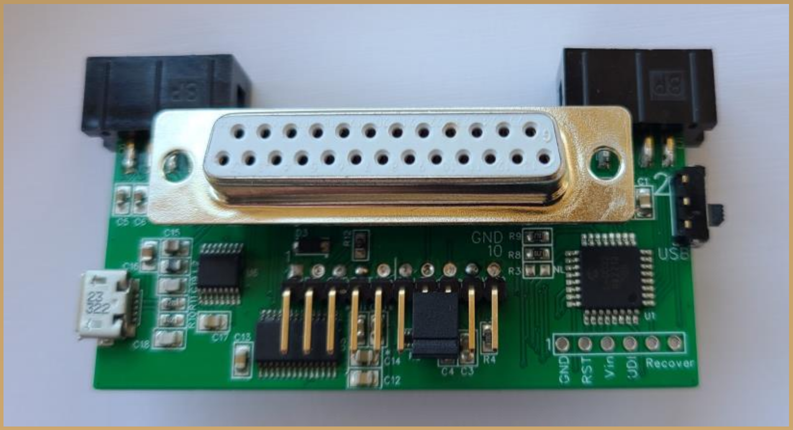
HEXBUS Adapter Manual



Version 1.1

31 March 2024

Document Changes 4

Software Release Notes 4

Introduction 5

HEXBUS adapter functions, features and restrictions 6

Using This Manual 7

PART I: Quick Start Guide 9

- Use with the TI-74 BasiCalc 10
- Use with the CC-40 Compact Computer 12
- Updating HEXBUS adapter Firmware using the BPD+ drive 14
- Troubleshooting 14

PART II: Using the HEXBUS adapter 15

- Commands supported using device ID 200 (HEXBUS adapter) 16
- Commands supported on the drive device ID (BPD+) 16

PART III: Command-Line Interface (CLI) 19

- Using CoolTerm™ with the HEXBUS adapter from a Mac/PC 20
- List of Commands with description 22
  - bye – Terminate CLI session 22
  - factory – Factory reset Command 22
  - help – Help Command 22
  - info – HEXBUS adapter Information 22
  - patch –Connect USB-Serial port directly 23
  - reboot – Reboot HEXBUS adapter module 23
  - reset – Reset HEXBUS adapter Drive 23
  - set – Set Command options 24
  - serial – Set the defaults for the serial port on the HEXBUS adapter module 25
  - usb – Set the defaults for the USB port on the HEXBUS adapter 26

Part IV: Understanding the HEXBUS adapter 27

- References and Background Reading 27

Theory of Operation 28

- Hardware Operation 28
- Firmware Operation 33

Annex A: Firmware Updates 34

- Installing the HEXBUS adapter Bootloader 34

Updating HEXBUS adapter Firmware using the BPD+ SD card......................................34

Recovering a ‘bricked’ HEXBUS adapter. 35

Annex B: AVR port usage 36

HEXBUS adapter port usage...36

Annex C: HEXBUS adapter Part List.. 37

Annex D: PCB 39

HEXBUS adapter PCB 39

Annex E: Open-source license reproductions 40

Bootloader for Backpack+ Drive...40

Document Changes

Date	Version	Changes
15 March 2024	1.0	Original draft

Software Release Notes

- V1.0

IMPORTANT: Parts of the HEXBUS adapter firmware use Open Source software that was/is available on Github or other web sites. The license agreements for any Open Source software are reproduced with links to those relevant parts in the Annex. As always, the web pages should be consulted for more up-to-date and definitive information.

1. Although not used in the HEXBUS adapter the optiboot_dx software did help with the AVR boot loader. https://github.com/SpenceKonde/DxCore/tree/master/megaavr/bootloaders/optiboot_dx

Welcome to the HEXBUS adapter User Manual

Introduction

The ability to store computer files on contemporary media such as Secure Digital (SD) cards has become an almost standard feature on modern computers. Additionally, most contemporary machines have an USB or serial port. In general, these benefits are somewhat lacking for older computers such as Texas Instruments™(TI) TI-74 BasiCalc (1985) and the earlier CC-40 Compact Computer (1983). Both TI machines came with an innovative, for the time, digital interface called the DockBus™ or Hexbus™[1] respectively (Hexbus will be used to refer to both the DockBus and Hexbus). The two busses use identical protocols although the physical format of the connector is different. The Hexbus can daisy chain up to 11 devices to provide connectivity to such Hexbus based peripherals as:

- HX-1000 Printer/Plotter
- HX-1010 Printer 80
- HX-2000 Wafer tape drive
- HX-3000 RS-232 interface
- HX-3100 Modem
- HX-3200 Centronics Printer Interface
- HX-5102 Disk Drive/Controller
- Quick Disk QD-02

Most of the Hexbus peripherals were never delivered in production quantities or are now extremely rare and hard to find. Some Hexbus peripherals never left the prototype stage. There are, of course, a few more modern Hexbus peripheral solutions available for these vintage machines that can provide program/data storage or serial interfaces. The solutions range from commercial offerings or open-source projects that are readily available online (e.g. GitHub or Instructables).

The HEXBUS adapter described in this User Manual is a simple, easy-to-use modern interface module that provides three connections from the Hexbus interface to a serial port, USB port and an optional SD based storage drive – Backpack Drive+ (BPD+). When used with the optional BPD+, the serial port is occupied, of course another HEXBUS adapter could be daisy chained to provide a serial port. Raw Hexbus protocol data is sent to the serial port when used for the storage device access, this data could be sent via a null serial cable to a local PC/MAC app to provide storage functions for the Hexbus device. The PC/MAC app is beyond the scope of this manual.

The HEXBUS adapter primarily mimics, as closely as possible, the capabilities of the original Hexbus serial port (HX-3000). The addition of the USB port allows connections to be made to more modern

[1] The TI-99/A home computer series also supports the Hexbus.

devices that support a USB interface. The form factor of the HEXBUS adapter is such that it can plug directly into the DockBus port on the TI-74 or be connected by an 8-pin ribbon cable to the CC-40.

The HEXBUS adapter provides:

- A Serial interface capable of running up to 38400bps, device ID 27
- Serial port control lines (RTS, CTS, DSR, DTR), not supported directly
- A USB port capable of running up to 38400bps, device ID 26
- USB power option, selected via switch
- DockBus power option, selected via switch
- Program/data storage using the optional BPD+, device ID 100
- A Command Line Interface for control and management of the adapter
- Ability to change the default serial port settings
- Device IDs can be modified as required, permits easier daisy chaining
- Command port to control/configure the adapter from the host computer, device ID 200
- Drive Command port to control/configure the BPD+ from the host computer

HEXBUS adapter functions, features and restrictions

To gain maximum benefit from the HEXBUS adapter it is recommended that users read the HX-3000 user manual and the BASIC reference manuals for the TI-74 and CC-40. The HEXBUS adapter supports most of the HX-3000 features as documented in the manual (see *References and Background Reading)*. The optional BPD+ supports most of the BASIC commands that access the storage media. Some commands are not supported such as cataloging of the drive. The device IDs of the three sections of the HEXBUS adapter use default IDs, however they can all be changed to allow daisy chaining with other HEXBUS adapters and extant Hexbus peripherals.

The table below lists the BASIC statements/features supported by the HEXBUS adapter. Some features are not supported because they cannot be applied to an SD card and others because they are no longer valid when using the HEXBUS adapter.

Command	Comments
`CALL IO`	Only supported for CALL IO (200,5)
`CLOSE`	Supported with DELETE qualifier
`DELETE`	Supported to delete files.
`EOF`	Supported.
`FORMAT`	Not Supported. SD Card cannot be formatted.
`INPUT`	Supported
`IO`	Supported, see CALL.
`LINPUT`	Supported
`LIST`	Supported.
`LOAD`	Supported, see CALL
`OLD`	Supported.
`OPEN`	Supported.
`PRINT`	Supported.
`RESTORE`	Supported.
`RUN`	Supported.
`SAVE`	Supported.
`VERIFY`	Supported.
BUS RESET	Supported, performed on power ON or OFF.

