NetBurner Blade Board Reference Guide

NBPKBM-100 Multi I/O Blade

Revision 1.1
May 2, 2008
Released

Table of Contents

1. Introduction

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

- 8 A/D 12-bit A/D channels with programmable input ranges: +/-10V, +10V, +/-5V and 5V. These signals may also be used as digital inputs.
- 16 digital I/O lines, jumper selectable to 3.3V or 5V.
- 2 16-bit DAC outputs, 0 to 4.096V.

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

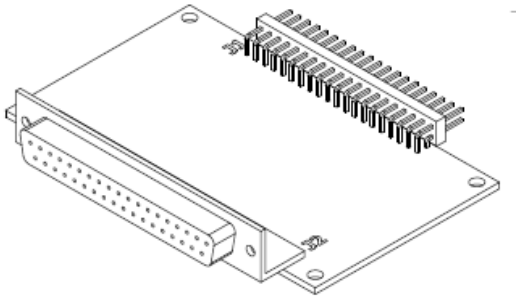
2. Additional Documentation

- PK70 (Hard Copy) Quick Start Guide
- PK70 Hardware Manual, 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 NBPKBM 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 NBPKBM to the PK70 interface connector.

To install the NBPKBM, 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.

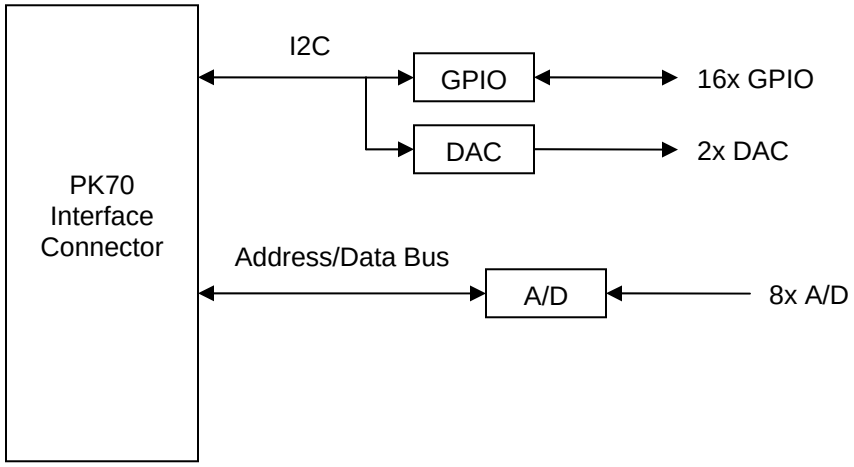
4. Hardware

The NBPKBM 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:

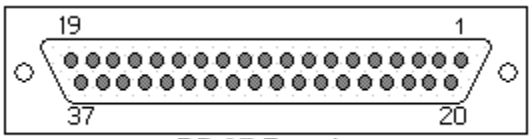
Function	Description
GPIO	Texas Instruments PCA9539DWR I2C 16-Bit I/O Expander
DAC	Texas Instruments DAC8571IDGKR I2C 16-Bit Digital to Analog Converter
A/D	Maxim MAX197BCAI+ 8 Channel Multi-range 12-Bit A/D with parallel bus interface

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	Selected GPIO supply voltage, 3.3V or 5.0V
10	Analog GND	29	DAC Output, or Digital Output
11	DAC Output, or Digital Output	30	GND
12	ADC or Digital Input	31	GND
13	ADC or Digital Input	32	GND
14	ADC or Digital Input	33	GND
15	ADC or Digital Input	34	GND
16	ADC or Digital Input	35	GND
17	ADC or Digital Input	36	GND
18	ADC or Digital Input	37	GND
19	ADC or Digital Input		

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

6. Software Programming

The programming interface for the NBPKBM 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\NBPKBM.h and \Nburn\PK70\system\NBPKBM.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

This section covers the dedicated GPIO pins (not DAC or A/D pins used as GPIO).

6.1.1 Dedicated GPIO Voltage Levels

The dedicated GPIO pins can be configured as 3.3V or 5V logic. The selection is made on the NBPKBM 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\NBPKBM.h. They are used as parameters in the Pins Class ***function()*** member function.

```
// Dedicated GPIO pins
#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 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)
```

```
// A/D pins that can be used as inputs
#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_GPIO (0)
```

6.1.3 GPIO Functions

The simplest method to program the GPIO pins is through direct assignment statements, such as **`NBPKBM_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
NBPKBM_J1[2].function( P2_FUNCTION_GPIO );
NBPKBM_J1[3].function( P3_FUNCTION_GPIO );
NBPKBM_J1[4].function( P4_FUNCTION_GPIO );
NBPKBM_J1[5].function( P5_FUNCTION_GPIO );
NBPKBM_J1[6].function( P6_FUNCTION_GPIO);
NBPKBM_J1[7].function( P7_FUNCTION_GPIO);
NBPKBM_J1[8].function( P8_FUNCTION_GPIO);
NBPKBM_J1[9].function( P9_FUNCTION_GPIO);

// Most common usage
NBPKBM_J1[2].read();  // Read input value (BOOL)
NBPKBM_J1[3] = 1;     // Set output high
NBPKBM_J1[4] = 0;     // Set output low

// Additional functions
NBPKBM_J1[5].set();   // Set output high
NBPKBM_J1[6].clr();   // Set output low
NBPKBM_J1[7].hiz();   // Set output to tristate
NBPKBM_J1[8].drive(); // Enable output drive (from tristate)
```

6.2 Digital to Analog (DAC) Pins

The 16-bit DAC has a programming range of 65535 counts, and an output voltage range from 0 – 4.095VDC. Initialize the DAC pin to specify the DAC function and the programming parameter: integer or floating point.

6.2.1 DAC Definitions

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

```
// DAC pins that can be used as outputs
#define P11_FUNCTION_GPIO (0)  // Use DAC as digital output
#define P11_FUNCTION_DACN (1)  // Set DAC value as 0-65535 int
#define P11_FUNCTION_DACV (2)  // Set DAC value as 0-4.095V float

#define P29_FUNCTION_GPIO (0)
#define P29_FUNCTION_DACN (1)
#define P29_FUNCTION_DACV (2)
```

6.2.2 DAC Functions

Examples:

```
// Initialize DAC pin with floating point assignments
NBPKBM_J1[11].function( P11_FUNCTION_DACV );

// Initialize DAC pin with integer assignments
NBPKBM_J1[29].function( P29_FUNCTION_DACN );

NBPKBM_J1[11] = 4.095;     // 4.095 volts
NBPKBM_J1[29] = 1024;      // 1024 DAC counts

// Configure a DAC pin as a general purpose output
NBPKBM_J1[11].function( P11_FUNCTION_GPIO);
NBPKBM_J1[11] = 0;
NBPKBM_J1[11] = 1;
```

6.3 Analog to Digital (A/D) Pins

The 12-bit A/D provides a resolution of 4095 counts. It has a programmable input voltage range of 0-5V, +/-5V, 0-10V and +/-10V. Before using an A/D input, you must initialize the pin to specify the input voltage range, and whether you want to read the number of counts, or a floating point representation of the measured voltage.

6.3.1 A/D Definitions

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

```
#define P12_FUNCTION_GPIO (0)      // Use ADC as digital input
#define P12_FUNCTION_ADN0_5 (1)    // 0-5VDC, read as 0-4095 counts
#define P12_FUNCTION_ADV0_5 (2)    // 0-5VDC, read as 0-5V float
#define P12_FUNCTION_ADN0_10 (3)   // 0-10VDC, read as 0-4095 counts
#define P12_FUNCTION_ADV0_10 (4)   // 0-10VDC, read as 0-10V float
#define P12_FUNCTION_ADNM5_5 (5)   // +/-5VDC, read as 0-4095 counts
#define P12_FUNCTION_ADVM5_5 (6)   // +/-5VDC, read as +/-5V float
#define P12_FUNCTION_ADNM10_10 (7) // +/-10VDC, read as 0-4095 counts
#define P12_FUNCTION_ADVM10_10 (8) // +/-10VDC, read as +/-10V float
```

6.3.2 A/D Functions

Examples:

```
// Initialize A/D pins with floating point results
NBPKBM_J1[12].function( P12_FUNCTION_ADV0_5 );
NBPKBM_J1[13].function( P13_FUNCTION_ADV0_10 );
NBPKBM_J1[14].function( P14_FUNCTION_ADV5_5 );
NBPKBM_J1[15].function( P15_FUNCTION_ADV10_10 );

// Initialize A/D pins with integer count results
NBPKBM_J1[16].function( P16_FUNCTION_ADN0_5 );
NBPKBM_J1[17].function( P17_FUNCTION_ADN0_10 );
NBPKBM_J1[18].function( P18_FUNCTION_ADN5_5 );
NBPKBM_J1[19].function( P19_FUNCTION_ADN10_10 );

float f = NBPKBM_J1[12];  // get float reading
int i   = NBPKBM_J1[16];  // get count reading

// To configure a A/D input as a general purpose input
NBPKBM_J1[12].function( P12_FUNCTION_GPIO);
BOOL Pin12 = NBPKBM_J1[12];
```