NetBurner Blade Board Reference Guide

NBPKBD-100 Digital GPIO Blade

Revision 1.0
June 12, 2007
Released

Table of Contents

1. Introduction

The NBPKBD-100CR is a “Personality Blade” for the NBPK70EX-100, and has the following hardware features:

- 32 Digital I/O Channels
- Each channel is programmable to High, Low, HiZ, or input
- Each channel has its own 74HCT125 driver for 20mA of current drive
- Jumper selectable output voltage: 3.3V or 5V
- 5V tolerant inputs

This reference guide also covers the software library API used for programming with the NBPKBD-100CR.

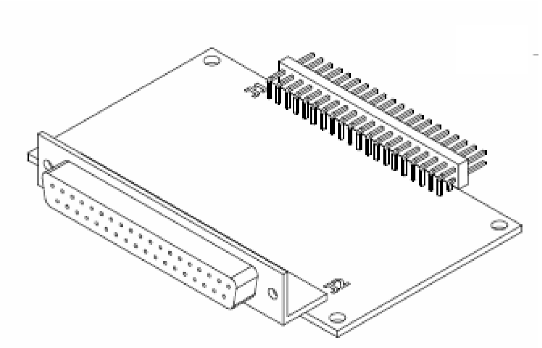
2. Additional Documentation

- PK70 (Hard Copy) Quick Start Guide
- PK70 Hardware Manual is located (by default) in your C:\Nburn\docs directory
- NNDK User's Manuals are located (by default) in your C:\Nburn\docs directory
- NNDK Programmer's Guide (PDF) is located (by default) in your C:\Nburn\docs directory
- NBEclipse Getting Started Guide (PDF) is located (by default) in your C:\Nburn\docs directory
- NetBurner Dev C++ Quick Start Guide (PDF) is located (by default) in your C:\Nburn\devcpp\Help directory
- HCC-Embedded - Embedded Flash File System Implementation Guide Version 2.62
    - All EFFS Documentation are located (by default) in your C:\Nburn\docs\files directory
- All License Information is located (by default) in your C:\Nburn\docs directory

3. Installation

The NBPKBD is mounted inside the PK70 enclosure. It has two connectors: a DB37 that connects to external devices, and a dual row 40-pin right angle header that connects the NBPKBD to the PK70 interface connector.

To install the NBPKBD, remove the PK70 cover, plug the 40-pin dual row right angle header (J1) into the 40-pin socket on the PK70, and install the four 4-40 mounting screws. Finally, replace the PK70 cover and cover screws.



The software libraries are automatically installed with your PK70 development kit.

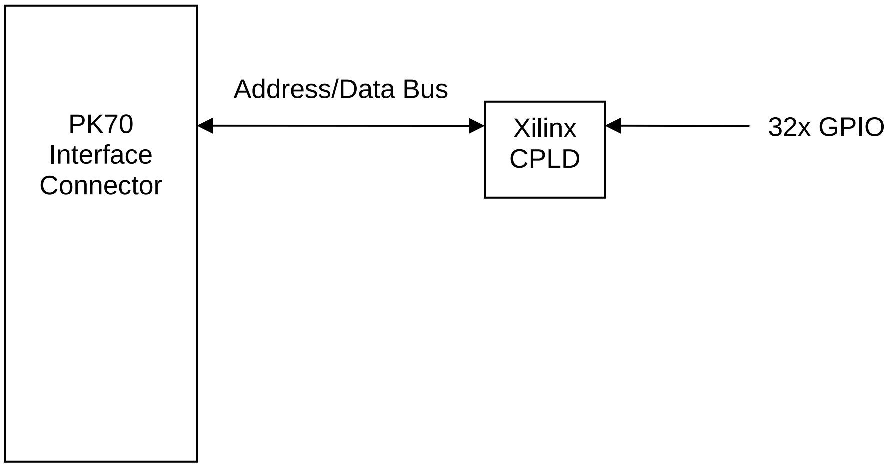
4. Hardware

The NBPKBD board interfaces to the PK70 through the 40 pin interface connector. This interface connector includes the address bus, data bus, I2C, chip selects, and interrupt signals.

The following hardware is used to implement the GPIO, DAC and A/D functions:

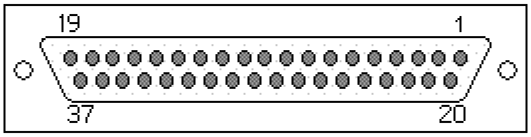
Function	Description
GPIO	Xilinx XC95144XL CPLD
GPIO Buffers	74HCT125 buffers providing selectable output voltage of 3.3V or 5V, and 20mA of current drive.