BASIC Commands supported by the HEXBUS adapter

When used with the optional BPD+ it is only possible to have one open SD-card file at a time.

If the drive is set to device ID 1 the BPD+ will mimic the Wafertape drive, storing the record length with the file. Otherwise, the drive assumes that any sequential or relative access use in a BASIC program correctly sets the record size, no checking is performed on the requested and actual record size.

It should be noted that TI defined exclusive blocks of device IDs in their original specification for Hexbus peripherals. TI expected that developers of Hexbus peripherals would request device IDs for their new devices or use those predefined by TI, consequently some applications expect certain device IDs related to real world peripherals. The HEXBUS adapter allows the IDs to be changed to either avoid these definitions or to meet specific device IDs.

Using This Manual

The “Quick Start” section contains the basic instructions to set up and use your HEXBUS adapter. It is suitable for use with the TI-74 and CC-40 series of machines. It has not been tested with the TI-99/A range of machines or with the TI-95. The section also describes additional features on the HEXBUS adapter that allow configuration/control of the HEXBUS adapter.

The “Using the HEXBUS adapter” section describes how to use the HEXBUS adapter with the TI-74 and CC-40. In general, most of the BASIC statements/commands that are supported by the TI machines are fully implemented by the HEXBUS adapter and optional BPD+. It is assumed that the user is familiar with the operations of the TI-74 or CC-40, this manual is intended to introduce the HEXBUS adapter.

The “Command Line Interface” (CLI) section describes the use of the "CLI" mode and explains how to issue commands to the HEXBUS adapter. Users should, at a minimum, familiarize themselves with the available commands as it can ease configuration when multiple HEXBUS adapters are to be daisy chained.

The “Theory of Operation” section describes the hardware and firmware operation in detail, including procedures for programming new firmware.

PART I: Quick Start Guide

This section provides a series of quick start guides for setting up and using the HEXBUS adapter. Items covered include:

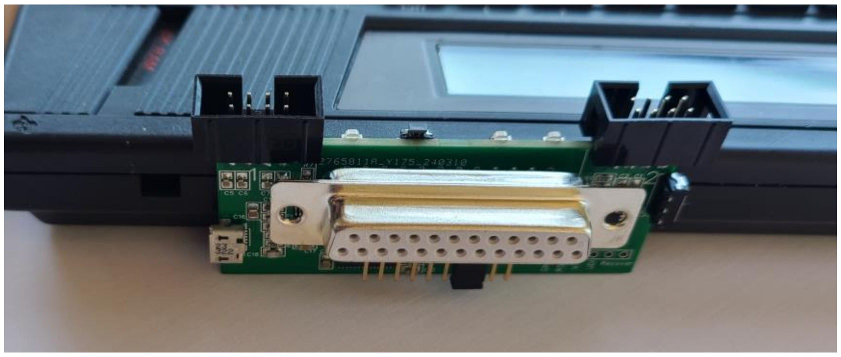
- TI-74 – set up for simple data storage access
- CC-40 – set up for simple data storage access
- Updating the HEXBUS adapter firmware
- Additional features of the HEXBUS adapter to allow control and configuration
- Simple troubleshooting of the HEXBUS adapter

Depending on the configuration of your laptop, some of these steps might not be required (e.g. previous use of CoolTerm or configuration of the adapter). Loading of the firmware is only required for modules that you have constructed; complete units will come already loaded with firmware **--** see Annex A. Updating the HEXBUS adapter is a necessary feature when new firmware is available.

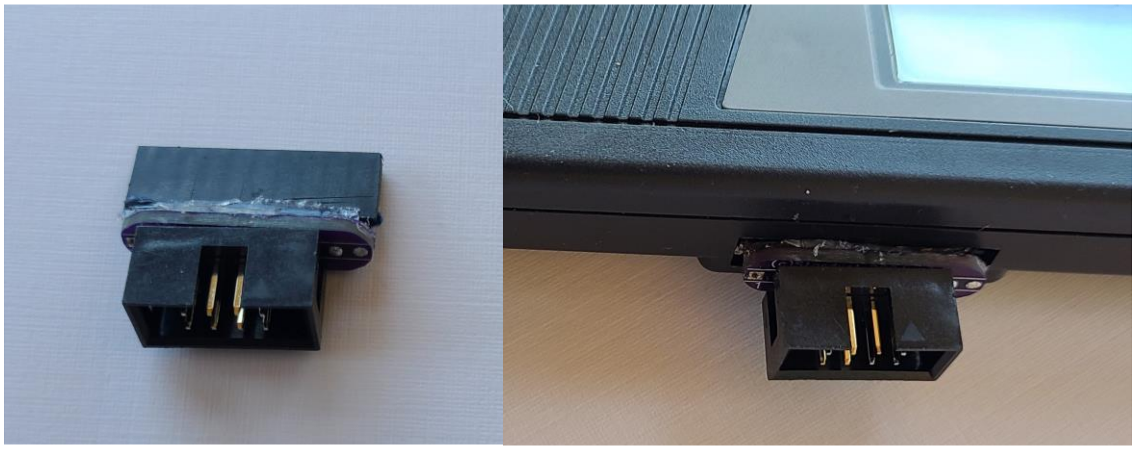
If the optional BPD+ is to be used with the HEXBUS adapter it is recommend that the user have access to the BPD+ User Manual.

Use with the TI-74 BasiCalc

Step	Operation
1	The HEXBUS adapter must always be powered when used with the TI-74. It can be powered either by the USB port or directly from the TI-74. It is recommended that USB power be used as much as possible so as not to drain the TI-74 battery. Select the power source using the slide switch. Set to USB (down) if using USB power or up (towards the LEDs) if using power from the TI-74. If the optional BPD+ is to be used, then it should be plugged into the serial port at this time. However, before doing so the `mode` option on the BPD+ should be set to `hex`. Use `set mode hex` on the BPD+ CLI. This operation will have to be performed using another computer with a serial port.
2	Make sure the TI-74 is powered off and then insert the 10-pin connector onto the 10-pin header on the rear of the TI-74, with the LEDs uppermost. The HEXBUS adapter should be nearly flush with the rear of the TI-74.
3	Power on the TI-74 using the ON key. The Red LED (Hexbus) should blink on for about 1 second. This indicates that the HEXBUS adapter is connected correctly. Powering on will reset the state of the HEXBUS adapter. The Green LED (serial port) and Yellow LED (USB port) may also blink. Powering OFF the TI-74 will cause the HEXBUS adapter to reset and the Green and Yellow LEDs should also blink. NOTE: If the power source has been set to use the TI-74 battery power then even when the TI-74 is 'OFF' the HEXBUS adapter will be powered. It is recommended that the HEXBUS adapter be removed if the TI-74 is off.
4	The HEXBUS adapter is now ready for use.



HEXBUS adapter installed directly on the TI-74

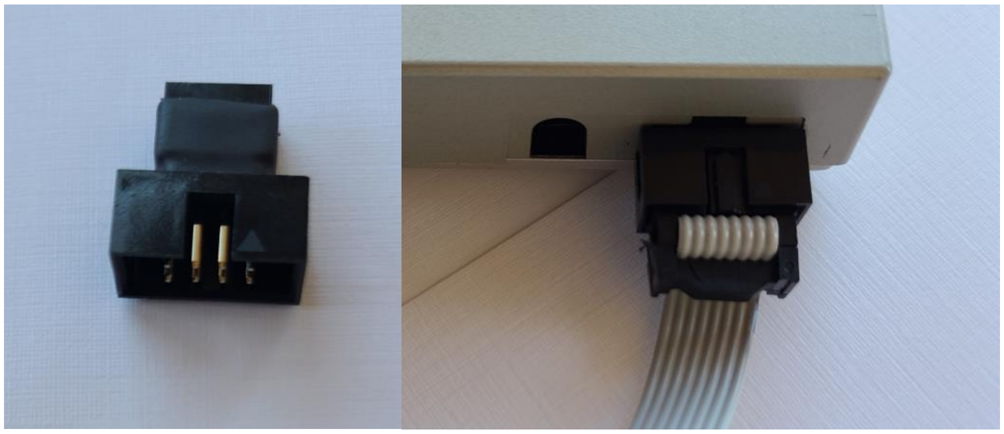


Dock 2 Hexbus (DOCK2HEX) Adapter installed

If the DOCK2HEX adapter is available, then it can be used to provide an 8-pin Hexbus header to plug in the HEXBUS adapter via a flat ribbon cable instead of directly to the TI-74. The 8-pin Hexbus header could also be used to plug in other Hexbus peripherals.

