XP-15 Vector Processor

User Guide

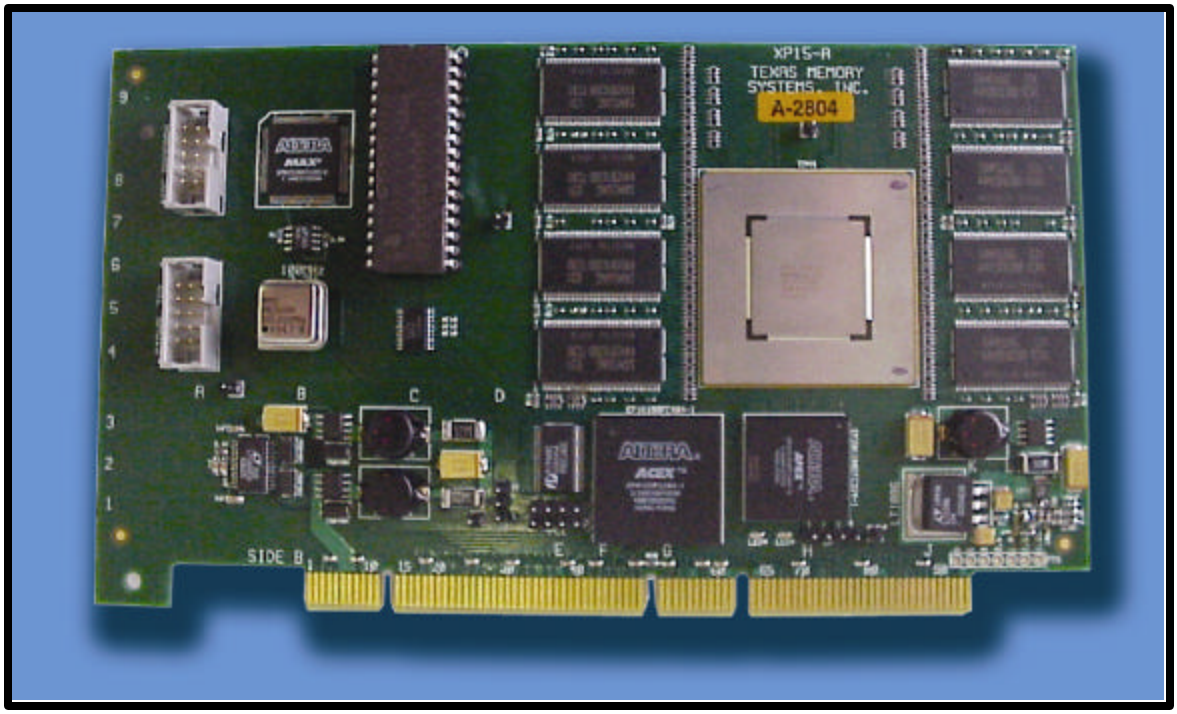

Table of Contents

Chapter 1 - Introduction

The XP-15 vector processor is a PCI card that can be installed in a workstation and that delivers SuperComputer-class vector processing performance. This document is concerned with the installation and set-up of the XP-15.

Chapter 2 gives an overview of the functionality of the XP-15. Chapter 3 describes the hardware and software installation of the XP-15. Chapter 4 describes a programming example for the XP-15 for use in an X-Midas environment.

1.1 Related Texts

The following Texas Memory Systems user documentation may be useful for reference purposes:

TM-44 Programming Overview
VP Scientific Math Library Reference Manual

1.2 Typographical Conventions

Text within this document is typed in the same typeface as this paragraph. User input and information is in the same typeface as the word filename in the example below:

```
Filename
```

Chapter 2 - Overview

The XP-15 is a high-performance vector processor implemented on a single card suitable for installing in a PCI Local Bus card slot in a general-purpose computer. The XP-15 is based on the 8 GFLOPS TM-44 DSP chip from Texas Memory Systems and includes four 32 MB banks of fast local memory. The XP-15 is programmed by making calls to move data from host memory to XP-15 local memory, to perform one or more vector processing operations on the data, and to move results back to host memory. There are more than 400 such function calls available to the XP-15 programmer; detailed specifications for these functions can be found in the *VP Scientific Math Library Reference Manual*, and an introduction to XP-15 programming techniques can be found in the *TM-44 Programming Overview*.