Block Diagram



5. Connector Pinout

The DB37 pinout is shown below:



DB-37 Female

Pin	Signal	Pin	Signal
1	GND	20	GPIO
2	GPIO	21	GPIO
3	GPIO	22	GPIO
4	GPIO	23	GPIO
5	GPIO	24	GPIO
6	GPIO	25	GPIO
7	GPIO	26	GPIO
8	GPIO	27	GPIO
9	GPIO	28	GND
10	GND	29	GPIO
11	GPIO	30	GPIO
12	GPIO	31	GPIO
13	GPIO	32	GPIO
14	GPIO	33	GPIO
15	GPIO	34	GPIO
16	GPIO	35	GPIO
17	GPIO	36	GPIO
18	GPIO	37	GND
19	GND		

- Please refer to your PK70 Hardware Manual (in C:\Nburn\docs) for the pinout of the 40-pin dual row interface connector.

6. Software Programming

The programming interface for the NBPKBD is a C++ class that enables you to access each function by the pin number of the DB37 connector. The procedure is to initialize each pin by specifying its ***function( )***, then read and write values to the pin. This is the same pins class concept used on other NetBurner platforms.

The source code is located in \Nburn\PK70\include\NBPKBD.h and \Nburn\PK70\system\NBPKBD.cpp. You can use or modify the existing pins class code, or use the pins class implementation as an example to create your own NBPKMB interface functions.

6.1 GPIO Pins

6.1.1 GPIO Voltage Levels

The dedicated GPIO pins can be configured as 3.3V or 5V logic. The selection is made on the NBPKBD board with the 3-pin JP1 header:

- 3.3V Logic: Install jumper on JP1 pins 2-3
- 5.0V Logic: Install jumper on JP1 pins 1-2

6.1.2 GPIO Definitions

The following definitions are located in \Nburn\PK70\include\NBPKBD.h. They are used as parameters in the Pins Class ***function()*** member function.

```
// Dedicated GPIO pins
#define P1_FUNCTION_GND (0)
#define P2_FUNCTION_GPIO (0)
#define P3_FUNCTION_GPIO (0)
#define P4_FUNCTION_GPIO (0)
#define P5_FUNCTION_GPIO (0)
#define P6_FUNCTION_GPIO (0)
#define P7_FUNCTION_GPIO (0)
#define P8_FUNCTION_GPIO (0)
#define P9_FUNCTION_GPIO (0)
#define P10_FUNCTION_GND (0)
#define P11_FUNCTION_GPIO (0)
#define P12_FUNCTION_GPIO (0)
#define P13_FUNCTION_GPIO (0)
#define P14_FUNCTION_GPIO (0)
#define P15_FUNCTION_GPIO (0)
#define P16_FUNCTION_GPIO (0)
#define P17_FUNCTION_GPIO (0)
#define P18_FUNCTION_GPIO (0)
```

```
#define P19_FUNCTION_GND (0)
#define P20_FUNCTION_GPIO (0)
#define P21_FUNCTION_GPIO (0)
#define P22_FUNCTION_GPIO (0)
#define P23_FUNCTION_GPIO (0)
#define P24_FUNCTION_GPIO (0)
#define P25_FUNCTION_GPIO (0)
#define P26_FUNCTION_GPIO (0)
#define P27_FUNCTION_GPIO (0)
#define P28_FUNCTION_GND (0)
#define P29_FUNCTION_GPIO (0)
#define P30_FUNCTION_GPIO (0)
#define P31_FUNCTION_GPIO (0)
#define P32_FUNCTION_GPIO (0)
#define P33_FUNCTION_GPIO (0)
#define P34_FUNCTION_GPIO (0)
#define P35_FUNCTION_GPIO (0)
#define P36_FUNCTION_GPIO (0)
#define P37_FUNCTION_GND (0)
```

6.1.3 GPIO Functions

The simplest method to program the GPIO pins is through direct assignment statements, such as **`NBPKBD_J1[3] = 1;`** . Additional functions are shown below.

```
void function( int ft );        // Set pin function
void set( BOOL = TRUE );        // Set output high
void clr() { set( FALSE ); };   // Set output low
BOOL read();                    // Read pin hi/low state
void hiz() { read(); };         // Set output to tristate
void drive();            // Turn output on (opposite of tristate)
```

Examples:

```
// Initialize pins to GPIO
NBPKBD_J1[2].function( P2_FUNCTION_GPIO );
NBPKBD_J1[3].function( P3_FUNCTION_GPIO );
NBPKBD_J1[4].function( P4_FUNCTION_GPIO );
NBPKBD_J1[5].function( P5_FUNCTION_GPIO );
NBPKBD_J1[6].function( P6_FUNCTION_GPIO);
NBPKBD_J1[7].function( P7_FUNCTION_GPIO);
NBPKBD_J1[8].function( P8_FUNCTION_GPIO);
NBPKBD_J1[9].function( P9_FUNCTION_GPIO);

// Most common usage
NBPKBD_J1[2].read();   // Read input value (BOOL)
BOOL b = NBPKBD_J1[2]; // Read input value (BOOL)
NBPKBD_J1[3] = 1;      // Set output high
```

```
NBPKBD_J1[4] = 0;      // Set output low

// Additional functions
NBPKBD_J1[5].set();   // Set output high
NBPKBD_J1[6].clr();   // Set output low
NBPKBD_J1[7].hiz();   // Set output to tristate
NBPKBD_J1[8].drive(); // Enable output drive (from tristate)
```