Use with the CC-40 Compact Computer

Step	Operation
1	The CC-40 does not provide any power over the Hexbus interface therefore the HEXBUS adapter must be powered by the USB port. Set the power select switch to USB (down) and insert a USB cable that is powered. If the optional BPD+ is to be used, then it should be plugged into the serial port at this time. However, before doing so the `mode` option on the BPD+ should be set to `hex`. Use `set mode hex` on the BPD+ CLI. This operation will have to be performed using another computer with a serial port.
2	Make sure the CC-40 is powered off and then insert the 8-pin extender connector into the Hexbus port on the CC-40 (take care the internal connector is not very well supported), see pictures below. The extender allows the use of standard 8-pin IDC headers on the flat ribbon cable. Now connect one end of the 8-pin flat ribbon cable to the extender connector. The other end is then connected to one of the two Hexbus connectors on the HEXBUS adapter. The two connectors are identical and simply allow for the daisy chaining of other Hexbus peripherals. NOTE: Original Hexbus peripheral cables could be used but at this time they have not been tested with the HEXBUS adapter.
3	Power on the CC-40 using the ON key. The Red LED (Hexbus) should blink on for about 1 second. This indicates that the HEXBUS adapter is connected correctly. Powering on will reset the state of the HEXBUS adapter. The Green LED (serial port) and Yellow LED (USB port) may also blink. Powering OFF the CC-40 will cause the HEXBUS adapter to reset and the Green and Yellow LEDs should also blink.
4	The HEXBUS adapter is now ready for use.



Hexbus to IDC Adapter installed

NOTE: The Hexbus connector on the CC-40 is generally not very well braced internally (the PCB appears to float a little, with the pins of the connector slanted downwards), take GREAT care when installing this adapter. The connector may feel loose when installed, make sure the 8-pins have been engaged properly. **Remove** the adapter when not in use to avoid damage from accidental knocks.

Updating HEXBUS adapter Firmware using the BPD+ drive

1. Copy the new firmware file to the root of the BPD+ SD card. The firmware should be called `HEXBUSFW.BIN.` Rename any existing file as a backup in case of issues with the new firmware.

2. Connect the BPD+ to the HEXBUS adapter and connect the USB port to a suitable computer. See the section on using CoolTerm on page 18 for more information. Set the power switch to USB

3. Enter command: `set update on`

4. Reboot the HEXBUS adapter: `reboot<Enter>`
   The boot process should display the message "Normal Update" followed by "Processing File" on the CoolTerm screen.

5. Verify the new firmware version: `info<Enter>`

The Green and Red LEDs will also flash to provide a visual indication of the update progress. The LEDs on the BPD+ should also flash indicating it is providing the source file.

Troubleshooting

Red LED stays on continuously on HEXBUS adapter

- Check that the TI-74 or CC-40 is not hung up and try power cycling the computer. The display may show the I/O annunciator this indicates that the Hexbus operation has hung.

- Check the 8-pin flat ribbon cable is installed correctly and not flipped around. Never force the connections past the keys on the connectors.

The computer hangs when accessing the optional BPD+

- Check the BPD+ is powered on.

- Power cycle the computer to reset the Hexbus peripherals. The BPD+ LEDs should blink when the computer is powered on.

- The BPD+ might enter sleep mode and not have woken fully. Retry the operation. The HEXBUS adapter will attempt to wake the BPD+ but it might not have been successful.

- Check the battery on the BPD+. If the battery is too low, it will not access the SD card causing any disk-based operations to fail. See BPD+ manual on replacing the AA battery.

PART II: Using the HEXBUS adapter

The HEXBUS adapter supports two built-in interfaces: a serial port and USB port. The serial port supports most of the features found in the original RS-232 interface Hexbus peripheral (HX-3000). However, a few features such as overrun and parity errors are ignored. The USB port is a new port not found in the original TI Hexbus peripheral series. The USB port adopts all the features found on the serial interface.

All Hexbus commands sent to the drive are pre-pended with XX (capital ASCII X). This is used to signify the start of a command to the drive. The Hexbus commands contain a length field which can then be used to determine how many characters to read.

The control/management features described in this section are not part of the original TI serial interface or storage interfaces. They are unique additions to the HEXBUS adapter provided as convenience features that can be used to programmatically control the adapter or optional BPD+

The HEXBUS adapter supports a third interface the drive port, that in conjunction with the optional BPD+ can support file operations, for example: OLD, SAVE, RUN, PRINT, INPUT, LIST and VERIFY etc. The BPD+ uses an SD Card for file storage, it also has its own built-in CLI (Command-Line Interface). The BPD+ `HEX` mode supports a limited number of commands to perform some simple file management functions via the Hexbus interface. All the functions are also available via the BPD+ CLI.

The device IDs of the Serial port, USB port and Drive port can all be changed to support daisy chaining with multiple HEXBUS adapters or other Hexbus peripherals.

Finally, the HEXBUS adapter has one device ID dedicated to control/management. This is called the command port (200). The 200 device ID allows the HEXBUS adapter to be configured without resorting to the use of the CLI. The command port can change the IDs of the other three interfaces as well as cause a factory reset of the HEXBUS adapter. The factory reset returns all the settings to their 'out of the box' state. The command port cannot be used when the HEXBUS adapter is in a daisy chained mode.

In order to send commands to the HEXBUS adapter or BPD+ a command channel must first be opened. The command channel on the HEXBUS adapter is opened using the `OPEN` BASIC command:

```
OPEN #1,"200".
```

The file number can be set to anything in the range 1 to 255. Similarly, the command port to the BPD+ is opened using:

```
OPEN #1,"100"
```

In this case 100 is the current value of the device ID for the drive port, this value would need to change to the current device ID if it had been set to a different value. Once the port has been opened `PRINT`

`#device ID` or `INPUT #device ID` statements can be used. The `OPEN` command must be closed using the `CLOSE` statement when the command port is no longer required.

Commands supported using device ID 200 (HEXBUS adapter)

The following commands are supported by the HEXBUS adapter. If no error is returned the command was successful.

`usb <device ID>` this command changes the device ID of the USB port. The default value is 26.

`ser <device ID>` this command changes the device ID of the serial port. The default value is 27.

`drv <device ID>` this command changes the device ID of the drive port. The default value is 100.

After opening the command port as described above the PRINT statement is used to send the command. For example, to set the USB port to 30 use:

```
PRINT #1, "usb 30"
```

assuming the device ID was opened with file number #1. Commands can be combined in one print statement using '`,`' as a separator. For example:

```
PRINT #1, "usb 30,ser 29"
```

Using an IO call it is possible to factory reset the HEXBUS adapter to the default values.

```
call IO (200,5)
```

Commands supported on the drive device ID (BPD+)

The following commands are supported by the BPD+ in `HEX` mode. The BPD+ commands are aimed at either file or directory management on the BPD+ SD Card or using the RTC built into the BPD+. The commands can be sent to the BPD+ using the `PRINT #device ID` statements. Data can be received using the `INPUT #device ID` or `LINPUT #device ID` statements. The commands, except `tm` and `tmset`, can be combined in a single `PRINT` statement when separated by commas. If no error is returned the command was successful.

`cd <directory>` This command will change to the specified directory on the SD Card, if that directory exists.

`ls <directory>` This command will begin listing the current directory contents. This command must be followed by `LINPUT` statements to read each filename. The command returns one filename per call. At the end of the directory `EOF` will be set. The ASCII data returned is filename, file size, file type. File type is either F (filename) or D (directory).

