Linux Driver Manual

PMC-Parallel-485-NG1

Manual Revision	01p1
Revision Date	04/18/2023
Corresponding Hardware	PMC-Parallel-485-NG1
Current Fab Number	10-1999-0305

Dynamic Engineering
150 DuBois St. Suite B/C
Santa Cruz, CA 95060
(831) 457-8891
https://www.dyneng.com
sales@dyneng.com
Est. 1988

Cautions and Warnings

The electronic equipment described herein generates, uses, and can radiate radio frequency energy. Operation of this equipment in a residential area is likely to cause radio interference, in which case the user, at their own expense, will be required to take whatever measures may be required to correct the interference.

Dynamic Engineering's products are not authorized for use as critical components in life support devices or systems without express written approval from the president of Dynamic Engineering.

Connection of incompatible hardware is likely to cause serious damage.

Table of Contents

Design Revision History 1
Manual Revision History 1
Product Description 2
Software Description 2
Driver Installation 2
Driver Startup 3
Software Quick Start 3
IOCTLs 3
  IOCTL Definitions 4
    DE_GET_BD_INFO 4
    DE_REG_WAIT 4
    DE_WAIT_INT 4
    DE_REG 4
Warranty and Repair 5
  Service Policy 5
  Out-of-Warranty Repairs 5
  Contact 5
Glossary 6

Figures

No table of figures entries found.

Tables

Table 1: Design Revision History 1
Table 2: Manual Revision History 1
Table 3: Header Files 2

Design Revision History

Table 1: Design Revision History

Revision	Date	Description

Manual Revision History

Table 2: Manual Revision History

Revision	Date	Description

NOTE: Dynamic Engineering has made every effort to ensure that this manual is accurate and complete; that being said, the company reserves the right to make improvements or changes to the product described in this document at any time and without notice. Furthermore, Dynamic Engineering assumes no liability arising out of the application or use of the device described herein.

Product Description

"PMC-Parallel-485-NG1" features a Spartan II Xilinx FPGA to implement the PCI interface, and IO processing, control and status for 32 differential IO. 50 MHz reference can be used for the clock divider. The NG1 version has some IO converted to special definitions [See HW manual] and two cascaded counters for generating system time.

Software Description

The PMC-Parallel-485-NG1 driver was developed on Ubuntu 18.04 with Kernel Version 5.4.0-144. The kernel modules are built via compile time definitions contained within the Makefile.

This package comes with the PmcPar485Ng1LnxUserApp, which is a stand-alone code set with a simple and powerful menu plus a series of tests that can be run on the installed hardware. Each of the tests execute calls to the driver, pass parameters and structures, and get results back. With the sequence of calls demonstrated, the functions of the hardware are utilized for loop-back testing. The software is used for manufacturing tests at Dynamic Engineering. The test software can be ported to your application to provide a running start. The tests are simple and will quickly demonstrate the end-to-end operation of your application making calls to the driver and interacting with the hardware.

The menu allows the user to add tests, to run sequences of tests, to run until a failure occurs and stop or to continue, to program a set number of loops to execute and more. The user can add tests to the provided test suite to try out application ideas before committing to your system configuration. In many cases the test configuration will allow faster debugging in a more controlled environment before integrating with the rest of the system.

Table 3: Key Header Files

File Name	Description
Ioctl.h	Defines the API for the device
PmcPar485Ng1.h	Defines the Ioctl calls and data structures used in these calls.

Driver Installation

As our customers often use various implementations of the Linux kernel (including custom kernels), we do not provide a pre-built kernel driver binary for loading. Instead, we provide a make file that will, in most systems, build a kernel driver binary (.ko) file. To build the binary for the Monitor:

1. Navigate to PmcPar485Ng1Lnx/build directory
2. Command: **sudo make**

Once the driver has been built, the next step is to use the script provided to install the driver.

3. Command: **sudo ./Install**

The script should output a device number (e.g., 238). This script also creates the following device node to connect to software applications:

- **/dev/de_PmcPar485Ng1_0**

NOTE: The installation script does not permanently install the device driver. Again, as our clients all have very different implementations on Linux, we have found it more beneficial to keep the installation process simple and allow clients to customize how they would like to administer their third-party drivers.

If you wish to install the driver permanently for Ubuntu, you must look at the kernel you are using (uname -r), then, place the .ko file in:

- **/lib/modules/<your-kerne>/kernel/drivers/** and add edit the config file at:
- **/etc/modules-laod.d/modules.conf** to load your specific driver.

Driver Startup

Once the driver has been installed, connecting to the device is fairly simple from software. Indeed, one only needs to call the standard system command open() while passing the device node name:

- (“**dev/ de_PmcPar485Ng1_0”** and the **O_RDWR flag**)

This will return the file descriptor for that device (for an example see: **PmcPar485Ng1LnxUserApp/main.cpp**).

Software Quick Start

Once the driver is installed and the software has been connected to the device using the standard open() system call, using the device is fairly straight forward using the API defined in ioctl.h, which provides access to the different registers in the device.

