NetBurner
Networking in 1 Day!

NBPKBU-100CR Quad UART RS-232 Blade

Programming Reference Guide

Revision 1.1
April 2, 2009

Released

Table of Contents

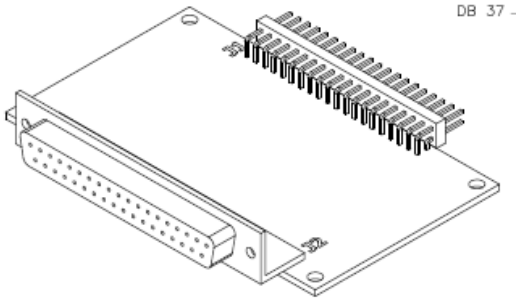
1. Introduction

The NBPKBU-100CR is a personality blade board for the NBPK70EX-100CR. A quad UART and RS-232 level shifters provide four serial input/output channels.

2. Installation

The NBPKBU-100CR 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 NBPKBU-100CR to the PK70 interface connector.

To install the NBPKBU-100CR, remove the PK70 cover, plug the dual row 40-pin right angle header (J1) into the 40-pin socket on the PK70, and then install the four 4-40 mounting screws. Finally, replace the PK70 cover and cover screws.

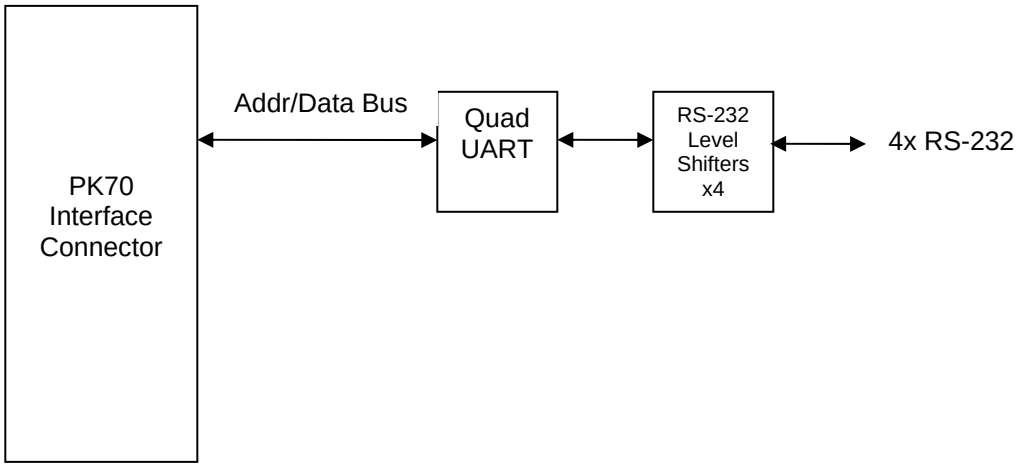


The software libraries are automatically installed with your PK70 development kit.

3. Hardware

The NBPKBU-100CR board interfaces to the PK70 through the 40-pin interface connector. This interface connector includes the address bus, data bus, QSPI, I^2C, clock, reset, chip selects, and interrupt signals. The blade board uses the parallel address/data bus to interface to a TI TL16C754BPN quad UART.

Block Diagram



4. Connector Pinouts

The NBPKBU-100CR's DB37 female connector has a pinout designed to follow the ribbon cable style for DB9 male crimp-type connectors. The table below shows the signal type for each pin number that comes out of the blade board's DB37 connector, and the corresponding pin number associated with one of the four DB9 connectors routed through the quad UART cable.

DB37 Pin	DB9-1 Pin	DB9-2 Pin	DB9-3 Pin	DB9-4 Pin	Signal
1				1	UART 3 – CD
2				2	UART 3 – RX
3				3	UART 3 – TX
4				4	UART 3 – DTR
5				5	GND
6			9		UART 2 – RI
7			8		UART 2 – CTS
8			7		UART 2 – RTS
9			6		UART 2 – DSR
10		1			UART 1 – CD
11		2			UART 1 – RX
12		3			UART 1 – TX
13		4			UART 1 – DTR
14		5			GND
15	9				UART 0 – RI
16	8				UART 0 – CTS
17	7				UART 0 – RTS
18	6				UART 0 – DSR
19	N O C O N N E C T I O N				
20				6	UART 3 – DSR
21				7	UART 3 – RTS
22				8	UART 3 – CTS
23				9	UART 3 – RI
24			5		GND
25			4		UART 2 – DTR
26			3		UART 2 – TX
27			2		UART 2 – RX
28			1		UART 2 – CD
29		6			UART 1 – DSR
30		7			UART 1 – RTS
31		8			UART 1 – CTS
32		9			UART 1 – RI
33	5				GND
34	4				UART 0 – DTR
35	3				UART 0 – TX
36	2				UART 0 – RX
37	1				UART 0 – CD