`md <directory name>` This command will make the directory name as specified. The directory name must not already exist in the current directory and must be no more that 8 characters.

`rd <directory>` The command will delete an empty directory. If the directory is not empty and error will be returned.

`rn <old filename>  <new filename>` The command will rename the old filename to the new filename. The old filename must exist in the current directory and the new filename should not exist. An error is generated if the action cannot be completed. The filenames must be of the 8.3 format.

`tm` This command will request the time from the RTC on the BPD+. The time can be read using an `LINPUT` statement. The time is given as ASCII: day, month, year, hours, mins, secs. The current time is provided at each LINPUT call until the `CLOSE` statement is used.

`tmset` This command is used to set the time on the BPD+ RTC. After the command is sent the next statement must be a `PRINT` statement that includes the time in ASCII format: day, month, year, hours, mins, secs. The command must then be closed using a `CLOSE` statement.

If the BPD+ command port has already been opened (`OPEN #1,”100”)`, then the `PRINT` statement can be used to send the command. For example, to rename a file use:

```
   PRINT #1, “rn file1 file2”
```

This will rename file1 as file2 on the BPD+ SD Card. Combined commands could be sent as:

```
   PRINT #1, “cd ti74,md cc40,cd cc40”
```

This command will change to the ti74 directory, then create a new directory cc40 and finally change to the new directory. If any command fails any subsequent commands will not be executed.

The `tm` and `tmset` commands can be used to access and set the RTC on the BPD+ as follows:

```
   PRINT #1,”tmset”

   PRINT #1,”dy$,mo$,yr$,hr$,mn$,sc$”

   CLOSE #1
```

The `dy$, mo$, yr$, hr$, mn$, sc$` variables are the day, month, year (4digits), hour, minute and seconds respectively. The names of the variables can be the programmer’s choice. The RTC time can be read as follows:

```
   PRINT #1,”tm”

   LINPUT #1,”dy$,mo$,yr$,hr$,mn$,sc$”
   CLOSE #1
```

As above the variables represent the date and time.

PART III: Command-Line Interface (CLI)

The HEXBUS adapter has two primary states: the HEXIF state and CLI state. In the HEXIF state, the HEXBUS adapter will service HexBus Protocol Commands from the laptop. To access the HEXBUS adapter CLI from the USB port (or serial port dependent on the setting of console - see commands) the CoolTerm app is recommended see below for more information. Once connected, the CLI state is accessed by pressing `<Enter>` (CR) four (4) times. This action should bring up the 'C' followed by the '`$`' prompt. You can now issue CLI commands to the HEXBUS adapter.

If a command is completed successfully then generally the "`OK`" response will be returned. Other positive and negative responses are generated depending on the command and the CLI state.

If an unrecognized command is entered, the "`Unknown Command`" error will be generated. Check the spelling of the command to make sure the command is valid and that it is available in the version of firmware on the HEXBUS adapter. It is anticipated that new commands might be added in future and existing ones might be enhanced or others might be removed.

If an unexpected or invalid parameter is entered for a command, the "`Bad Parameter`" error will be generated. Check that the parameters are valid for the command and that they are formatted correctly.

Relevant commands can accept control keys to change their actions. The following control keys are recognized:

`Ctrl-C`	: terminates any command in progress
`Ctrl-S`	: resume listing
`Ctrl-Q`	: pause listing -- make sure pause is ended as the module will not return to TPDD state
`Ctrl-Z`	: close open file
`<space>`	: continue
`<enter>`	: continue

The example commands shown below use the following nomenclature:

<…>	: Required field(s)
`[…]`	: Optional field(s)
`|`	: alternate options
..	: range of values

All commands follow the same format: command, followed by a parameter list if required.

Using CoolTerm™ with the HEXBUS adapter from a Mac/PC

For convenience, the HEXBUS adapter CLI can be accessed from a Mac or PC through either the RS232 or USB connection using any serial terminal program. The port used for the CLI is based on the `console` setting, refer to the CLI commands for further information. The default `console` setting is `USB`. The program CoolTerm™ is tested and recommended. CoolTerm should be configured to access the correct *serial port* associated with the serial/USB cable plugged into the HEXBUS adapter. Configure the CoolTerm Serial Port as: baud rate: 38400, Data Bits: 8, Parity: None and Stop Bit 1. The DTR and RTS can be set ON. Most of the Terminal default modes will work however *"Enter Key Emulation*" should be set to CR only, and the *"Convert Non-Printable characters*" and *"Handle BS and DEL characters*" should be checked.

Other serial terminal programs can be used, but configuration details will vary and are not covered here.

Command Summary

Command	Parameters	Description
`bye`		Exit CLI State
`factory`		Reset to factory settings
`help`		Print list of commands
`info`		List information
`Patch`		Connect the serial and USB ports.
`reboot`		Reboot the module
`reset`		Reboot the module

`set`	`[update <OFF | ON> |`
	`sleep <0 – 60> |`
	`console <USB | Serial> |`
	`usb <1..199,201.255> |`
	`serial <1..199,201.255> |`
	`drive <1..199,201.255>]`
		Set various configuration options on module.

`serial`	`[<B= 50|75|110|300|600|1200|2400|`
	`4800|9600|19200|38400>,`
	`  | <D= 5 | 6 | 7 | 8>,`
	`  | <S= 1 | 2>,`
	`  | <P= O | E | S | M | N>,`
	`  | <C= Y | N>,`
	`  | <E= Y | N>,`
	`  | <N= 0..99>,`
	`  | <O= Y | N>,`
	`  | <R= N | C | L>,`
	`  | <T= R | C | W> ]`
		Set the DEFAULT configuration of the serial port.

`usb`	`See serial port configuration parameters.`	Set the DEFAULT configuration of the USB port.

List of Commands with description

bye – Terminate CLI session

```
bye
```

This command is used to return to the Hexbus interface state. Normally after 10s of inactivity the adapter will return to the Hexbus interface state automatically, this prevents the adapter being locked in the CLI state. The automatic return will NOT happen if the output in the CLI has been paused.

factory – Factory reset Command

```
factory
```

This command is used to restore the HEXBUS adapter to a factory condition; it will set all the parameters back to their default condition.

The BASIC command `call io (200,5)` can also be used to cause a factory reset, when the Hexbus is connected.

help – Help Command

```
help
```

This command is used to provide brief help information on the CLI commands available on the HEXBUS adapter.

Example:

```
$ help
Commands (see manual for details)
set usb serial info patch factory help
patch bye reset reboot
To reset via Hexbus use: cmd io(200,5)
usb or serial params
Baud Rate    B= opt
  opt = 50|75|110|300|600|1200|2400|4800|9600|19200|38400
Data bits    D= 5 | 6 | 7 | 8
Stop bits    S= 1 | 2
Parity       P= O | E | S | M | N
Check parity C= Y | N
Echo         E= Y | N
Nulls        N= 0..99
Data Overrun O= Y | N
CR or LF     R= N | C | L
Trans Type   T= R | C | W
$
```

info – HEXBUS adapter Information

```
info
```

This command is used to display information about the HEXBUS adapter. The version and build date will vary depending on the firmware loaded onto the adapter. The Hardware (HW) version is used to indicate the version of hardware being used.

The information for the HEXBUS adapter includes the board voltage; it should be in the range 4.5 – 5.1V dependent on the setting of the power select switch.

Example:

```
$ info
AVR Hexbus adapter V1.0
Main = 5.03V
Built Mar  8 2024 20:17:49  HW V:02
Serial_ID=27 USB_ID=26 Drive_ID=100 CMD_ID=200
Ser:42225361510018160145011500000000
$
```

patch –Connect USB-Serial port directly

`patch`

This command is used to interconnect the USB port and Serial port on the HEXBUS adapter. If the USB or Serial port is connected to a serial program on a PC/MAC (e.g. CoolTerm) then it is possible to issues commands. For example, the optional BPD+ could be connected to the Serial port and the USB port could be plugged into a suitable computer running CoolTerm. Commands can then be issued to the BPD+ from the connected computer.