Although many computer systems provide PCI Local Bus card slots, a driver for the XP15 is currently available for computer systems from Compaq under Tru64 version 5.1 only. Accordingly, this document covers the installation and checkout of an XP-15 in a Compaq Alpha-based computer system under Tru64.

Chapter 3 - Installation

The XP-15 vector processor has currently been tested in the following computer platforms:

- ❑ Compaq ALPHASERVER ES40.
- ❑ Compaq ALPHASERVER DS10.

XP-15 Software is currently available for the following operating system revisions:

- ❑ TRU64 (Digital Unix) 5.1

3.1 Installing the XP-15 Hardware

3.1.1 Jumper Settings

The XP-15 has no user-selectable jumpers.

3.1.2 Installing the XP-15 in a Computer

Power-off the computer system. Select a suitable vacant PCI slot and remove the faceplate. Insert the XP15 card firmly into the vacant slot. Lock down the XP15 faceplate using the screw that previously held in the blank faceplate.

3.2 Installing the XP-15 Software

XP-15 software is delivered on CDROM. This chapter describes the steps that should be used to install the software. The account that performs the installation must have certain system privileges, and is normally the system account.

Release notes for the delivered version of the XP-15 hardware and software are included on the CDROM in a file called README.TXT. This file lists enhancements since the previous version, known hardware and software deficiencies, documentation errors, and other information not included in the manuals.

3.2.1 Tru64 (Digital Unix) Installation

Log on to the computer with system privileges, this is normally root. Insert and mount the CDROM using a command along the lines of the one below:

```
# mount –r /dev/disk/cdrom0c /cdrom
```

There are two script files located on the CDROM, one is to load the user software and the other is to load the XP15 driver. First load the user software as follows:

```
# cd /cdrom                          change directory to the CDROM mount point
# ./install_software                 invoke the software install script
```

The script will run and will ask questions about where the software is located and where to install it on the host machine. A typical run is shown below with normal user responses underlined.

```
Installing Texas Memory Systems XP15 USER software

Are you installing the software on a Compaq Tru64 system (y/n)?yes
```

```
Enter the mount point of the CDROM - /cdrom

What directory do you want to install the XP-15 software to - /usr/tms

The XP15 software will be installed from /cdrom to /usr/tms
Is this correct (y/n)?yes

Installing the XP15 software.

Finished installing the USER part of the XP15 software.
```

The final step in installing the XP15 is to install the driver using a script that also asks a few questions. A typical run is shown, again with user responses underlined.

```
#cd /cdrom
#./install_driver

Installing Texas Memory Systems XP15 DRIVER software

Are you installing the DRIVER software on a Compaq Tru64 system (y/n)?yes

Enter the mount point of the CDROM - /cdrom

The XP15 driver software will be install from /cdrom
Copy driver source
Add path to driver source files
Create make and make the driver
Create soft links
Update system configuration and load the XP15 driver

You need to reboot the system to make the new driver available

Do you want to reboot the system now (y/n)?yes
```

3.3 Running the XP-15 Confidence Test

The XP-15 confidence test is a two-stage process, starting with a simple diagnostic test followed by running a simple programming example.

3.3.1 Diagnostic Confidence Test

The XP-15 diagnostic is located in the **$TMS/bin** directory that was installed from the cdrom and is called XPMON. XPMON has many features and commands, but the one command that tests the basic operation of the XP-15 is the **T99** command. A sample run is shown, the XP-15 to be tested is **/dev/xp0**.

```
$xpmon –d /dev/xp0

     XP-15 Diagnostic Monitor v1.0

XP-15 device “/dev/xp0” opened successfully

Allocating host data buffers:
   DMA write buffer size = 0x2000000 bytes
   DMA read buffer size = 0x2000000 bytes

Verbosity set to HIGH
```

```
Xpmon[0]> t99
Running all tests:

Performing SPRAG tests …
            .
            .
    testing many functions
            .
            .
```

All tests should run without any errors. If any errors occur please contact Texas Memory Systems.

