Dyalog for Raspberry Pi
User Guide

Dyalog version 17.1

DYALOG

The tool of thought for software solutions

Dyalog for Raspberry Pi User Guide

Dyalog version 17.1
Document Revision: 20190808_171

Unless stated otherwise, all examples in this document assume that `⎕IO ⎕ML ← 1`

email: support@dyalog.com
https://www.dyalog.com

Contents

1 About This Document

This document is intended for anyone who wants to run Dyalog on a Raspberry Pi.

This document does not provide information on the Raspberry Pi or Dyalog – information and documentation for these products can be downloaded from http://www.raspberrypi.org/ and https://docs.dyalog.com respectively.

The information provided in this document supplements that given in the *Dyalog for UNIX Installation and Configuration Guide* and the *Dyalog for UNIX UI Guide* (available to download from https://docs.dyalog.com). In cases where the information given is different between the two documents, this document should be regarded as the definitive source for Dyalog on the Raspberry Pi.

1.1 Audience

It is assumed that the reader has a basic understanding of Linux on the Raspberry Pi. No prior knowledge of Dyalog is required.

For those who are new to Dyalog, some guidance on resources that can help with getting started is included in *Chapter 5*.

1.2 Conventions

Unless explicitly stated otherwise, all examples in Dyalog documentation assume that ⎕IO and ⎕ML are both 1.

Various icons are used in this document to emphasise specific material.

General note icons, and the type of material that they are used to emphasise, include:

Hints, tips, best practice and recommendations from Dyalog Ltd.

Information note highlighting material of particular significance or relevance.

Legacy information pertaining to behaviour in earlier releases of Dyalog or to functionality that still exists but has been superseded and is no longer recommended.

Warnings about actions that can impact the behaviour of Dyalog or have unforeseen consequences.

Although the Dyalog programming language is identical on all platforms, differences do exist in the way some functionality is implemented and in the tools and interfaces that are available. A full list of the platforms on which Dyalog version 17.1 is supported is available at https://www.dyalog.com/dyalog/current-platforms.htm. Within this document, differences in behaviour between operating systems are identified with the following icons (representing macOS, Linux, UNIX and Microsoft Windows respectively):

2 Introduction

Dyalog – the tool of thought for software solutions – is often used in research and business applications where it is important to integrate domain experts in the software development process. One of the reasons why Dyalog is selected is that applications written using APL often out-perform and outscale competing products written using compiled languages, even though compiled code should theoretically out-perform interpreted code. Dyalog, the only APL interpreter for the Raspberry Pi, allows the benefits of APL to be leveraged on the Raspberry Pi by bringing the concise, expressive power and performance of Dyalog to the compact, credit-card sized Raspberry Pi computer.

With the appropriate sensors and attachments, Dyalog can turn the Raspberry Pi into much more than a computer, for example, a temperature monitor, a proximity indicator or a robot. The progress of a robot being developed at Dyalog can be monitored through the following blogs:

- Dyalog developer blog: http://www.jasonrivers.co.uk/category/dyalog/
  (includes information on how to replicate the robot's hardware)
- Dyalog blog: https://www.dyalog.com/blog/category/robots

Some examples of code written for the Raspberry Pi using Dyalog are included in *Appendix C*.

The licence for Dyalog on the Raspberry Pi is for non-commercial use only. To license Dyalog on the Raspberry Pi for commercial use, please contact sales@dyalog.com.

3 Getting Started

This chapter covers the system requirements and installation information for installing 32-bit Unicode Dyalog on the Raspberry Pi, as well as the information needed to upgrade or remove an installation.

Only the 32-bit Unicode edition of Dyalog is available for the Raspberry Pi.

Dyalog on the Raspberry Pi can be accessed through:

- the RIDE (Dyalog's Remote Integrated Development Environment) – this provides a graphical user interface.
- a terminal window in the Pi desktop.
- terminal emulators on another Linux desktop (as described in the *Dyalog for UNIX Installation and Configuration Guide*).
- PuTTY on Microsoft Windows (it is not possible to use RDP, VNC or any X-window emulators from Microsoft Windows).

Dyalog recommends using the RIDE where possible (for more information on the RIDE, see the *RIDE User Guide*).