DB9 Signal Descriptions of Supplied DB37-to-Quad-DB9 Cable

Direction is with respect to the Data Terminal Equipment (DTE).

Pin	Name	Direction	Description
1	CD	In	Carrier Detect. Raised by DCE (Data Communications Equipment) when modem is synchronized.
2	RX	In	Receive Data.
3	TX	Out	Transmit Data.
4	DTR	Out	Data Terminal Ready. Raised by DTE when powered on.
5	GND	-	Ground.
6	DSR	In	Data Set Ready. Raised by DCE to indicate ready.
7	RTS	Out	Request to Send. Raised by DTE when it wishes to send; expects CTS from DCE.
8	CTS	In	Clear to Send. Raised by DCE in response to RTS from DTE.
9	RI	In	Ring Indicator. Set when incoming ring detected; used for auto-answer applications; DTE raised DTR to answer.

For the pinout of the 40-pin J1 header that connects to the PK70 module, please refer to the PK70 Hardware Manual (NBPK70.pdf), located in:

```
\Nburn\docs\platform\PK70
```

5. Application Programming Interface

The following functions enable programming of the NBPKBU-100CR. The source code for this library are located in \Nburn\PK70\include\NBPKQuadSerial.h and \Nburn\PK70\system\NBPKQuadSerial.cpp.

5.1 Serial Port Numbering Convention

The serial port numbering can be a bit confusing when looking at the hardware or schematics versus the software drivers. The ports are numbered from 1 to 4 when referencing hardware, including the quad UART cable. To maintain software conventions with other NetBurner serial drivers, the ports are numbered from 0 to 3 in the software drivers.

5.2 Baud Rate Calculation

When you specify a baud rate in the PK70QuadOpenSerial() function, the baud rate for each port is calculated based on the quad UART crystal frequency as shown below:

```
divider = 14745600 / ( baud * 16 )
```

For example, the divider for a baud rate of 115,200 bits per second is eight. You can achieve any baud rate with a whole number divider value.

5.3 Open a Serial Port

Description:

Opens a serial port and returns a file descriptor if successful.

Syntax:

```
int PK70QuadOpenSerial( int portnum,
     unsigned int baudrate, int stop_bits, int data_bits,
     parity_mode parity );
```

Parameters:

Parameter	Type	Description
portnum	int	UART to open; valid values are 0-3.
baudrate	unsigned int	Baud rate in bits per second.
stop_bits	int	Number of stop bits; valid values are 1 and 2.
data_bits	int	Number of data bits; valid values are 5-8.
parity	parity_mode	Valid values are eParityNone, eParityOdd, and eParityEven.

Returns:

Value	Description
(fd > 0)	File descriptor associated with the opened serial port if successful.
-1	SERIAL_ERR_NOSUCH_PORT
-3	SERIAL_ERR_PORT_ALREADYOPEN
-4	SERIAL_ERR_PARAM_ERROR (returned if stop/data bit value or parity mode is invalid)

A simpler version of this function called PK70QuadSimpleOpenSerial() is also available. The only parameters required are the port number and baud rate. The stop bits, data bits, and parity are automatically set to 1, 8, and eParityNone, respectively.

```
int PK70QuadSimpleOpenSerial( int portnum,
     unsigned int baudrate );
```

5.4 Close a Serial Port

Description:

This function closes a serial port that is currently open.

Syntax:

```
int PK70QuadSerialClose( int portnum );
```

Parameters:

Parameter	Type	Description
`portnum`	int	UART to close; valid values are 0-3.

Returns:

Value	Description
0	UART successfully closed.
-1	SERIAL_ERR_NOSUCH_PORT
-2	SERIAL_ERR_PORT_NOTOPEN

5.5 Software Flow Control

Description:

These functions enable and disable the sending (Rx flow control) or acknowledgement (Tx flow control) of XON/XOFF flow control characters.