The USB and Serial port data rates can be independently set, see the `serial` and `usb` commands below.

To exit this mode use `***` (i.e. three `*` in a row).

reboot – Reboot HEXBUS adapter module

`reboot`

This command is used to reboot the HEXBUS adapter. Generally, this command is used to initiate an update procedure.

reset – Reset HEXBUS adapter Drive

`reset`

This command is used to reboot the HEXBUS adapter. Generally, this command is used to initiate an update procedure.

set – Set Command options

```
set [update <OFF | ON> | sleep <0 – 60> |
serial <1..199,201..255> | usb <1..199,201..255> | drive
<1..199,201..255> | console <usb | serial>]
```

This command is used to set various configuration parameters on the HEXBUS adapter. If the command is entered without any parameters, then the current settings are displayed. The various settings can be concatenated into a space separated line if desired.

- `console`: sets the console serial port. This allows commands/information to be directed at either the `USB` or `serial` port. The default is the USB port.

- `serial`: sets the default port values for the serial interface (factory setting 27). The value can range from `1` to `255`, it cannot overlap the USB or drive port value or equal 200.
  The serial interface default value can also be changed via BASIC using the `OPEN` command and the `PRINT` command. E.g.,
  ```
  open #1,”200”
  print #1,”ser 30”
  close #1
  ```

- `USB`: sets the default port value for the USB interface (factory setting 26). The value can range from `1` to `255`, it cannot overlap the serial or drive port value or equal 200.
  The USB interface default value can be changed via BASIC using the `OPEN` command and the `PRINT` command. For example.
  ```
  open #1,”200”
  print #1,”usb 30”
  close #1
  ```

- `drive`: sets the default port value for the drive interface (factory setting 100) . The value can range from `1` to `255`, it cannot overlap the USB or serial port value or equal 200.
  The serial interface default value can be changed via BASIC using the `OPEN` command and the `PRINT` command. For example.
  ```
  open #1,”200”
  print #1,”drv 110”
  close #1
  ```

- `update`: sets the update feature `on` or `off`. The update feature allows the updating of the firmware on the HEXBUS adapter see Annex A for further details.

- `sleep`: sets the inactivity time before sleeping, the value can range from 0 to 60 mins. The 0 setting will turn off the sleep mode. The default value is 2 mins. Generally, it will wake when it receives a character on the serial interface, USB interface or Hexbus interface.

Example:

```
$ set console usb
OK
$ set usb 20
OK
$ set
Sleep=2 Console=USB Update=OFF
Serial_ID=27 USB_ID=20 Drive_ID=100 CMD_ID=200
$
```

serial – Set the defaults for the serial port on the HEXBUS adapter module

`serial [parameters listed below]`

This command is used to set the persistent default parameters for the serial port. The command takes the same parameters as the regular serial port command on the CC-40/TI-74 (shown below). Although the `Check Parity` and `Data Overrun` options are allowed, they cause no action on the HEXBUS adapter as the module does not perform these checks.

If the command is used without parameters, then the current settings are displayed.

With modern serial interfaces changing the default values should not be required. The `OPEN` command can be used to override these settings.

```
Baud Rate     B= 50|75|110|300|600|1200|2400|4800|9600|19200|38400
Data bits     D= 5 | 6 | 7 | 8
Stop bits     S= 1 | 2
Parity        P= O | E | S | M | N
Check Parity  C= Y | N
Echo          E= Y | N
Nulls         N= 0..99
Data Overrun  O= Y | N
CR or LF      R= N | C | L
Trans Type    T= R | C | W
```

Example:

```
$ serial b=19200,d=8
OK
$
```

usb – Set the defaults for the USB port on the HEXBUS adapter

```
usb [parameters listed above]
```

This command is used to set the persistent default parameters for the USB port. The command takes the same parameters as the regular serial port command on the CC-40/TI-74 (shown above in the `serial` command options). Although the `Check Parity` and `Data Overrun` options are allowed, they cause no action on the HEXBUS adapter as the module does not perform these checks.

With modern USB interfaces changing the default values should not be required. The `OPEN` command can be used override these settings.

Part IV: Understanding the HEXBUS adapter

This section of the document discusses the theory of operation HEXBUS adapter to provide further information if you want to build your own HEXBUS adapter or repurpose the module to support other devices. As the HEXBUS adapter features are provided entirely in firmware they should be easily updatable to correct problems or add new options.

The HEXBUS adapter module uses the Microchip AVR64DD32, this MCU has 64Kbyte Flash, 8K SRAM and 512B EEPROM. The HEXBUS adapter uses the UPDI pin for programming which should fit with most Microchip programming devices e.g., PIC 4 or Atmel – ICE.

References and Background Reading

This section provides some useful background reading to understand the HEXBUS adapter as well as the genesis of the module. There are probably many other solutions and references that can be found to aid in understanding if required, as well as act as inspiration to build something of your own.

1. TI-74 Programming Reference Guide
2. TI-74 Users Guide. This guide includes information on how to use the TI-74 and TI BASIC.
3. HX-3000 Texas Instruments RS232 Manual. This manual covers the use of the RS-232 interface.
4. CC40 user guide. This guide includes information on how to use the CC-40 and TI BASIC.
5. Texas Instruments Hexbus specification (This is a collection of specs for the Hexbus and peripherals). The document describes the Hexbus protocol as well as the requirements for the Serial interface, Wafertape, Modem and Printer. It also includes the specification for the TI Hexbus interface chip,
6. Microchip: AVR64DD32 Datasheet.
7. Texas Instruments: TRS3243E 3- to 5.5-V multichannel 600kbps RS-232 line driver/receiver with +/-15-kV IEC-ESD protection datasheet. Serial port interface.
8. FTDI FT230X USB to basic UART IC Version 1.5

Theory of Operation

This section describes the hardware and firmware of the HEXBUS adapter to assist with understanding the operational aspects of the device, especially if the unit is to be used in other applications beyond those outlined in this manual.

Hardware Operation

A block diagram of the HEXBUS adapter is shown in Fig 1. The module is designed to be as simple and low cost as possible while still providing useful and versatile functionality. As mentioned above the unit uses the Microchip AVR64DD32 set to run at 16MHz.

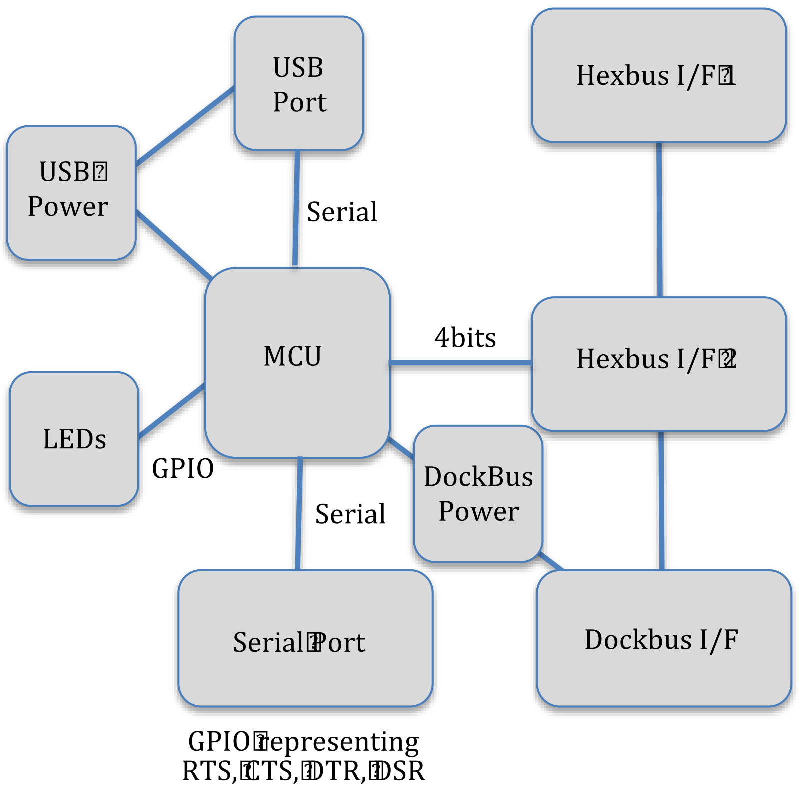


Figure 1: HEXBUS adapter Block Diagram

The MCU interfaces to 4 separate sections: Hexbus Interface, Serial Driver interface, USB Interface and LEDs. The HEXBUS adapter supports two power options: USB power from the USB interface and, if used with the TI-74, power from the DockBus. The power source is selected via a slide switch. In addition, the HEXBUS adapter has the option to provide power to one or both Hexbus 8-pin connectors via pin 6 of that interface. The option to send power to the connector is selected via a shorting jumper on the lower 10-pin connector. The power is taken from the selected power source so care should be taken if power is selected from the DockBus interface as this could drain the AA batteries. If using an ‘original’ Hexbus peripheral,

then the pin 6 power option **should NOT be used**.
NOTE: The USB interface does not provide power to the DockBus interface so cannot power the TI-74.