3.3.2 Programming Example Confidence Test

The XP-15 confidence test is located on the XP-15 software CDROM in the **`examples/cfft`** directory, and a listing of the source code can be found in Appendix A. This simple application creates a signal containing a single impulse, transfers the signal from host memory to XP-15 local memory, performs a one million point complex FFT on the signal in the XP-15, reads the results back to host memory, and checks the results.

Copy the contents of the **`examples/cfft/`** directory from the CDROM to a suitable local directory. For instructions on reading the CDROM, please refer to the *Installing the XP-15 Software* section for the relevant computer and operating system earlier in this document.

```
% cp $CDROM/sources/examples/cfft/* .
```

This will copy two files from the CDROM: **`cfft.c`** and **`Makefile`**. The make utility can be used to build the program. The make file assumes that the XP-15 software has been installed in the $TMS directory as described in the software installation section above. If this is not so, then the make file will have to be modified to reflect the new software location.

```
% make
gcc -I$TMS/include -o cfft cfft.c -L$TMS/lib –lvp -lsam
```

The sample application is executed with one parameter on the command line: the XP-15 unit to be used. Typically, this will be unit 0 (zero) unless multiple XP-15s are installed in this host computer.

```
% cfft -x 0
XP-15 Confidence test performing a 1048576 point CFFT on XP-15 #0
Allocate memory for test signal and FFT results
Clearing results buffer
Creating test signal
Performing CFFT
Wait for all XP-15 functions to complete
First 10 values of CFFT results:
00: [ 16.0000,   0.0000]
01: [ 16.0000,   0.0000]
02: [ 16.0000,   0.0000]
03: [ 16.0000,   0.0000]
04: [ 16.0000,   0.0000]
05: [ 16.0000,   0.0000]
06: [ 16.0000,   0.0000]
07: [ 16.0000,   0.0000]
08: [ 16.0000,   0.0000]
09: [ 16.0000,   0.0000]
Checking result vector..
XP-15 confidence test completed with no errors
```

Chapter 4 - X-Midas Programming Example: XTMIC

This chapter describes the X-Midas programming example, how to execute it, and how to use it as a template for XP-15 applications. The source code is located on the XP-15 software CDROM in the **examples/xtmic/** directory.

4.1 Overview

The X-Midas programming example is a less robust version of the SAMTMIC software that currently runs on the SAM-450. The XP-15 implementation, referred to as the XTMIC, performs the same basic processing as the SAMTMIC, but with predefined parameters and static weight files. The XTMIC performs spectral analysis and interference canceling on two channels of 16-bit data using an FFT size of 128K. The algorithm starts with a forward four-tap polyphase filter, and a forward FFT, applies a set of predefined weights, performs the inverse FFT, and then the inverse polyphase filter. The auto spectrum of each signal is calculated, as well as the cross spectrum of the two signals, and averaged versions of these spectra are sent to the host computer for display. In addition to the auto and cross spectra signals, the end result of the TMIC calculations is also piped back to the host.

4.2 Running the XTMIC Example

Copy the contents of the **examples/xtmic/** directory from the CDROM to a suitable local directory. For instructions on reading the CDROM, please refer to the *Installing the XP-15 Software* section for the relevant computer and operating system earlier in this document.

```
% cp –r $CDROM/examples/xtmic/* .
```

This will copy the entire option tree to the specified location. Start the X-Midas programming environment, and then associate the XTMIC option tree with this location using the X-Midas **xmopt** function. For example, if the xtmic option tree was copied to **/home/midas/projects/xtmic/**, then one would type:

```
% xmopt XTMIC /home/midas/projects/xtmic/
```

Then, to add the XTMIC project to the current path:

```
% xmpath +XTMIC
```

The **xmbopt** utility can then be used to build the program.

```
% xmbopt XTMIC
```

The build file assumes that the XP-15 software has been installed in the $TMS directory as described in the software installation section above. If this is not so, then the build file will have to be modified to reflect the new software location.