3.1 Pre-requisites

Dyalog can be installed and run on the following Raspberry Pi versions (all models):

- Zero
- 1
- 2
- 3
- 4

The RIDE client is not supported on the Zero or 1 models.

The Raspberry Pi must be running a Raspbian operating system; Dyalog version 17.1 requires Jessie or later. It must also have at least 150 MB available in the root file system for the installation of Dyalog – another 70 MB is required temporarily for the installation image.

3.2 Installing Dyalog on the Raspberry Pi

The following information is also available at https://packages.dyalog.com. In cases where the information differs, the webpage should be regarded as the definitive source.

To install Dyalog on the Raspberry Pi

1. Configure a repository using the following commands:

   ```
   $ wget -O - https://packages.dyalog.com/dyalog-apt-key.gpg.key | sudo apt-key add -
   $ CODENAME='lsb_release -sc'
   $ echo "deb https://packages.dyalog.com ${CODENAME} main" | sudo tee /etc/apt/sources.list.d/dyalog.list
   ```

2. Install Dyalog by entering the following commands:

   ```
   $ sudo apt-get update
   ```

   One of the following:

   - To install Dyalog with the RIDE GUI (recommended):
     ```
     $ sudo apt-get install dyalog-unicode
     ```
   - To install Dyalog without the RIDE GUI:
     ```
     $ sudo apt-get install dyalog-unicode-170
     ```

3. Read and accept the licence agreement. For non-commercial use, accepting the licence agreement entitles you to a free, unregistered version of Dyalog for Raspberry Pi. To license Dyalog on the Raspberry Pi for commercial use, please contact sales@dyalog.com.

Dyalog can now be run on the Raspberry Pi (see *Section 4.1*).

The following command can be used to see all the available packages:

```
$ sudo apt-cache search dyalog
```

3.2.1 Having Multiple Versions of Dyalog on the Raspberry Pi

Multiple versions of Dyalog can be installed on the Raspberry Pi. To install additional versions, change to the root user and enter the following command:

```
$ sudo apt-get install dyalog-unicode-<version-specific number>
```

where `<version-specific number>` is a three-digit number indicating the version. Supported values are:

- 141 (for Dyalog version 14.1 – RIDE not available)
- 150 (for Dyalog version 15.0 – RIDE not available)
- 160 (for Dyalog version 16.0 without the RIDE)
- 170 (for Dyalog version 17.0 without the RIDE)

3.3 Upgrading Dyalog on the Raspberry Pi

Dyalog updates are included with system updates and can be performed at the same time.

To upgrade to a later release or version of Dyalog

1. Determine the packages available to upgrade using the following command:
   ```
   $ sudo apt-get update
   ```
2. Upgrade the identified packages using the following command:
   ```
   $ sudo apt-get dist-upgrade
   ```

Dyalog is now upgraded to the latest version.

3.4 Uninstalling Dyalog from the Raspberry Pi

If necessary, Dyalog can be uninstalled from the Raspberry Pi.

To uninstall Dyalog from the Raspberry Pi

1. Uninstall Dyalog by entering the following commands:
   ```
   $ apt-get remove dyalog*
   $ apt-get purge dyalog*
   ```

Dyalog is now uninstalled.

Uninstalling Dyalog does not destroy the repository configured in *Section 3.2*.

4 Dyalog on the Raspberry Pi

This chapter covers the commands necessary to initialise and use 32-bit Unicode Dyalog on the Raspberry Pi.

4.1 Initialising Dyalog on the Raspberry Pi

To start Dyalog in a TTY window

1. Do one of the following:
   - In a terminal window, enter `dyalog`
   - From the start menu, select **Programming > Dyalog APL (TTY)**

   This display a new TTY window running a Dyalog Session. The session window is identical to the standard Linux non-GUI Dyalog session window as documented in the *Dyalog for UNIX UI Guide*.

To start Dyalog with a RIDE GUI

1. From the start menu, select **Programming > Dyalog APL**

   This display a new GUI window running a Dyalog Session. The session window shows the Dyalog development environment available through the RIDE, as documented in the *RIDE User Guide*.

If multiple versions of Dyalog are installed on the Raspberry Pi (see *Section 3.2.1*), then entering `dyalog` in a terminal window will start the latest version installed. Alternative versions can be started by entering **/opt/mdyalog/<version>/32/unicode/mapl**.

4.2 Setting the Dyalog Serial Number

If you have registered your copy of Dyalog or have a commercial licence then you will have been sent a Dyalog serial number; this serial number is individual to you and corresponds to the type of licence that you are entitled to use.

Dyalog Ltd recommends setting the serial number either by editing a file containing the serial number directly or by running a function in a Dyalog Session to update the file containing the serial number. The next time Dyalog is started after setting the serial number, the DYALOG_SERIAL environment variable is set to the contents of this file.

If the DYALOG_SERIAL environment variable already exists and has a non-empty value, then its value is not updated with the contents of the serial file.

In a multi-user environment it might be desirable to set the DYALOG_SERIAL environment variable in a system configuration file so that the serial number is held in a single location.

To set your Dyalog serial number by editing the serial number file directly

1. In your preferred text editor, edit the **$HOME/.dyalog/serial** text file so that it contains:

   ```
   serialnumber
   ```

   where *serialnumber* is your Dyalog serial number.

2. Save **$HOME/.dyalog/serial**.

   $HOME/.dyalog/serial is the default location for your serial number file but you can set the DYALOG_SERIALFILE environment variable to point to any other valid location.

To set your Dyalog serial number from within a Session

1. In a Dyalog Session, enter:

   ```
   ]SE.Dyalog.Serial serialnumber
   ```

   where *serialnumber* is your Dyalog serial number.

   The Dyalog serial number is saved in the serial number file **$HOME/.dyalog/serial**.

2. Exit and restart the Session.

Your serial number is displayed in the banner when you start a Session. To see your serial number at any time, enter:

```
      +2⎕NQ'.' 'GetEnvironment' 'DYALOG_SERIAL'
```

or

```
      ⎕SE.Dyalog.Serial ''
```

Using or entering a serial number other than the one issued to you is not permitted. Transferring the serial number to anyone else is not permitted. For the full licence terms and conditions, see https://www.dyalog.com/uploads/documents/terms_and_conditions.pdf.

4.3 Configuring Dyalog on the Raspberry Pi

Some configuration is a matter of choice (for example, the colour scheme). However, failure to configure the font set can result in Dyalog not displaying correctly in the TTY window.

By default, the Migration Level ⎕ML is 1 – this is reflected in this document and code samples that are available on the open source repository (https://github.com/APLPi).

The configuration options described in this chapter do not impact Dyalog when running using the RIDE GUI.

4.3.1 Configuring the Font Set

Dyalog uses Unicode APL glyphs, which do not display well in all fonts. Dyalog Ltd recommends using the APL385 Unicode font which is included with the installation.

To enable the Session to display the complete range of glyphs, this font must be used – it is installed as part of the Dyalog installation and must be set as the default terminal font the first time that Dyalog is run on the Raspberry Pi.

Setting the default terminal font to be the APL385 Unicode font impacts all terminal windows. However, as it is a monospace font the effect of doing this is minimal.

To configure the default terminal font to be the Dyalog font

1. In the Dyalog Session, select **Edit > Preferences**.

   The **Preferences** window is displayed.

2. In the **Style** tab of the **Preferences** window, click on the **Terminal Font** option. By default, this is *Monospace*.

   The **Font Selection** window is displayed.
3. From the **Family** list, select *APL385 Unicode*.
4. Click **OK** to confirm your selection and be returned to the **Preferences** window.
5. Click **OK** to be returned to the Dyalog Session.
6. In the Dyalog Session, press the **Control+L** keys simultaneously to refresh the window and restore the text that was previously visible.

4.3.2 Changing the Background Colour

This section is not specific to Dyalog on the Raspberry Pi and contains instructions that can be performed at any time. Changing the background colour of the default terminal window also changes the background colour of all terminal windows.

To change the background colour of the terminal window

1. In the Dyalog Session, select **Edit > Preferences**.

   The **Preferences** window is displayed.
2. In the **Style** tab of the **Preferences** window, click on the **Background** option. By default, this is *black*.

   The **Pick a Colour** dialog box is displayed.
3. Select the colour to use as the background colour by doing one of the following:
   - left-clicking the appropriate colour in the triangle on the left-hand side.
   - specifying the Red/Green/Blue values on the right-hand side.
4. Click **OK** to confirm your selection and be returned to the **Preferences** window.
5. Click **OK** to be returned to the Dyalog Session.
6. In the Dyalog Session, press the **Control+L** keys simultaneously to refresh the window and restore the text that was previously visible.

4.3.3 Changing the Colours for Syntax Colouring

When running in a TTY window, the Dyalog Session employs different colours for different syntactical purposes during input. For example, an entry between APL quotes is a different colour to an entry that cannot be evaluated.

Before changing the font colours

1. Open a terminal window
2. Copy **/opt/mdyalog/<version>/32/unicode/apltrans/xterm** to a suitable location, for example, **/home/<name>/dyalog/apltrans/**:

```
$ mkdir -p ~/dyalog/apltrans
$ cp /opt/mdyalog/14.0/32/unicode/apltrans/xterm ~/dyalog/apltrans/
```

3. Open the **~/.config/dyalog/config** file and add the following line:

```
export APLTRANS=/home/<name>/dyalog/apltrans:$APLTRANS
```

The **dyalog** directory is automatically created under the **XDG_CONFIG_HOME** directory (by default this is **~/.config**) the first time Dyalog is run.

4. Save the amended **config** file.

To change the font colours

1. Open the **~/dyalog/apltrans/xterm** file for editing.
2. Locate the appropriate entry and change it as required (see *Section 4.3.3.1*).
3. Save the amended **xterm** file.

4.3.3.1 Colour Values

The colour definitions in the **xterm** file use precisely-defined formats that follow the convention *<what is being defined> : <definition> <colour> m*. This can be appended with *+* and a comment if required.

For font colours, the definitions are formatted as: *27 91 51 56 59 53 59 <colour> 109*

To determine the <colour> value

1. Use http://en.wikipedia.org/wiki/File:Xterm_256color_chart.svg to identify the required colour's number (1-255).
2. Take each digit in the number individually and add 48 to it.

   The resulting set of one, two or three double-digit numbers is the <colour> value. For example, 162 has a <colour> value of 49 54 50 and 82 has a <colour> value of 56 50.

4.4 Exploring Dyalog on the Raspberry Pi

Details of the RIDE GUI (menu bars, configuration options, and so on) are available in the *RIDE User Guide*.

All the information that is needed to run Dyalog on the Raspberry Pi is available from the Dyalog Session.

Start a Dyalog Session (see *Chapter 4.1*) and display the `intro` workspace:

```
)LOAD intro
```

This displays some introductory information. This information can be displayed again at any time using the same command.

Four other commands can be used to display different categories of information:

`get_started`	useful commands to get you started (see *Section 4.4.1*)
`APLKeys`	APL keyboard layout (see *Section 4.4.2*)
`workspaces`	example workspaces to try (see *Section 4.4.3*)
`references`	useful websites (see *Section 4.4.4*)

These commands can only be run from within the `intro` workspace.

4.4.1 get_started

The `get_started` command displays a list of valid commands that can be used to manipulate workspaces. These include:

`)LOAD <myws>`	loads `<myws>` into the Dyalog session
`)SAVE <myws>`	saves active workspace as `<myws>`
`)SAVE`	saves active workspace
`)CLEAR`	clears active workspace
`)RESET`	resets state indicator
`)ED <item>`	opens the Dyalog Editor in the Session for the specified item/items
`)OFF`	exits the Dyalog Session

4.4.2 APLKeys

The `APLKeys` command displays an image of the keyboard showing the keys used for Dyalog glyphs. This mapping applies to all Linux keyboard layouts.

Users who are familiar with the Dyalog user interface under Microsoft Windows should familiarise themselves with any differences between their usual keyboard layout and the Linux keyboard layout.

An image of Linux keyboard mappings is reproduced in *Appendix B*. This image is also available in the xterm file in the **/opt/mdyalog/<version>/32/unicode/aplkeys** directory.

To enter APL glyphs in a workspace, hold down the **Windows** key while pressing the appropriate character key. When there are two glyphs on a key, hold down the **Windows** key while pressing the appropriate character key to get the lower of the two and hold down the **Windows** key and the **Shift** key while pressing the appropriate character key to get the upper of the two.