The Hexbus interface uses 6 GPIO pins of Port A on the MCU. The Port A pins are connected to the BAV, HSK and D0-D3 lines of the Hexbus interface. The BAV line is also used as an input that can wake up a sleeping MCU (The serial and USB interfaces can also wake the MCU). The Hexbus interface is a very simple handshake-based scheme that relies on a bus master to control the transfer of 4 bits of data at a time, two transfers are required to transfer a byte. The master also waits for the response from the peripheral. If the peripheral is slow or requires input, then it can hold the Hexbus master while the data to be transferred is prepared. Although the Hexbus specification allows different peripherals to become master of the bus this scheme is not supported by the HEXBUS adapter. The 6-pins are routed to the DockBus interface and both Hexbus interfaces.

The serial interface uses the TX/RX pins of UART1 on the MCU and 4 GPIO pins to provide the control lines RTS, CTS, DTS and DSR lines. The control pins currently provide no flow control except when used with the optional BPD+. The control pins are there to allow compatibility between the serial port software and HEXBUS adapter as well as future enhancements. A Texas Instruments TRS3243E IC provides the serial voltages and interface. In sleep mode the serial interface is shutdown to save power. A blinking green LED indicates activity on the serial port; however, the LED is also used to indicate other states (see below). The sleeping MCU can be taken out of sleep mode when a character is received on the serial line (The USB and Hexbus interfaces can also wake the MCU).

The USB interface uses the TX/RX pins of UART0 on the MCU. An FTDI FT230XS-R chip provides the protocol interface to the USB port. A blinking yellow LED indicates TX/RX activity on the USB port, this is tied to the CBUS1 line. The CBUS0 line is connected to the MCU to be used to wake up a sleeping MCU (The serial and Hexbus interfaces can also wake the MCU).

To reduce power use, the MCU can go to sleep after a set number of minutes of no activity. This feature can be controlled using the CLI. It is also possible to turn off sleep mode if desired. Blinking red and green LEDs (see table below) are used to indicate entry into the sleep state. The MCU will remain sleeping until either a new character is received on the serial port or a character on the USB is received or the request on the Hexbus interface is received at which point the MCU will wake and begin normal activity. As mentioned above while in sleep mode all the pins on the serial port are disconnected and will float to almost 0V.

A sleep time from 3 – 10 mins is recommended. When the MCU is in the sleep state the current drain decreases to <1mA. During normal operation the average current drain is ~30mA depending on the precise activity.

Two LEDs (Green and Red) are driven directly from GPIO pins on the MCU. A third Yellow LED is driven by the USB interface. The LEDs are used to visually signal various states of the HEXBUS adapter.

Their primary use is to indicate activity on the serial port with a blinking Green LED and activity on the Hexbus with a blinking Red LED, as described above. On power up the Green and Red LEDs will blink alternatively to indicate successful power on. If the Red LED remains illuminated, then a problem exists with the Hexbus interface. The table below lists some of the states indicated by the LEDs.

Green LED	Red LED	Yellow LED	Comment
blinking	X	X	Serial port activity
X	blinking	X	Hexbus Activity
X	steady on	X	Hexbus problem
X	X	blinking	USB Activity
on on off	on off off	X X X	Entering sleep mode
on ~2s off ~2s	on ~2s off ~2s	X X X X	Cycle repeats for 15s. Fatal error Possible low battery or SD card issue
During Update procedures			
on blink	on off	blink blink	Update failed, file not found
on off	on blink	blink blink	Update succeeded
rapid blink	off	blink	Error occurred during update. Repeat.

The programming pin header can be used to reprogram the MCU on the HEXBUS adapter using either the Microchip PIC 4 or Atmel ICE. Pin 6 is used by the bootloader to determine whether an update should be forced. When pin 5 and 6 are shorted using a shorting header no action will be taken on reboot or power cycle. If pin 5 and 6 are not shorted, then an attempt will be made to update the firmware from the BPD+ (see Annex A). A shorting pin is always required to make sure it is available when needed. If the shorting header were not required for normal operation, then finding one would be impossible when required!

Pin Number	Description
1	GND
2	Reset
3	+5V
4	UDI
5	Recover
6	GND

Table 1: Programming header pin descriptions

DB-25 Pin	Description
2	Tx
20	DTR
3	Rx
4	RTS
5	CTS
6	DSR
7	GND

Table 2: Serial port pin descriptions from the HEBUS adaper

DockBus Pin	Description
1	+6V
2	Pwr In
3	D0
4	D1
5	D2
6	D3
7	HSK
8	BAV
9	RESET-
10	GND

Table 3: DockBus pin description looking into the TI-74 connector

DockBus Pin	Description
1	Pin 6 Hexbus 1
2	Pwr
3	GND
4	RESET
5	+5V
6	UDI
7	Recover
8	GND
9	Pwr
10	Pin 6 Hexbus 2

Table 4: 10 pin right angle header

DockBus Pin	Description
1	D0
2	D1
3	BAV
4	GND
5	HSK
6	PWR*
7	D2
8	D3

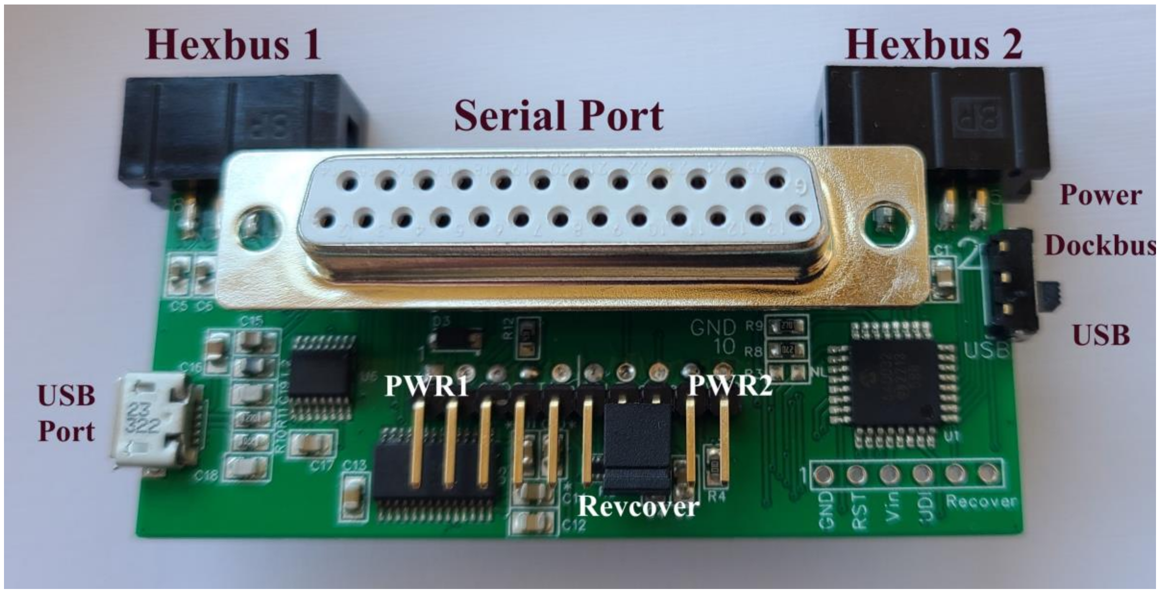
*Pin 6 PWR pin is not standard used by HEXBUS Adapter

Table 5: Hexbus pin description

```
 _______
|4 3 2 1|
|8 7 6 5|
 -------
```

Table 6: Hexbus pin layout looking into the CC40 connector

Figures 2 shows the completed board for reference. Note the board color and parts might vary depending on supplier.