The XTMIC macro requires two command line parameters that determine which files the XTMIC will use for its processing. The input files must contain 16-bit integer data, and must have at least 1507328 elements. Assuming the input files are called input_a and input_b, the command line required to run the XTMIC would look like:

```
% X-Midas> run_x input_a input_b
 XTMIC> Starting xtmic demo.
 XTMIC> Setting up files/pipes.
 XTMIC> Initializing XP15 buffers.
 XTMIC> Entering loop.
 Total iterations:    10.0000000000000
 Elapsed seconds:     5.945312
 Average outer loop speed was    2.01839685440063
 XTMIC> Finished xtmic demo.
```

Appendix A – XP-15 Confidence Test

```
/*****************************************************************************

           Copyright (c) 2001 by Texas Memory Systems, Inc.

                  Texas Memory Systems, Inc.
                    11200 Westheimer #1000
                      Houston, TX  77042
                        (713) 266-3200

      Module Name:     cfft.c
      Package:         CFFT demo
      Version:         $Revision:    $
      Release Date:    $Date:        $
      Description:     XP15 FFT demo
 *****************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <samdef.h>
#include <vp.h>

#define FFT_SIZE  (1024*1024)

static void help(void);

/*****************************************************************
Function:   main

Description:      entry point for XP15 FFT demo

Arguments:  argc  -     number of inputs
            argv  -     pointer to array of inputs

Returns:    none
*****************************************************************/


int main (int argc, char *argv[])
{
    int          cc;
    extern char *optarg;
    extern int   optind;

    int          ii;
    int          ecount;
    float        real_val, imag_val;

    TMS_INT      xp15     = -1;
    TMS_INT      fft_size = FFT_SIZE;
    TMS_INT      status;
    TMS_COMPLEX *signal, *results;
    TMS_COMPLEX *cu, *cy;

/*
 *  Get command line arguments
```

```
 */
   while ((cc = getopt(argc, argv, "hx:")) != EOF)
   {
      switch (cc)
      {
      case 'x':
         xp15 = atoi(optarg);
         break;
      case 'h':
      case '?':
      default :
         help();
      }
   }

/*
 *  Print some useful information
 */
   printf("XP-15 Confidence test performing a %d point CFFT on XP-15 #%d\n",
          fft_size,
          xp15);

/*
 *  Allocate host memory for a test signal
 */
   printf("Allocate memory for test signal and FFT results\n");
   signal = (TMS_COMPLEX *)malloc(fft_size * sizeof(TMS_COMPLEX));
   if (signal == NULL)
   {
     printf("Failed to allocate signal buffer memory\n");
     exit(-1);
   }
   results = (TMS_COMPLEX *)malloc(fft_size * sizeof(TMS_COMPLEX));
   if (results == NULL)
   {
     printf("Failed to allocate results buffer memory\n");
     exit(-1);
   }

/*
 *  Create a test signal, a single spike, and clear results buffer
 */
   printf("Clearing results buffer\n");
   memset(results, 0, fft_size * sizeof(TMS_COMPLEX));
   printf("Creating test signal\n");
   memset(signal, 0, fft_size * sizeof(TMS_COMPLEX));
   signal[0].real = 16.0;
   real_val = signal[0].real;
   imag_val = 0.0;

/*
 *  Allocate XP-15 buffers
 */
   cu = (TMS_COMPLEX *)vb_pointer(xp15, VP_RAM_A_MEM, 0);
   cy = (TMS_COMPLEX *)vb_pointer(xp15, VP_RAM_B_MEM, 0);

/*
 *  Move the input signal into XP-15 local memory
 */
```

```
    status = vmov((TMS_FLOAT *)signal, (TMS_FLOAT *)cu, fft_size * 2);
    if (status < 0)
      sam_error(status);

/*
 *  Perform the CFFT
 */
    printf("Performing CFFT\n");
    status = cfft(cu,                   /* input buffer               */
                  NULL,                 /* Twids not needed by XP-15  */
                  cy,                   /* output/scratch buffer      */
                  1,                    /* repeat value - 1 this time */
                  1.0,                  /* forward FFT scaled by 1.0  */
                  fft_size);            /* one million complex points */
    if (status < 0)
      sam_error(status);

/*
 *  Move the FFT results back to host memory
 *      (returned status indicates which buffer holds the final results)
 */
    if (status == 1)
      status = vmov((TMS_FLOAT *)cu, (TMS_FLOAT *)results, fft_size * 2);
    else
      status = vmov((TMS_FLOAT *)cy, (TMS_FLOAT *)results, fft_size * 2);
    if (status < 0)
      sam_error(status);

/*
 *  Synchronize with the XP-15
 */
    printf("Wait for all XP-15 functions to complete\n");
    status = viper_sync(xp15, -1);
    if (status < 0)
      sam_error(status);

/*
 *  Print values
 */
    printf ("First 10 values of CFFT results:\n");
    for (ii=0; ii<10; ii++)
        printf ("%02d: [%8.4f, %8.4f]\n",
                ii, results[ii].real, results[ii].imag);

/*
 *  Check result vector
 */
    ecount = 0;
    printf ("Checking result vector..\n");
    for (ii=0; ii<fft_size; ii++)
    {
      if ((results[ii].real != real_val) || (results[ii].imag != imag_val))
      {
          if (ecount < 5)
          {
            printf ("Error in element %d", ii);
            printf ("    ShB: [%8.4f, %8.4f] ", real_val, imag_val);
            printf ("Is: [%8.4f, %8.4f]\n",
                        results[ii].real,
```

```
                        results[ii].imag);
            }
            ecount++;
        }
    }
    if (ecount == 0)
        printf ("XP-15 confidence test completed with no errors\n");
    else
        printf ("XP-15 confidence test completed with %d errors\n", ecount);

    exit(0);
}

/*********************************************************************
Function:   help

Description:      display help screen, and exits

Arguments:  none

Returns:    none
*********************************************************************/
static void help(void)
{
    fprintf (stdout, "parameters:\n");
    fprintf (stdout,
           "         -x         : XP-15 device name\n");
    fprintf (stdout,
           "         -h         : this message\n");

    exit (0);
}
```

Appendix B – PCI Local Bus Specifications

The XP-15 interface adapter is a 3.3V signaling environment expansion card, as defined by the *PCI Local Bus Specification 2.0, 2.1, and 2.2*. This means that the XP-15 is designed to operate exclusively in 3.3V signaling environment system motherboards.

During the development of the XP-15, the liberty was taken to add a second key slot on the PCI edge connector. This allows for operation and testing in most 5V signaling environment motherboards that provide a 3.3V power rail. This is neither a recommended nor a supported configuration.

The following table clarifies the requirements for system motherboards under the different revisions of the PCI Specification.

System Motherboards	5V Signaling Environment	3.3V Signaling Environment
PCI Local Bus Specification 2.0	+5V, +12V, and -12V required. +3.3V may either be included with system, or a means to add it afterwards must be provided.	+5V, +3.3V, +12V, and -12V rails are all required.
PCI Local Bus Specification 2.1	+5V, +12V, and -12V required. +3.3V may either be included with system, or a means to add it afterwards must be provided.	+5V, +3.3V, +12V, and -12V rails are all required.
PCI Local Bus Specification 2.2	+5V, +3.3V, +12V, and -12V rails are all required.	+5V, +3.3V, +12V, and -12V rails are all required.

The following table clarifies the requirements for expansion cards in the different signaling environments.

Expansion Cards	5V Signaling Environment	3.3V Signaling Environment
5V Expansion Card	I/O Buffers are powered by the 5V power rail. *2.0,2.1*: +5V, +12V, -12V supplied. If +3.3V is needed, expansion card must provide its own 3.3V. *2.2*: +5V, +3.3V, +12V, and -12V rails are all supplied.	5V Expansion Cards are not designed to work in a 3.3V signaling environment.
3.3V Expansion Card (XP-15)	3.3V Expansion Cards are not designed to work in a 5V signaling environment.	I/O Buffers are powered by the 3.3V power rail. +5V, +3.3V, +12V, and -12V rails are all supplied.
Universal Expansion Card	I/O Buffers are powered by the 5V power rail. *2.0,2.1*: +5V, +12V, -12V supplied. If +3.3V is needed, expansion card must provide its own 3.3V. *2.2*: +5V, +3.3V, +12V, and -12V rails are all supplied.	I/O Buffers are powered by the 3.3V power rail. *2.0,2.1*: +5V, +12V, -12V supplied. If +3.3V is needed, expansion card must provide its own 3.3V. *2.2*: +5V, +3.3V, +12V, and -12V rails are all supplied.