4.4.3 workspaces

The `workspace` command displays some example workspaces that can be loaded using the `)LOAD <workspace>` command. These include:

`sudoku`	sudoku solver for 2-D and 3-D puzzles
`life`	animated Conway's Game of Life
`echo`	TCP echo test (runs on port 7000)

4.4.4 references

The `references` command displays a list of websites that can assist with understanding and learning Dyalog. Further useful references are located in *Appendix A*.

4.5 Logging the Session

By default, a session logfile called **default.dlf** is created in the **~/.dyalog** directory. This binary file's default name and location can be overridden.

To override the session logfile name and location

1. Open a terminal window.
2. Open the **$HOME/.dyalog/dyalog.config** file and add the following line:
   ```
   export LOG_FILE=/<path>/<logfilename>.dlf
   ```
3. Save the amended **dyalog.config** file.

5 Starting with Dyalog

If you've not used Dyalog before then welcome! Dyalog is a concise and powerful language that has built a following of passionate advocates and it's always good to welcome someone new into the community.

Dyalog needs to be understood for its elegance and simplicity to be appreciated – in this way it is like every other coding language. Dyalog Ltd recommends the following resources, used together, to start your APL journey:

- Try Dyalog online: https://tryapl.org/

  TryAPL is a free application in which many of the functions and operators that form the core of Dyalog can be tested.

  

  When accessing the TryAPL application using Midori, the information in the left-hand side of each tab needs to be highlighted to be displayed. This rendering issue is due to the browser not fully supporting CSS/Javascript.

- "Mastering Dyalog APL": https://www.dyalog.com/mastering-dyalog-apl.htm

  Bernard Legrand's "Mastering Dyalog APL" is a complete guide to Dyalog. Its clear explanations and tutorials help the reader to advance from novice to specialist. Sample exercises are provided throughout – many of these can be performed in the TryAPL user interface. Alternatively, try them using the Dyalog Session on your Raspberry Pi. "Mastering Dyalog APL" is available as a free pdf download.

- "Learn APL on the $5 Raspberry Pi": https://leanpub.com/learnapl

  Romilly Cocking's "Learn APL on the $5 Raspberry Pi" provides a fast-paced introductory text and plenty of example code that together help the reader to discover programming in APL.

- The APL Orchard: https://chat.stackexchange.com/rooms/info/52405/the-apl-orchard?tab=conversations

  A series of lessons (on Stack Exchange) on individual aspects of APL.

Additional resources are detailed in *Appendix A*. Some examples of what can be achieved using Dyalog on the Raspberry Pi are included in *Appendix C*.

A Useful Resources

The following links provide information that might be useful when mastering Dyalog and enhancing the capabilities of your Dyalog-driven Raspberry Pi:

- Dyalog documentation: https://docs.dyalog.com
- Dyalog sample dfn workspaces: https://dfns.dyalog.com
- Try Dyalog online: https://tryapl.org and https://tio.run/#apl-dyalog
- Dyalog forums: https://forums.dyalog.com
- Dyalog blog of posts related to the Raspberry Pi: https://www.dyalog.com/blog/category/raspberry-pi/
- Dyalog blog of posts related to the DyaBot: https://www.dyalog.com/blog/category/robots/
- Dyalog's open source repository for code samples: https://github.com/APLPi
- "Mastering Dyalog APL": https://www.dyalog.com/mastering-dyalog-apl.htm
- "Learn APL on the $5 Raspberry Pi": https://leanpub.com/learnapl
- The APL Orchard (individual lessons on specific aspects of APL): https://chat.stackexchange.com/rooms/info/52405/the-apl-orchard?tab=conversations
- APL Wiki pages: http://aplwiki.com
- Editor keycodes and their common keystrokes are included in the appendices of the *Dyalog for UNIX UI Guide*. This information is also included in the **xterm** file located in the **/opt/mdyalog/<version>/32/unicode/aplkeys** directory.

Dyalog is also available for Linux (x86/x86_64), Microsoft Windows, macOS and IBM AIX. For more information, see https://www.dyalog.com/dyalog/current-platforms.htm.

B Keyboard Key Mappings

Figure B-1 shows the Linux keyboard key mappings used for Dyalog glyphs on a UK keyboard.