IOCTLs

In Addition to the above, the drivers use a few other IO Control calls (IOCTLs) to access the device. These are defined in, but there are also wrapper functions for each of these in the API:

- **PmcPar485Ng1.h** and
- **de_Common.h**

They can be used by calling the system call: **ioclt()**

```
int ioctl(
  int           fd,        // file descriptor returned from open()
  unsigned long  request,    // Control code defined in API header file
  …,                       // pointer to structure, when required
);
```

IOCTL Definitions

DE_GET_BD_INFO

- Functions: Gets the relevant board information for each port. This IOCTL takes a struct pointer as a parameter and then fills it.
- Input: **de_rev_t ***
- Output: **de_rev_t ***
- Notes:

```
typedef struct de_rev {
    //driver version major number
    unsigned long   driver_major;
    //driver version minor
    unsigned long   driver_minor;
    // FPGA major revision
    unsigned char   major;
    // FPGA minor revision
    unsigned char   minor;
    // DIP switch settings
    unsigned char   dips;
} de_rev_t;
```

DE_REG_WAIT

- Function: Registers that the driver will perform a wait for interrupt event
- Input: **None**
- Output: **None**

DE_WAIT_INT

- Function: This call will block, waiting for interrupt event
- Input: **long ***
- Output: **None**
- Notes:

  Input is the timeout period

DE_REG

- Function: Gets/Sets specific register values at given level (port or base) and offset
- Input: **de_reg_cmd_t ***
- Output: **de_reg_cmd_t ***
- Notes:

```
typedef struct de_reg_cmd {
    /* DE_GET_OP, DE_SET_OP, DE_RMW_OP      */
    de_op_t         op;
    /* Value read, or to be written         */
    unsigned int    val;
    /* Reg number or word offset from base  */
    unsigned int    reg;
    /* Mask if op is RMW                    */
    unsigned int    mask;
} de_reg_cmd_t;
```

Warranty and Repair

Please refer to the warranty page on our website for the warranty and options that are currently offered.

www.dyneng.com/warranty

Service Policy

Before returning a product for repair, verify to the best of your ability, that the suspected unit is as fault. Then call the Dynamic Engineering Customer Service Department for a Return Material Authorization (RMA) number. Carefully package the product, in the original packaging if possible, and ship prepaid and insured with the RMA number clearly written on the outside of the package. Include a return address and the telephone number of a technical contact. For out-of-warranty repairs, a purchase order for repair charges must accompany the return. Dynamic Engineering will not be responsible for damages due to improper packaging of returned items. For service on Dynamic Engineering products not purchased directly from Dynamic Engineering, contact your reseller. Products returned to Dynamic Engineering for repair by anyone other than the original customer will be treated as out-of-warranty.

Out-of-Warranty Repairs

Out-of-warranty repairs will be billed on a material and labor basis. Customer approval will be obtained before repairing any item if the repair charges will exceed one half of the list price for one of that kind of unit. Return transportation and insurance will be billed as part of the repair in addition to the minimum RMA charge.

Contact:

Customer Service Department
Dynamic Engineering
150 DuBois St. Suite B/C
Santa Cruz, CA 95005
(831) 457-8891
support@dyneng.com

Glossary

Baud	Used as the bit period when talking about UARTs; Not strictly correct, but is the common usage when talking about UARTs.
CardID	Unique number assigned to a design to distinguish between all designs of a particular vendor
CFM	Cubic feet per minute
FIFO	First In First Out memory
Flash	Non-volatile memory used on Dynamic Engineering boards to store FPGA configurations or BIOS
JTAG	Joint Test Action Group – a standard used to control serial data transfer for test and programming operations.
LFM	Linear feet per minute
LVDS	Low Voltage Differential Signaling
MUX	Multiplexor – multiple signals multiplexed to one with a selection mechanism to control which path is active.
Packed	When UART characters are always sent/received in groups of four, allowing full use of host bus/FIFO bandwidth.
Packet	Group of characters transferred. When the characteristics of the group of characters is known, the data can be stored in packets and transferred as such; the system is optimized as a result. Any number of characters can be transferred.
PCI	Peripheral Component Interconnect – parallel bus from host to this device
PIM	PMC Interface Module (PIM). Provides rear I/O in cPCI systems. Mounts to PIM Carrier
PIM Carrier	PIM Mounting Device. Mounts on rear of cPCI backplane.
PMC	PCI Mezzanine Card – establishes common connectors, connections, size and other mechanical features.
TAP	Test Access Port – basically a multi-state port that can be controlled with JTAG [TMS, TDI, TDO, TCK]. The TAP States are the states in the State Machine that are controlled by the commands received over the JTAG link.
TCK	Test Clock provides synchronization for the TDI, TDO, and TMS signals

TDI	Test Data in – this serial line provides the data input to the device controlled by the TMS commands. For example, the data to program the FLASH comes on the TDI line while the commands to the state machine to move through the necessary states comes over TMS. Rising edge of TCK valid.
TDO	Test Data Out is the shifted data out. Valid on the falling edge of the TCK. Not all states output data.
TMS	Test Mode State – this serial line provides the state switching controls. '1' indicates to move to the next state, '0' means stay put in cases where delays can happen; otherwise, 0,2 are used to choose which branch to take. Due to the complexity of state manipulation, the instructions are usually precompiled. Rising edge of TCK valid.
UART	Universal Asynchronous Receiver Transmitter. Common serialized data transfer with start bit, stop bit, optional parity, optional 7/8 bit data. Can be over any electrical interface. RS232 and RS422 are most common.
Unpacked	When UART characters are sent on an unknown basis requiring single character storage and transfer over the host bus
VendorID	Manufacturers number for PCI/PCIe boards. DCBA is Dynamic Engineering's VendorID
VME	Versa Module European
VPX	Family of standards based on the VITA 46.0
XMC	Switched mezzanine card (PMC with PCIe)