Texas Memory Systems recommends the XP-15 adapter for use in 3.3V signaling environments.

Appendix C – XP-15 Hardware Control and Status Registers

The XP-15 is a custom hardware adapter designed to provide processing power to a host computer (DEC Alpha, Intel PC, etc..). The XP-15's data bus is 32-bits or 64-bits wide (depending on the host PCI slot).

The following tables describe the XP-15's hardware control registers. Each register is classified as read only, write only, or read/write. The least significant bit of each register is always tied to the least significant bit of the PCI bus. Any unspecified bits should be masked in software after they are read, even though they will not be used.

As defined by the PCI Local Bus Specification Rev. 2.2, the XP-15 has one Base Address Regions. All registers are defined as offsets into this one address region. Bit-level definitions of all registers, including rules for their use, appear in the following tables and paragraphs.

XP-15 Hardware Registers				
Register Name	Base Address Region	Offset	Type	Size
Control-Status Register (CSR)	0	0h	R/W	32 bits
Low PCI SN command address register[31..0]	0	4h	R/W	32 bits
High PCI SN command address register[63..32]	0	8h	R/W	32 bits
Low PCI SN response address register[31..0]	0	Ch	R/W	32 bits
High PCI SN response address register[63..32]	0	10h	R/W	32 bits

Control-Status Register (32 bits, R/W)			
Bit #	Name	Type	Description
31-16	RESERVED	N/A	These bits are reserved.
15	Reset	WO	Softreset, puts the XP-15 in the power-up state.
14	Start	WO	Informs XP-15 to start processing SN command.
13	END	WO	Indicates host is big Endian.
12-10	RESERVED	N/A	These bits are reserved.
9	64	WO	Indicates XP-15 is in a 64-bit PCI slot
8	RESERVED	N/A	This bit is reserved.
7	DMADONE	RO	Indicates the DMA is finished.
6	STOP	RO	Indicates the STOP bit has been encountered.
5	PAUSE	RO	Indicates the XP-15 is paused.
4	ERR	RO	Indicates the XP-15 has received an SN checksum error.

3	ECHO	RO	Indicates the XP-15 has returned an ECHO packet
2	SPR	RO	Indicates the XP-15 has returned an SPR packet
1	DIAG	RO	Indicates the XP-15 has returned a diagnostic packet
0	STAT	RO	Indicates the XP-15 has returned a status packet

Low PCI SN command address register (32 bits, R/W)			
Bit #	Name	Type	Description
31-3	NA	R/W	This register is used to establish the low 32-bits of the PCI address for SN commands to the XP-15. This register consists of the bottom 32-bits of a 32-bit counter with the low-order three bits hardwired to zeroes. Transfers must begin on a Double DWORD (64-bit) boundary.
2-0	Reserved	RO	These bits are reserved (must be written with zeroes).

High PCI SN command address register (32 bits, R/W)			
Bit #	Name	Type	Description
63-32	NA	R/W	Upper bits of this register are not used.

Low PCI SN response address register (32 bits, R/W)			
Bit #	Name	Type	Description
31-3	NA	R/W	This register is used to establish the low 32-bits of the PCI address for SN responses from the XP-15. This register consists of the bottom 32-bits of a 32-bit counter with the low-order three bits hardwired to zeroes. Transfers must begin on a Double DWORD (64-bit) boundary.
2-0	Reserved	RO	These bits are reserved (must be written with zeroes).

High PCI SN response address register (32 bits, R/W)			
Bit #	Name	Type	Description
63-32	NA	R/W	Upper bits of this register are not used.