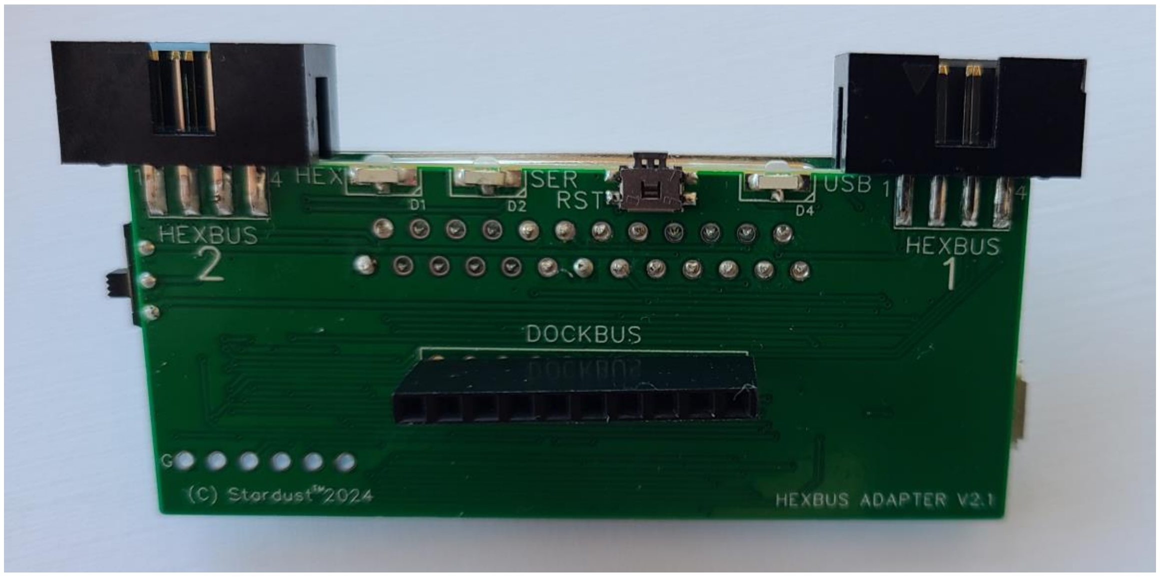


Figure 2: HEXBUS adapter

Firmware Operation

The firmware can operate in three separate states: HEXIF state, CLI state and Sleep state. The HEXIF state is the default state of operation that is used to provide Hexbus protocol services to the TI-74 and CC-40. While in the HEXIF state, the Hexbus interface is continuously monitored for Hexbus Commands, or ENTER/RETURN keys (CR) on the console port to potentially start entry to the CLI state. The Hexbus Protocol Commands generally start with the BAV line going low making the start of the command easy to detect. If 4 ENTERs in sequence are received, then the CLI state will be entered.

Entering four CRs in sequence will move the HEXBUS adapter to the CLI state from the HEXIF state. This can be done from any terminal program eg CoolTerm. Once in the CLI state a 'C' and '$' prompt will be displayed, at this point a number of commands can be used to configure the HEXBUS adapter and perform some management functions. See the Command Line Interface (CLI) section for a description of the commands that can be used. If no activity is detected on the serial port for 10s while in the CLI state, then the HEXIF State will be re-entered. Typing '`bye`' on the terminal will enter the HEXIF State immediately.

The Sleep state is entered when no activity has been detected on the interfaces for a set length of time. On the HEXBUS adapter the inactivity time before sleep can be set via the CLI. There is also an option to turn off the sleep mode if required. As described above in the hardware section in the Sleep state the processor, serial interface and Hexbus interface are all put into a low power mode to conserve power. The Sleep state is exited either when a character is received on the serial interface, or the USB interface receives a character or the Hexbus BAV line goes low, or the unit is power cycled/reset.

Annex A: Firmware Updates

The HEXBUS adapter employs a bootloader to permit field upgrades without the need to use a device programmer - once the bootloader has been installed. The HEXBUS adapter can be updated using the optional BPD+. Suitable *.bin* files for the bootloader can be found on the support page for the HEXBUS adapter.

Installing the HEXBUS adapter Bootloader

The bootloader is based on the zevero/avrboot (github: https://github.com/zevero/avr_boot) software it uses a canned Hexbus protocol based set of commands to control the BPD+ and request a file called `HEXBUSFW.BIN` from the BPD+ SD Card. Installing the bootloader will require an AVR programmer such as the Microchip PIC 4 or Atmel-ICE and Microchip visual studio to load the .bin file. The bootsize fuse should be set to 0x0B. Once programmed the bootloader will search for the default file `HEXBUSFW.BIN` on the BPD+ SD card. If the file is found this will be programmed into the flash memory and the HEXBUS adapter should be operational. If the file is not found, then the bootloader will continue to loop until an SD card with the file is found. The update file `HEXBUSFW.BIN` should be in the root directory of the SD card. The `HEXBUSFW.BIN` file should be a multiple of 512 bytes for proper operation.

The shorting plug should be removed when programming the bootloader, see the hardware description above for the location of the pin.

Updating HEXBUS adapter Firmware using the BPD+ SD card

1. Copy the new firmware file to the root of the BPD+ SD card.
2. Start the CoolTerm program and access the CLI (See “Command Line Interface”)
3. Enter command: `set update on<Enter>`
4. Reboot the HEXBUS adapter: `REBOOT<Enter>`
   The boot process should display the message “Normal Update” followed by “Processing File” on the CoolTerm screen.
5. Verify the new firmware version: `INFO<Enter>`

The shorting plug should be left in during this operation. Once the BIN file has been loaded into memory the HEXBUS adapter should reboot with the new firmware. The green and red LEDs will flash to provide a visual indication of the update progress.

Example using reboot command displayed on the USB port: (The version shown might be different)

```
$ set update on
OK
$ reboot
Hexbus BOOT V1.0
Normal Update using: HEXBUSFW.BIN
****
Processing File
****************************************************************
********************************
Complete - Restarting
Update ran
```

Recovering a 'bricked' HEXBUS adapter.

If after an update or due to other reasons the HEXBUS adapter stops responding to HEXIF or CLI commands, then it is possible to perform a hard recovery provided the bootloader has not been corrupted. If the bootloader has been corrupted, then it will be necessary to start the bootloader installation operation assuming a blank MCU as described earlier.

To perform a hard recovery first power down the HEXBUS adapter and remove the shorting header from the programming pins. Then make sure the BPD+ is plugged into the serial port and the SD card has a copy of a known good firmware image and titled `HEXBUSFW.BIN` – this is the default image, at the root level of the directories. The unit can then be powered up to reprogram the firmware. If the USB port is connected the text below should be visible, if it is not displayed on power up then it is very likely the bootloader is corrupt. A successful read of the SD card should result in a recovered HEXBUS adapter.

```
Hexbus BOOT V1.0
Recovery mode using: HEXBUSFW.BIN
****
Processing File
****************************************************************
********************************
Complete - Restarting
Update ran
```

Annex B: AVR port usage

The following tables lists the ports used by the HEXBUS adapter the information is provided to allow the reuse of the adapter in other applications.