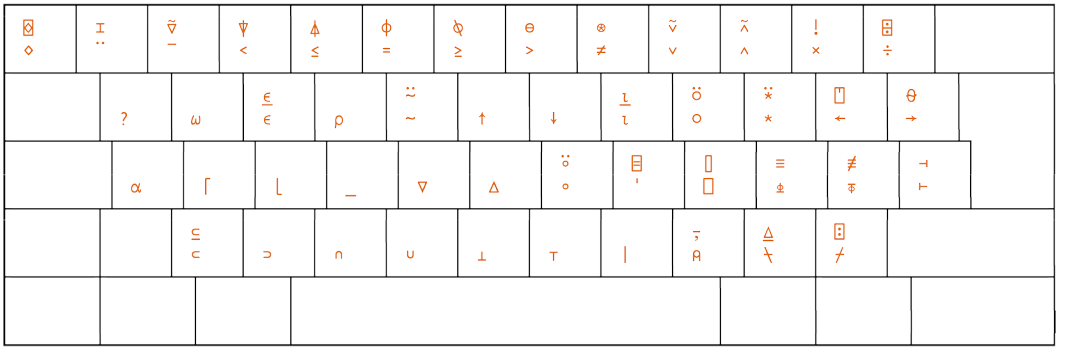

Figure B-1: *Linux keyboard key mappings used for Dyalog glyphs*

C Example Code

This appendix includes example code to illustrate some uses of Dyalog on the Raspberry Pi.

Sample code for Dyalog on the Raspberry Pi is available from the open source repository https://github.com/APLPi. This repository includes the code described in this appendix as well as further examples, and will be maintained and extended as the Dyalog for Raspberry Pi project develops.

C.1 Morse Code

This example code shows how the Raspberry Pi, when connected to a BBC micro:bit installed in a Kitronik Inventor's Kit, can use the included 5x5 LED panel to communicate messages in Morse code. The micro:bit is connected using USB, allowing the Raspberry Pi to connect to an interactive MicroPython REPL. MicroPython libraries include an API that gives access to all the I/O ports on the micro:bit, including direct support for the LED panel and the two buttons next to it.

The GitHub repository https://github.com/APLPi/microbit contains the following files:

- **Morse.dyalog** – APL code for the Morse code example.
- **microbit.dyalog** – general interface code between APL on the Raspberry Pi and the Python REPL.
- **MorseCode.txt** – a translation table which defines the dots and dashes required to render letters, digits and a few symbols as Morse code.

To run the example code

1. Clone the entire https://github.com/APLPi/microbit Github repository, or download the three files listed above into a single directory.
2. Ensure that the Python REPL is running on the micro:bit.

The easiest way to do this is to start the mu editor, and either flash a Python application to the micro:bit or use the REPL from mu – and then exit from mu to release the serial connection.

3. Start a Dyalog Session and load the example code using the `]Load` user command. For example (assuming the files are located in **/home/<username>/microbit**:

   ```
   ]Load /home/<username>/microbit/Morse
   ```

   The microbit class will be automatically loaded due to the `⍝∇:Require` statement at the start of the Morse file.

4. Initialise the Morse code table. For example:

   ```
   Morse.Init '/home/<username>/microbit/MorseCode.txt'
   ```

5. Use the `Display` function with the string to be output, that is,

   ```
   Morse.Display '<string>'
   ```

   For example, to output the Morse code characters for "SOS", enter `Morse.Display 'SOS'`. A video showing the expected result is available to view at https://www.youtube.com/watch?v=yfGsSLEifAs.

The Dyalog code in **morse.dyalog** is as follows:

```
:Namespace Morse
 ⍝ Display Morse code using the LEDs on a BBC micro:bit
 ⍝ Assumes the micro:bit is exposing the MicroPython REPL
 ⍝∇:require =microbit

 dit_ms←200÷1000 ⍝ length of "dit" = 200ms ("dah" will be 3x this)
 on←'09990:99999:99999:99999:09990'
 off←29⍴'00000:'

 upcase←1⌽(819⌶)          ⍝ Function to convert to Upper Case

 ∇ Init filename;t
 ⍝ Read MorseCode.txt file (actual name passed as argument
   t←⊃⎕NGET filename 1    ⍝ Read file into a vector of vectors
   Chars←(1⊃¨t),' '       ⍝ Chars are in column 1 (and add space)
   Codes←(1↓¨t)⍳¨⊂'.-'    ⍝ Codes are any .- chars after 1st char
   Codes,←⊂'  '           ⍝ Space is as a double long pause
   InitMB
 ∇
```

```
 ∇ InitMB⍝ Ensure existence of instance of micro:bit class
   :If 0=⎕NC '#.mb'
   #.mb←⎕NEW #.microbit ''
   :EndIf
 ∇

 ∇ Lights onoff;image;z
 ⍝ Use micro:bit Python API to manipulate LEDs
 ⍝ http://microbit-micropython.readthedocs.io/en/latest/microbit_
micropython_api.html
   image←(1+onoff)⊃off on ⍝ Select image
   z←#.mb.PyREPL 'display.show(Image(''',image,'''))
 ∇

 ∇ Display message;didah;duration;index;m;output
 ⍝ Output Morse code using LED connected to GPIO pin

   index←Chars⍳⎕upcase message ⍝ Look message up in Chars

   :If ∨/m←index>⍴Chars       ⍝ Any chars not found?
     ('UNSUPPORTED CHARS: ',m/message)⎕SIGNAL 11
   :Else ⋄ output←∈Codes[index],¨⊂' ⍝ one long string, w/","
between symbols
   :EndIf

   :For didah :In output
     duration←(1 3 3 7)['.-, '⍳didah]
     Lights didah∈'.-'      ⍝ Turn light on if dit or dah, keep
off for pause
     ⎕DL duration×dit_ms    ⍝ Leave lights on/off for appropriate
time
     Lights 0               ⍝ Turn off
     ⎕DL dit_ms             ⍝ Inter-didah pause
   EndFor
 ∇

 :EndNamespace
```

C.2 Average Dice Throw Values

This example code enables the Raspberry Pi to plot its output using SharpPlot (SharpPlot is a graphics package that is included with the Dyalog installation – it supports the

production of high quality charts in a variety of output formats using a simple set of Dyalog functions). No additional hardware is required to generate output from this example code.

To run the example code

1. Open a Dyalog Session window and enter the following code:

```
⍝ Load SharpPlot (needed for graphical output)
)LOAD sharpplot

⍝ Define function for cumulative average of throws
CumAvgRolls←{(+\?⍵⍴6)÷⍳⍵}
```

2. Optionally, test your function:

```
CumAvgRolls 10
```

   This should generate output similar to the following ten values:

```
1 3 3 3.25 3 2.666666667 3 3.25 3.555555556 3.5
```

3. Label the graph axes and set the line type:

```
⍝  Instantiate a SharpPlot object
sp←#.⎕NEW #.SharpPlot

⍝ Add a heading to the plot - (⎕UCS 10) is line feed
sp.SetHeading 'Cumulative Avgs of',(⎕UCS 10),'100 Dice
Throws'

⍝ Add a label to the Y-axis
sp.SetYCaption 'Average'

⍝ Add a label to the X-axis
sp.SetXCaption 'Throws'

⍝ Use only solid lines
sp.SetLineStyles #.LineStyle.Solid
```

4. Generate and plot your values using SharpPlot:

```
⍝ Roll 3 dice 100 times each and plot results
sp.DrawLineGraph ⊂CumAvgRolls¨3⍴100

⍝ Save SVG file
sp.SaveSvg '<path>/<filename>.svg'
```

   A file called **<filename>.svg** is saved in the specified **<path>**. Opening this file displays a graph similar to that shown in *Figure C-1*.

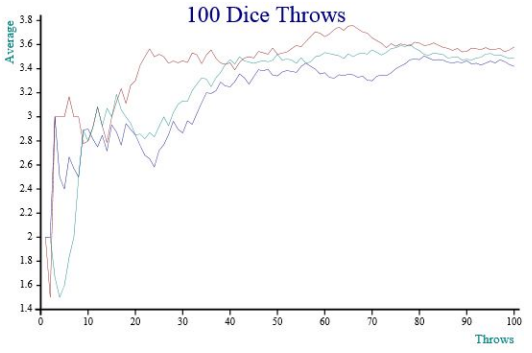

Figure C-1: *Graphical representation of cumulative averages*

Index