    goto ErrorExit;
  }
  else
    printf ("Set Buffers to 0x%16.16I64x.\n", BufferLength);

  /* The Swing Buffer must be reset in order to receive FPDP data */
  if (ICS500R_OK != ics500RBufferReset (hDevice)) {
    printf ("ICS500R Swing Buffer Reset Failed!\n");
    goto ErrorExit;
  }

  if (ICS500R_OK != ics500REnable (hDevice)) {
    printf ("ICS500R Enable failed!\n");
    goto ErrorExit;
  }

  if (ICS500R_OK != ics500RTrigger (hDevice)) {
    printf ("ICS500R Trigger failed!\n");
    goto ErrorExit;
  }


  if (ICS500R_OK != ics500RWaitBufferReadInt (hDevice, &WaitSeconds)) {
    printf ("Time Out! Test ICS500R Continuous Mode FAILED.\n");
    goto ErrorExit;
  }

  if(!ReadFile (hDevice, (LPVOID) buffer, (DWORD)(BufferLength * sizeof(ULONGLONG)), &i, NULL)) {
    LPVOID lpErrBuffer = NULL;
    DWORD  ErrCode;
```

```
    ErrCode = GetLastError();
    FormatMessage( FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM, NULL, ErrCode,
                   MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                   (LPTSTR) &lpErrBuffer, 0, NULL );

    printf ("Read data failed!\n");
    printf("    error code = %u\n", ErrCode );
    printf("    message    = %s\n", (LPTSTR) lpErrBuffer );

    LocalFree( lpErrBuffer );
  }


  if (ICS500R_OK != ics500RDisable (hDevice)) {
    printf ("ICS500R Disable failed!\n");
  }

  if (ICS500R_OK != ics500RBufferReset (hDevice)) {
    printf ("ICS500R Swing Buffer Reset Failed!\n");
  }

  if(NULL != (fd = fopen ("test.dat", "w"))) {
    for (i = 0; i < (DWORD) BufferLength; i++)
      fprintf (fd, "0x%16.16I64x%c", buffer[i], (i + 1) % 4 ? ' ' : '\n');

    fprintf (fd, "\n");
    fclose (fd);
    fd = NULL;
  }

ErrorExit:

  if (INVALID_HANDLE_VALUE != hDevice) {
    CloseHandle (hDevice);
    hDevice = INVALID_HANDLE_VALUE;
  }

  if (NULL != buffer) {
    free (buffer);
    buffer = NULL;
  }

  printf ("\n\nRun Sample Code Finished!\n\n");

  return 0;
}
```

ICS SENSOR PROCESSING

Interactive Circuits and Systems Ltd.
5430 Canotek Road
Ottawa, Ontario K1J 9G2
Canada

TEL: (613) 749-9241
USA: 1-800-267-9794
FAX: (613) 749-9461

http://www.ics-ltd.com
email: sales@icd-ltd.com

INPUT is published by ICS Ltd. and is available free of charge. Your input is welcome. Please call or write to ICS.