Syntax:

```
void PK70QuadSerialEnableTxFlow( int port, int enab );
void PK70QuadSerialEnableRxFlow( int port, int enab );
```

Parameters:

Parameter	Type	Description
`port`	int	UART to enable/disable flow control on; valid values are 0-3.
`enab`	int	'0' disables flow control; any non-zero value will enable it.

Returns:

Nothing to return.

5.6 Hardware Flow Control

Description:

These functions enable and disable the sending of request-to-send (RTS) signals to regulate incoming data (Rx flow control) or receiving of clear-to-send (CTS) signals to throttle transmission (Tx flow control).

Syntax:

```
void PK70QuadSerialEnableHwTxFlow( int port, int enab );
void PK70QuadSerialEnableHwRxFlow( int port, int enab );
```

Parameters:

Parameter	Type	Description
`port`	int	UART to enable/disable flow control on; valid values are 0-3.
`enab`	int	'0' disables flow control; any non-zero value will enable it.

Returns:

Nothing to return.

5.7 Send Break

Description:

Sets a break in the UART transmission for a given period of time.

Syntax:

```
void PK70QuadSendBreak( int port, DWORD time );
```

Parameters:

Parameter	Type	Description
`port`	int	UART to set the break transmission on; valid values are 0-3.
`time`	DWORD	The length of time that the transmission break will occur, in ticks per second (20 ticks = 1 second by default).

Returns:

Nothing to return.

5.8 Read State of Carrier Detect

Description:

Gets the current state of the carrier detect (CD) pin from a specified UART.

Syntax:

```
BOOL PK70QuadGetCD( int port );
```

Parameters:

Parameter	Type	Description
port	int	UART associated with the CD pin to read; valid values are 0-3.

Returns:

Value	Description
TRUE	Carrier Detect signal is asserted.
FALSE	Carrier Detect signal is either negated or the given port is invalid.

5.9 Read State of Ring Indicator

Description:

Gets the current state of the ring indicator (RI) pin from a specified UART.

Syntax:

```
BOOL PK70QuadGetRI( int port );
```

Parameters:

Parameter	Type	Description
port	int	UART associated with the RI pin to read; valid values are 0-3.

Returns:

Value	Description
TRUE	Ring Indicator signal is asserted.
FALSE	Ring Indicator signal is either negated or the given port is invalid.

5.10 Read State of Data Set Ready

Description:

Gets the current state of the data set ready (DSR) pin from a specified UART.

Syntax:

```
BOOL PK70QuadGetDSR( int port );
```

Parameters:

Parameter	Type	Description
port	int	UART associated with the DSR pin to read; valid values are 0-3.

Returns:

Value	Description
TRUE	Data Set Ready signal is asserted.
FALSE	Data Set Ready signal is either negated or the given port is invalid.

5.11 Read State of Clear to Send

Description:

Gets the current state of the clear-to-send (CTS) pin from a specified UART.

Syntax:

```
BOOL PK70QuadGetCTS( int port );
```

Parameters:

Parameter	Type	Description
`port`	int	UART associated with the CTS pin to read; valid values are 0-3.

Returns:

Value	Description
TRUE	Clear-to-Send signal is asserted.
FALSE	Clear-to-Send signal is either negated or the given port is invalid.

5.12 Write State of Data Terminal Ready

Description:

Manually sets or clears the data terminal ready (DTR) pin for a specified UART.

Syntax:

```
void PK70QuadSetDTR( int port, BOOL val );
```

Parameters:

Parameter	Type	Description
`port`	int	UART associated with the DTR pin to set/clear; valid values are 0-3.
`val`	BOOL	“TRUE” asserts the signal; “FALSE” negates it.

Returns:

Nothing to return.

5.13 Write State of Request to Send

Description:

Manually sets or clears the request-to-send (RTS) pin for a specified UART.

Syntax:

```
void PK70QuadSetRTS( int port, BOOL val );
```

Parameters:

Parameter	Type	Description
`port`	int	UART associated with the RTS pin to set/clear; valid values are 0-3.
`val`	BOOL	“TRUE” asserts the signal; “FALSE” negates it.

Returns:

Nothing to return.