HEXBUS adapter port usage

```
* PA0 = Unused
* PA1 = Unused
* PA2 = HSK - TRI
* PA3 = BAV - TRI
* PA4 = D0  - TRI
* PA5 = D1  - TRI
* PA6 = D2  - TRI
* PA7 = D3  - TRI
* DDRA  = 00000000 = 0x00
* PORTA = 00000000 = 0x00

* PC0 = UART_TX1 - OUT
* PC1 = UART_RX1 - IN
* PC2 = RS232_EN - OUT
* PC3 = unused
* PC4 = unused
* PC5 = unused
* PC6 = unused
* PC7 = Unused
* DDRC  = 00000101 = 0x05
* PORTC = 00000101 = 0x05

* PD0 = unused
* PD1 = LED2 - OUT
* PD2 = LED1 - OUT
* PD3 = RS232_SHDN - OUT
* PD4 = UART_TX0 - OUT
* PD5 = UART_RX0 - IN
* PD6 = SEL2 - IN
* PD7 = VBATT - IN
* DDRD  =  00011110 = 0x1E
* PORTD =  00011110 = 0x1E

* PF0 = unused
* PF1 = unused
* PF2 = DTR - OUT
* PF3 = RTS - OUT
* PF4 = DSR - IN
* PF5 = CTS - IN
* PF6 = RESET- - IN
* PF7 = UDI    - IN
* DDRF  =  00001100 = 0x0C
* PORTF =  00001100 = 0x0C
```

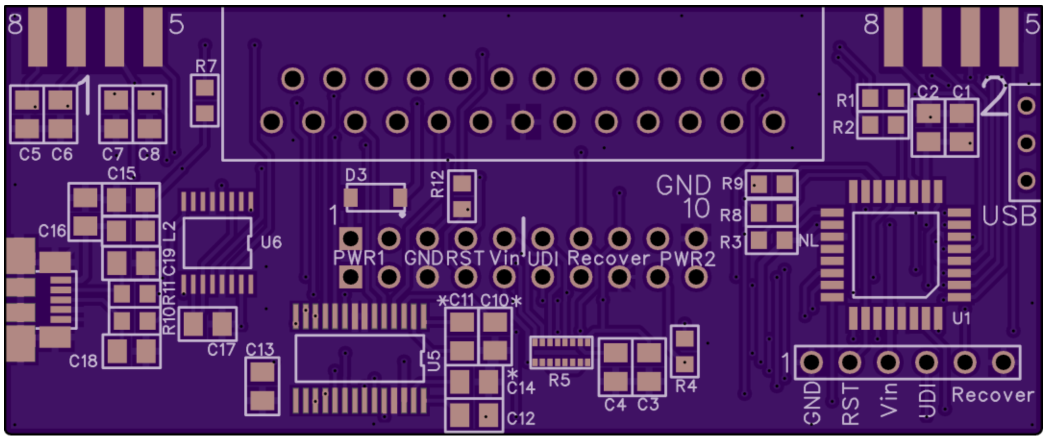
Annex C: HEXBUS adapter Part List

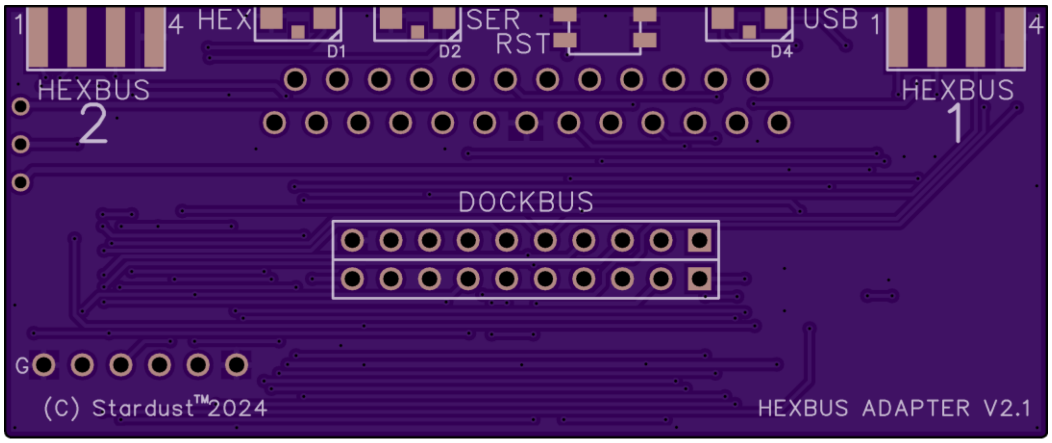
Manufacturer Part Number	Digi-Key Part Number	Ref	Qty	Description
CL21B104KBCNFNC	1276-2444-1-ND	C1, C12,C13,C17	4	CAP CER 0.1UF 50V X7R 0805
CL21B474KOFNNNG	1276-6483-1-ND	C10, C11,C14	3	CAP CER 0.47UF 16V X7R 0805
CL21B475KOFNFNE	1276-2970-1-ND	C15	6	CAP CER 4.7UF 16V X7R 0805
C0805X103K1RAC3316	399-C0805X103K1RAC3316CT-ND	C16	6	CAP CER 0805 10NF 100V X7R
C0805C470K5RAC7800	399-C0805C470K5RAC7800CT-ND	C18,C19	2	CAP CER 47PF 50V X7R 0805
CL21B105KOFNNNG	1276-6471-1-ND	C2	1	CAP CER 1UF 16V X7R 0805
CC0805KRX7R9BB181	311-4325-1-ND	C3,C4,C5,C6,C7,C8	6	CAP CER 180PF 50V X7R 0805
APA3010EC-GX	754-1579-1-ND	D1	1	LED RED CLEAR SMD R/A
APA3010SGC-GX	754-1586-1-ND	D2	1	LED GREEN CLEAR SMD R/A
1N4148W-13-F	1N4148W-13FDICT-ND	D3	1	DIODE GEN PURP 100V 300MA SOD123
APA3010YC-GX	754-1576-1-ND	D4	1	LED YELLOW CLEAR SMD R/A
Jameco 1003-25S	Jameco 5165	K1	1	CONN D-SUB PLUG 25POS VERT SLDR
BHR-08-VUA	2057-BHR-08-VUA-ND	K2,K3	1	CONN HEADER VERT 8POS 2.54MM

Manufacturer Part Number	Digi-Key Part Number	Ref	Qty	Description
0473460001	WM17141CT-ND	K3	1	CONN RCPT USB2.0 MICRO B SMD R/A
MH2029-221Y	MH2029-221YCT-ND	L1	1	FERRITE BEAD 220 OHM 0805 1LN
CRGCQ0603F180R	A129679CT-ND	R1,R2, R7	3	RES 180 OHM 1% 1/10W 0603
RMCF0603FT470R	RMCF0603FT470RCT-ND	R12	6	RES 470 OHM 1% 1/10W 0603
EXB-2HV822JV	Y1822CT-ND	R5	1	8.2k Ohm ±5% Isolated 8 Resistor Network/Array
RC0603JR-0727RL	311-27GRCT-ND	R8,R9,R10,R11	4	RES 27 OHM 5% 1/10W 0603
MHSS1105	679-1849-ND	SW1	1	SWITCH SLIDE SPDT 300MA 6V
EVQ-PUA02K	P10847SCT-ND	SW2	1	SWITCH TACTILE SPST-NO 0.05A 12V
AVR64DD32-I/PT	150-AVR64DD32-I/PT-ND	U1	1	64KB, 8KB RAM, 32P, 24MHZ, MVIO
TRS3243ECPW	TRS3243ECPW-ND	U5	1	IC TRANSCEIVER FULL 3/5 28TSSOP
FT230XS-R	768-1135-1-ND	U6	1	USB Bridge, USB to UART USB 2.0 UART Interface 16-SSOP

Annex D: PCB

HEXBUS adapter PCB





Annex E: Open-source license reproductions

Licenses for the various open-source software used in the project are reproduced below. Reference should be made to the relevant Github pages for up-to-date copies.

Bootloader for Backpack+ Drive

Github: https://github.com/zevero/avr_boot

License from Github: