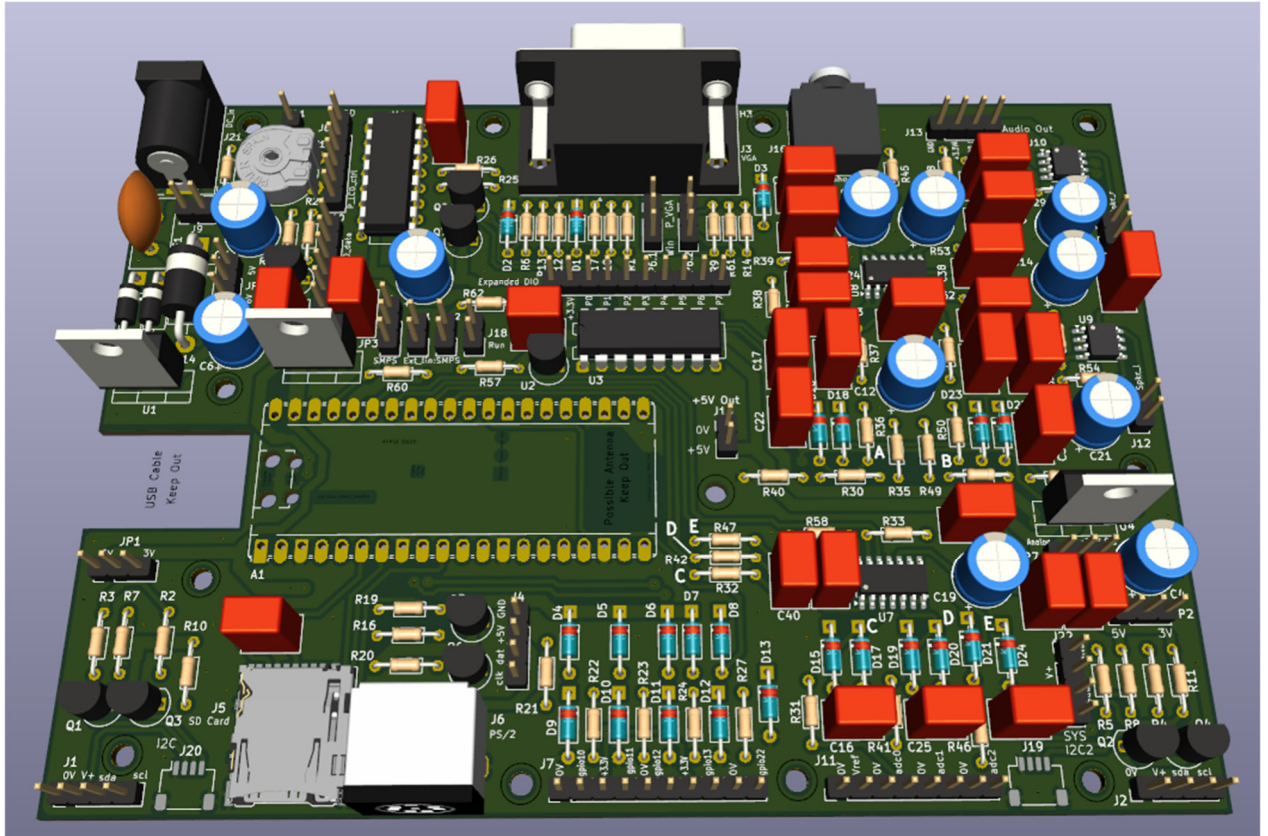


Building the Pi-PicoBuddy Computer V1.2



Acknowledgements

The Pi-PicoBuddy Computer is designed around the PicoMite BASIC firmware for the Raspberry Pi Pico. All due acknowledgements to Geoff Graham, Peter Mather and everyone else involved in the development of PicoMite Basic.

Acknowledgements are also due to Volhout, Javavi , Mixtel90 on the BackShed forum for their helpful suggestions regarding the implementation of the VGA output circuit.

Version updates

V1.2

- JP7 idents corrected
- Edge connector pin idents moved so they are not obscured by right-angle headers
- Audio filter and analogue input buffer filter bypass points updated
- Q1-9 pad spacing increased to simplify soldering

Contents

Introduction.....	1
System Overview.....	2
1 Schematic Overview	4
2 Power Management	5
3 PS/2 Keyboard	7
4 SD Card Memory	7
5 User Inputs / Outputs.....	8
6 VGA Monitor Interface	10
7 Peripheral Expansion	12
8 LCD Character Display (PLCD).....	14
9 Audio Output.....	15
10 Installing PicoMite Basic.....	17
Appendix A Power Supply Configurations	18
Appendix B Full Parts List.....	19
Appendix C Code Samples	23

Introduction

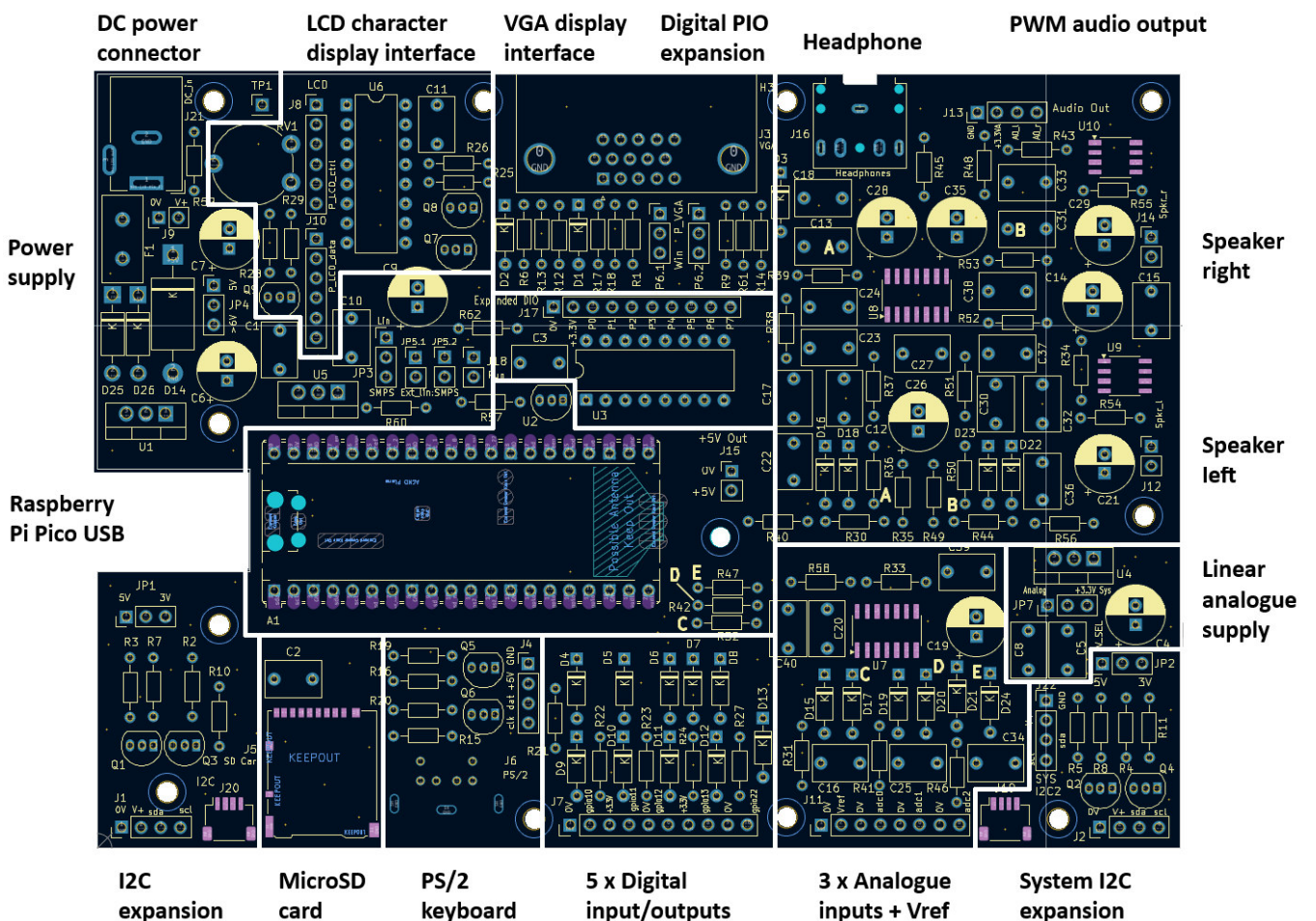
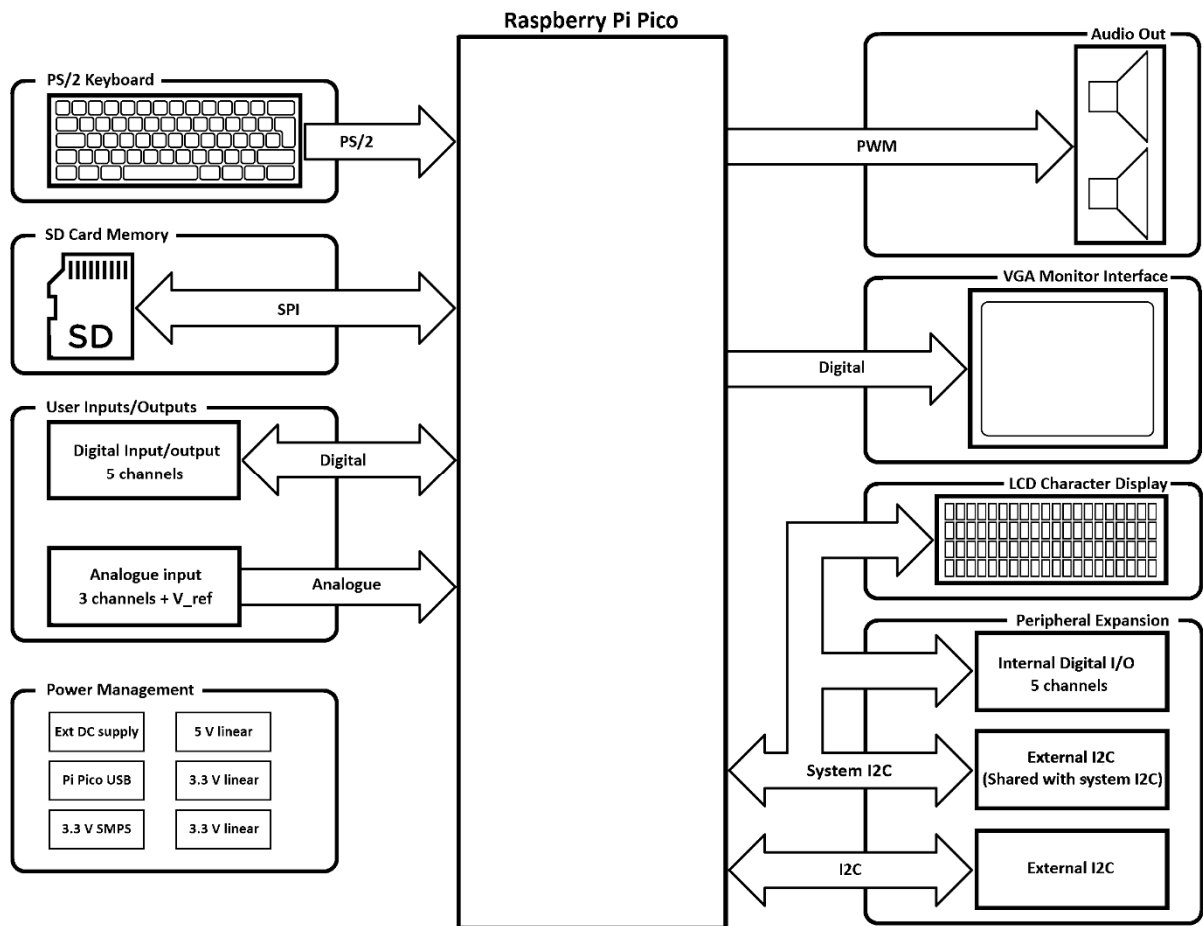
The Pi-PicoBuddy Computer project set out to build an old-style Boot-To-Basic computer in the same vein as the plethora of machines from the late 70's/early 80's. It also set out to be to be a fun, possibly even educational, project for my kids to build.

To achieve this the Pi-PicoBuddy Computer has been designed as a platform supporting the excellent PicoMite BASIC firmware for the Raspberry Pi Pico (<https://geoffg.net/picomite.html>).

The Pi-PicoBuddy Computer hardware project set out with the following design aims:

- **Low-cost to build.** The target was to be able to build a computer from a kit for the same price or less as the original ZX81 kit, £49.99. This meant using readily available components without relying on specialised or short life-expectancy parts. It also meant choosing as few *different* parts as possible so that instead of buying twenty different values of resistor, the builder would only need to buy multiples of the same component to benefit from price breaks where possible.
- **Simple to build.** The target was to make the build possible with only basic soldering tools and skills, and as uncomplicated as I could make it (e.g. using through-hole components wherever practical). The components had to be distinct for ease of assembly (e.g. resistors should avoid similar colour codes) and the orientation of polarised components should be strictly controlled on the PCB to aid visual inspection.
- **Multiple applications.** The computer should support as many options for expansion as could be managed within the limits of the price constraint. I kept in mind the BBC Micro, which was designed to maximise its “usefulness” in real-world and educational settings by having a huge number (for the time) of interfaces built-in.
- **Multiple power options.** The machine should be able to run from a variety of power sources. It also needed provide options for improved internal supplies e.g. to minimise noise and EMI and to remove supply “hiss” when generating audio.
- **Flexible implementation.** Whilst including the additional hardware options, the machine should not be reliant on them, so that it can be assembled with a minimal configuration (i.e. just a keyboard and VGA) as described in the PicoMite Basic firmware documentation.
- **Simple installation.** None of the additional hardware options should rely on alterations being made to the stock PicoMite Basic firmware. They should work either with the firmware out-of-the-box, or through the use of simple BASIC routines that can be included in the user's own programs.

System Overview



Raspberry Pi Pico running PicoMite VGA system firmware

- Raspberry Pi Pico (RP2040 or RP2035) running unmodified PicoMite VGA firmware. At the time of writing this is Ver 6.01.00.
- The Pi-PicoBuddy has also been tested with the USB-enabled version of PicoMite. When using this firmware the Pi-PicoBuddy must be powered by one of the external supply options through the DC Power Connector.

PS/2 keyboard

- 6-way Mini DIN connector, plus pin header for attaching a custom external PS/2 keyboard

VGA output

- See PicoMite VGA documentation for full details of the display modes available
- Option to switch between PicoMite VGA and Windows 16-style colour palettes.

SD Card memory

- Onboard micro-SD card socket

PWM audio

- See PicoMite VGA documentation for specifications
- Option to include an improved op-amp low-pass filter, with headphone output and direct 8 Ω speaker drive.

Configurable power supply

- The system can operate solely from the Raspberry Pi Pico's USB input or from an external supply. Configurable options for using linear regulators for sections of the power system supplies to minimise noise/EMI.

User I/O

- 5 x digital inputs/outputs
- 3 x analogue inputs with an externally available voltage reference.
- Additional digital I/O accessed via the system I2C bus to allow for internal functions e.g. detecting when the external DC power connector is used.

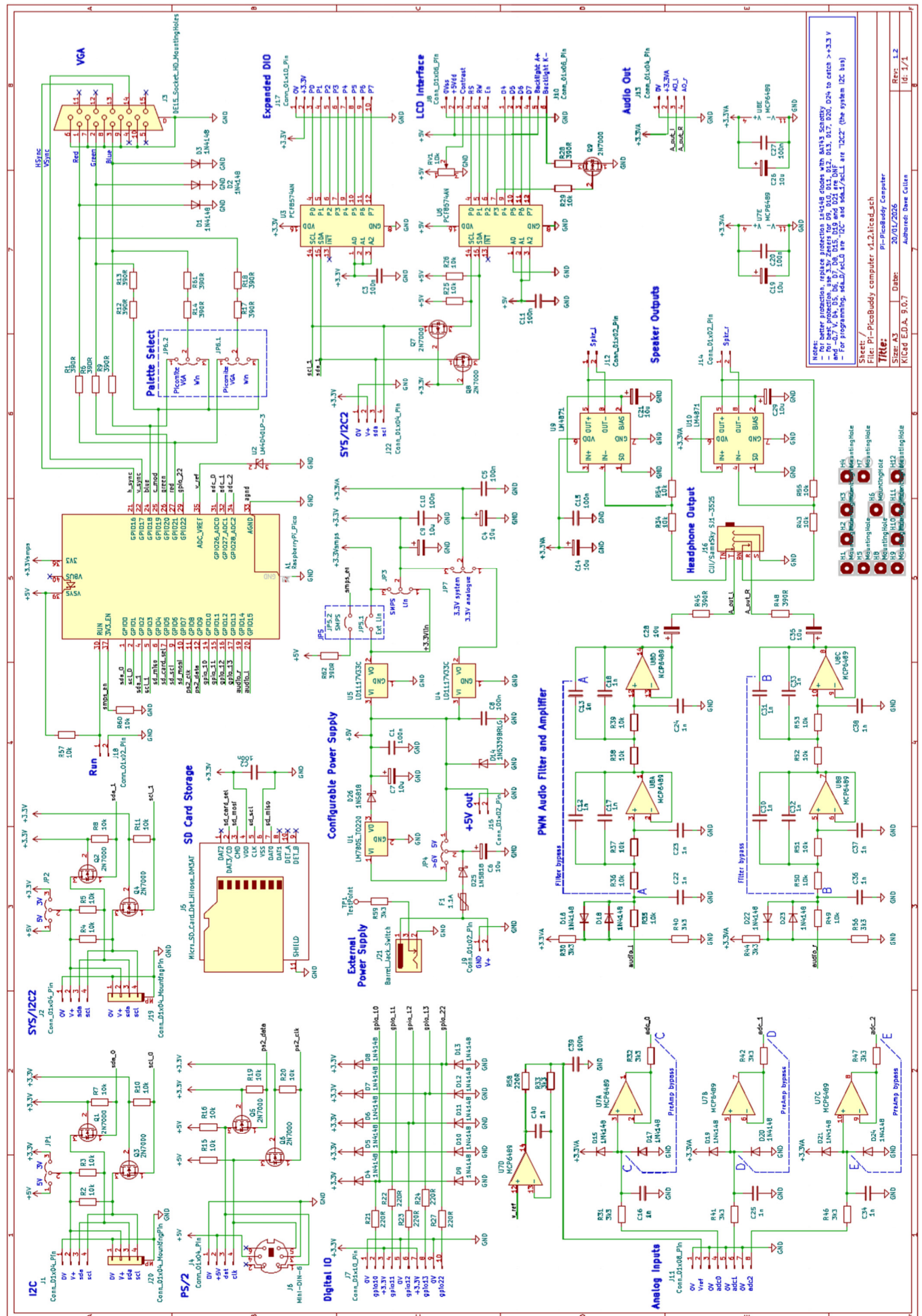
I2C expansion bus

- Two externally accessible I2C channels, each configurable to work with either 3.3V or 5V modules. One channel (I2C2) is shared with the system I2C.
- Connection by either general-purpose header pins or QWIIC/STEMMA QT-style headers.

LCD character display interface

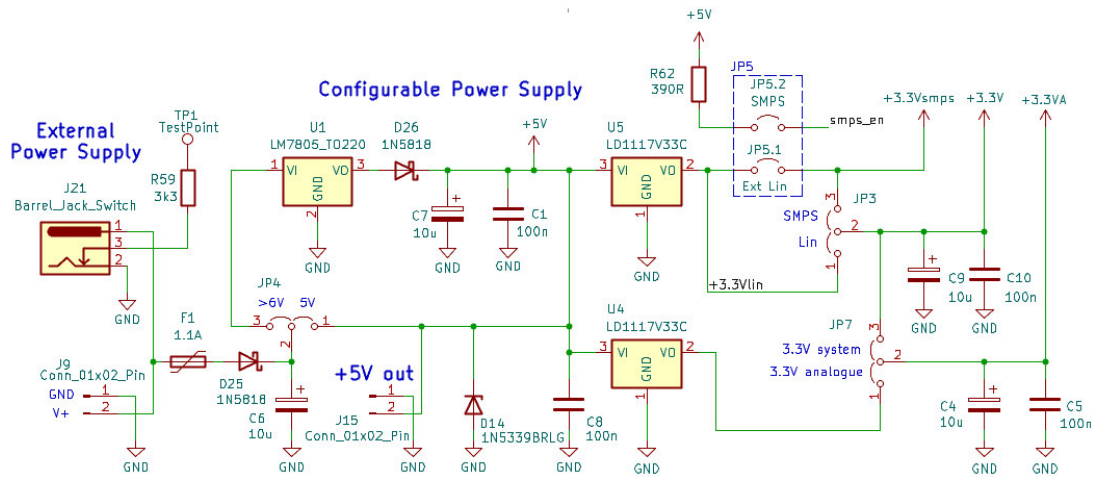
- On-board interface header for direct parallel connection to an HD44780 compatible LCD character display with support for up to 4-line devices.
- 4-bit parallel data connection with backlight ON/OFF control. Accessed through the system I2C bus.

1 Schematic Overview



2 Power Management

Description



Configuration

Supply configuration		Supply	Supply Rails			External Supply Rating		Pi Pico SMPS or Linear 3.3 V regulation (note 1)		3.3V System Rail		3.3V Analogue Rail	
From Pi Pico USB only		Input	5V	3.3V system	3.3V analogue	JP4		JP5.1	JP5.2	JP3		JP7	
						Ext 5V	Ext >5V	Lin 3.3V	Pi Pico SMPS enable	3.3V Lin	3.3V SMPS	Separate 3.3V linear rail	=3.3V system rail
						1-2	2-3			1-2	2-3	1-2	2-3
Basic	Pico USB supply only (no analogue 3.3V rail)	Pi Pico USB	Pi Pico 5V	Pico SMPS	n/a	n/a		X	Link	X	Link	n/a	n/a
	Pico USB supply + 3.3V SMPS analogue rail	Pi Pico USB	Pi Pico 5V	Pico SMPS		n/a		X	Link	X	Link	X	Link
Split	5V USB supply + linear 3.3V system rail only	Pi Pico USB	Pi Pico 5V	Linear U5	n/a	n/a		Link	X	Link	X	n/a	n/a
	5V USB supply + linear 3.3V analogue rail only	Pi Pico USB	Pi Pico 5V	Pico SMPS	Linear U4	n/a		X	Link	X	Link	Link	X
	5V USB supply + shared linear 3.3V rails	Pi Pico USB	Pi Pico 5V	Linear U5		n/a		Link	X	Link	X	X	Link
	5V USB supply + separated linear 3.3V rails	Pi Pico USB	Pi Pico 5V	Linear U5	Linear U4	n/a		Link	X	Link	X	Link	X
From External 5V regulated supply													
Basic	Ext 5V supply only (no analogue 3.3V)	Ext 5V	Ext 5V	Pico SMPS	n/a	Link	n/a	X	Link	X	Link	n/a	n/a
	Ext 5V supply + 3.3V SMPS analogue rail	Ext 5V	Ext 5V	Pico SMPS		Link	n/a	X	Link	X	Link	X	Link
Split	Ext 5V + linear 3.3V system rail only	Ext 5V	Ext 5V	Linear U5	n/a	Link	n/a	Link	X	Link	X	n/a	n/a
	Ext 5V supply + linear 3.3V analogue rail only	Ext 5V	Ext 5V	Pico SMPS	Linear U4	Link	n/a	X	Link	X	Link	Link	X
	Ext 5V supply + shared linear 3.3V rails	Ext 5V	Ext 5V	Linear U5		Link	n/a	Link	X	Link	X	X	Link
	Ext 5V supply + separated linear 3.3V rails	Ext 5V	Ext 5V	Linear U5	Linear U4	Link	n/a	Link	X	Link	X	Link	X
From External >5V supply													
Basic	Ext >5V supply only (no analogue 3.3V)	Ext >5V	Linear U1	Pico SMPS	n/a	n/a	Link	X	Link	X	Link	n/a	n/a
	Ext >5V supply + 3.3V SMPS analogue rail	Ext >5V	Linear U1	Pico SMPS		n/a	Link	X	Link	X	Link	X	Link
Split	Ext >5V + linear 3.3V system rail only	Ext >5V	Linear U1	Linear U5	n/a	n/a	Link	Link	X	Link	X	n/a	n/a
	Ext >5V supply + linear 3.3V analogue rail only	Ext >5V	Linear U1	Pico SMPS	Linear U4	n/a	Link	X	Link	X	Link	Link	X
	Ext >5V supply + shared linear 3.3V rails	Ext >5V	Linear U1	Linear U5		n/a	Link	Link	X	Link	X	X	Link
	Ext >5V supply + separated linear 3.3V rails	Ext >5V	Linear U1	Linear U5	Linear U4	n/a	Link	Link	X	Link	X	Link	X

Notes: 1 JP5.1 and JP5.2 **MUST NOT** be linked at the same time, otherwise the Pi Pico SMPS will conflict with U5's output
2 J9 can be used to input a supply from an external connector, or to tap off the supply from the barrel jack to a peripheral e.g. pass-through supply to a low-voltage LCD monitor. If not needed, it can be omitted.
3 Linear regulator U1 may be replaced by a "7805" style DC-DC converter module for better efficiency/lower heat dissipation.
4 R59 can be omitted if Barrel Jack switch detection is not needed

Parts needed for each power supply configuration

Supply configuration		Supply	Supply Rails			Components Needed				
		Input	5V	3.3V system	3.3V analogue	Connectors / Jumpers (note 2)	Capacitors	Diodes	Regulators (note 3)	Other
From Pi Pico USB only										
Basic	Pico USB supply only (no analogue 3.3V rail)	Pi Pico USB	Pi Pico 5V	Pico SMPS	n/a	JP3, JP5.2	C1, C7, C9, C10	D14	n/a	n/a
	Pico USB supply + 3.3V SMPS analogue rail	Pi Pico USB	Pi Pico 5V	Pico SMPS		JP3, JP5.2, JP7	C1, C4, C5, C7, C9, C10	D14	n/a	n/a
Split	5V USB supply + linear 3.3V system rail only	Pi Pico USB	Pi Pico 5V	Linear U5	n/a	JP3, JP5.2	C1, C7, C9, C10	D14	U5	n/a
	5V USB supply + linear 3.3V analogue rail only	Pi Pico USB	Pi Pico 5V	Pico SMPS	Linear U4	JP3, JP5.2, JP7	C1, C4, C5, C7, C8, C9, C10	D14	U4	n/a
	5V USB supply + shared linear 3.3V rails	Pi Pico USB	Pi Pico 5V	Linear U5		JP3, JP5.2, JP7	C1, C4, C5, C7, C9, C10	D14	U5	n/a
	5V USB supply + separated linear 3.3V rails	Pi Pico USB	Pi Pico 5V	Linear U5	Linear U4	JP3, JP5.2, JP7	C1, C4, C5, C7, C8, C9, C10	D14	U4, U5	n/a
From External 5V regulated supply										
Basic	Ext 5V supply only (no analogue 3.3V)	Ext 5V	Ext 5V	Pico SMPS	n/a	JP3, JP4, JP5.2, J9, J21	C1, C6, C7, C9, C10	D14, D25	n/a	F1, R59 (opt)
	Ext 5V supply + 3.3V SMPS analogue rail	Ext 5V	Ext 5V	Pico SMPS		JP3, JP4, JP5.2, JP7, J9, J21	C1, C4, C5, C6, C7, C9, C10	D14, D25	n/a	F1, R59 (opt)
Split	Ext 5V + linear 3.3V system rail only	Ext 5V	Ext 5V	Linear U5	n/a	JP3, JP4, JP5.2, J9, J21	C1, C6, C7, C9, C10	D14, D25	U5	F1, R59 (opt)
	Ext 5V supply + linear 3.3V analogue rail only	Ext 5V	Ext 5V	Pico SMPS	Linear U4	JP3, JP4, JP5.2, JP7, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25	U4	F1, R59 (opt)
	Ext 5V supply + shared linear 3.3V rails	Ext 5V	Ext 5V	Linear U5		JP3, JP4, JP5.2, JP7, J9, J21	C1, C4, C5, C6, C7, C9, C10	D14, D25	U5	F1, R59 (opt)
	Ext 5V supply + separated linear 3.3V rails	Ext 5V	Ext 5V	Linear U5	Linear U4	JP3, JP4, JP5.2, JP7, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25	U4, U5	F1, R59 (opt)
From External >5V supply										
Basic	Ext >5V supply only (no analogue 3.3V)	Ext >5V	Linear U1	Pico SMPS	n/a	JP3, JP4, JP5.2, J9, J21	C1, C6, C7, C9, C10	D14, D25, D26	U1	F1, R59 (opt)
	Ext >5V supply + 3.3V SMPS analogue rail	Ext >5V	Linear U1	Pico SMPS		JP3, JP4, JP5.2, JP7, J9, J21	C1, C4, C5, C6, C7, C9, C10	D14, D25, D26	U1	F1, R59 (opt)
Split	Ext >5V + linear 3.3V system rail only	Ext >5V	Linear U1	Linear U5	n/a	JP3, JP4, JP5.2, J9, J21	C1, C6, C7, C9, C10	D14, D25, D26	U5, U1	F1, R59 (opt)
	Ext >5V supply + linear 3.3V analogue rail only	Ext >5V	Linear U1	Pico SMPS	Linear U4	JP3, JP4, JP5.2, JP7, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25, D26	U4, U1	F1, R59 (opt)
	Ext >5V supply + shared linear 3.3V rails	Ext >5V	Linear U1	Linear U5		JP3, JP4, JP5.2, JP7, J9, J21	C1, C4, C5, C6, C7, C9, C10	D14, D25, D26	U5, U1	F1, R59 (opt)
	Ext >5V supply + separated linear 3.3V rails	Ext >5V	Linear U1	Linear U5	Linear U4	JP3, JP4, JP5.2, JP7, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25, D26	U4, U5, U1	F1, R59 (opt)

Notes: 1 JP5.1 and JP5.2 **MUST NOT** be linked at the same time, otherwise the Pi Pico SMPS will conflict with U5's output
2 J9 can be used to input a supply from an external connector, or to tap off the supply from the barrel jack to a peripheral e.g. pass-through supply to a low-voltage LCD monitor
3 Linear regulator U1 may be replaced by a "7805" style DC-DC converter module for better efficiency/lower heat dissipation.
4 R59 can be omitted if Barrel Jack switch detection is not needed

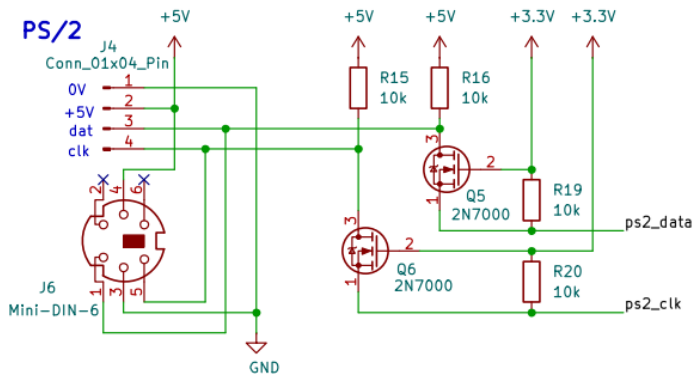
In the simplest configuration and lowest part count, the Pi-PicoBuddy Computer can be powered directly and solely from the Raspberry Pi Pico USB connector, with the +3.3V rails for the board circuitry connected to the +3.3V supply provided by the Pico's switched-mode power supply (SMPS). In this mode the Pi-PicoBuddy Computer typically draws around 130 mA, not including additional connected devices or modules.

Space is provided to replace the SMPS with a 3.3V linear voltage regulator. This lowers the system noise at the cost of lower efficiency and attention must be given to the amount of heat dissipated by the linear regulator.

There is also space for a separate 3.3V linear regulator to power the analogue sections of the circuit (analogue inputs, PWM audio output) for minimal noise on the audio circuits in particular.

Options to power the Pi-PicoBuddy Computer directly from an external +5V supply are included. There is space for a 5V "7805"-style regulator to be fitted so that >5V external supplies can be used e.g. a 9V supply. A drop-in "7805-style" switching regulator can also be used for improved efficiency and lower heat dissipation.

3 PS/2 Keyboard



Description

A 6-pin mini-DIN connector (J6) is fitted for connection to a PS/2 keyboard. Alternatively, a custom hardwired PS/2 protocol keyboard can be connected to the breakout connector (J4).

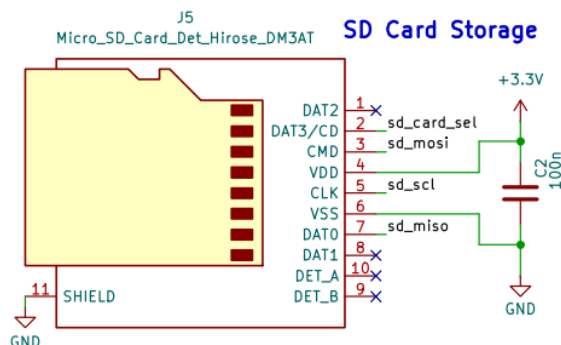
PicoMite Basic Configuration

Enter either one of these commands at the command prompt as appropriate for the keyboard attached. These commands force a reboot, but will be remembered by the system and so will not need to be re-entered.

To enable a UK keyboard: **OPTION KEYBOARD UK**

To enable a US keyboard: **OPTION KEYBOARD US**

4 SD Card Memory



Description:

The PCB accepts the Hirose DM3AT-SF-PEJM5(11) micro-SD connector. The pins are very close together, so check for shorts between adjacent pads. Check the physical compatibility before using an alternative model.

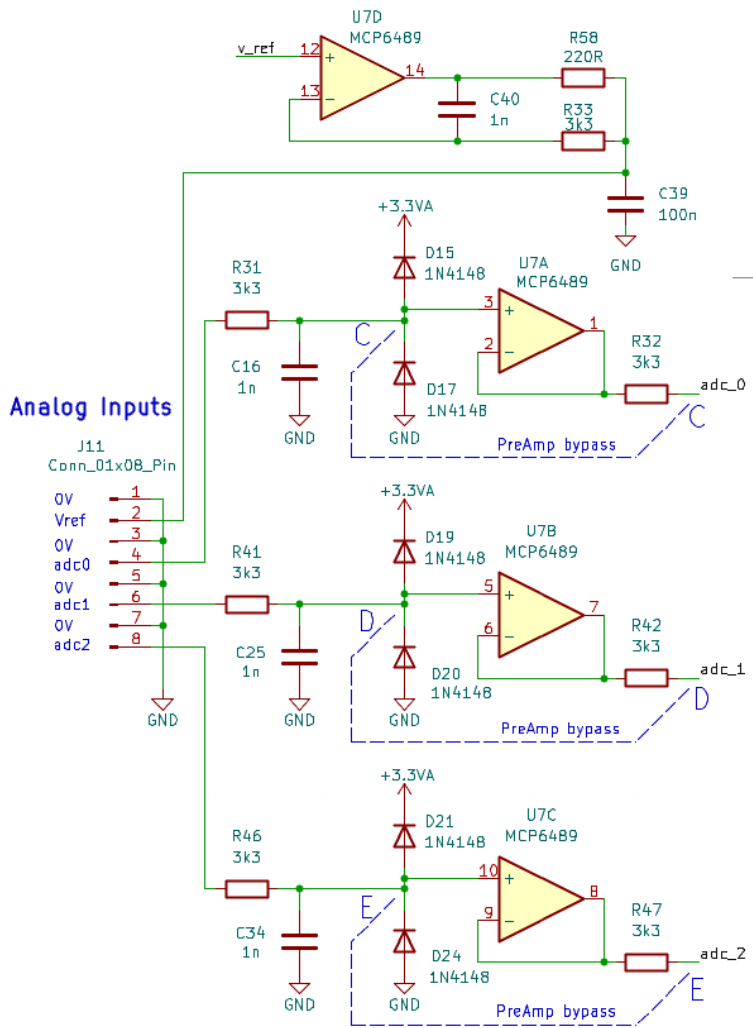
PicoMite Basic Configuration

The system SPI serial interface is not used by the Pi-PicoBuddy Computer, so the following command must be entered at the command line to enable the MicroSD card storage. This will force a reboot, after which the MicroSD card will be accessible as the “B:” drive.

OPTION SDCARD GP5, GP6, GP7, GP4

5 User Inputs / Outputs

Analogue Input



Description

Analogue-to-digital converter (ADC) inputs are present on connector (J11). Each input features a low-pass R-C filter (approximately 48kHz cutoff), diode voltage clamps for over- and reverse-voltage protection, and a buffer amplifier. Replacing the 3.3kΩ resistors with 10kΩ resistors will lower the filter cutoff frequency to approximately 16kHz, if preferred.

An MCP6489 is listed in the BOM. The MCP6489 is a low-cost quad op-amp with 10 MHz GBW, rail-to-rail operation, in a standard 14-pin SOIC package. Alternative quad op-amps may also be suitable, provided they can operate with input & output signals down to the 0V rail.

If the buffer amplifiers are not needed then (U7) can be omitted and bypassed by linking the C-C, D-D and E-E points indicated. If you prefer to use these pins for additional digital IO instead, omit U7, C16, C25, and C34, R32, R42, and R47, add the C-C, D-D and E-E links and replace R31, R41, and R46 with values that suit your circuit.

The Raspberry Pi Pico analogue reference voltage is buffered and sent out to the connector. This allows passive sensors (potentiometers for example) to be connected to the analogue input pins and provide full-scale range without needing an external excitation circuit. The V_{ref} buffered output has a 220Ω current-limiting resistor for current-limit protection. This must be considered when choosing the value of potentiometer, as the op-amp (U7D) output voltage increases to compensate for the load current through R58. Potentiometer values of 5kΩ or greater are recommended.

PicoMite Basic Configuration

No OPTION command is needed, but the following must be entered to enable each of the three ADC channels.

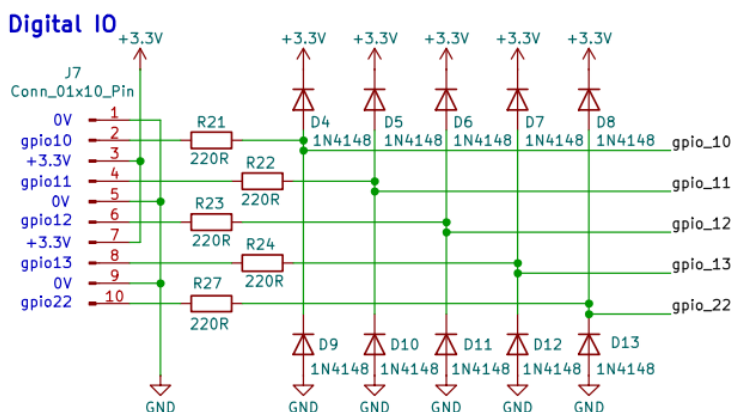
To enable ADC0, **SETPIN 31, AIN** (referenced in PicoMite Basic as **AIN_0**)

To enable ADC1 **SETPIN 32, AIN** (referenced in PicoMite Basic as **AIN_1**)

To enable ADC2 **SETPIN 34, AIN** (referenced in PicoMite Basic as **AIN_2**)

Refer to the PicoMite User Manual for details of reading the analogue-to-digital input pins.

Digital Input/Output



Description

General-purpose IO pins GPIO_10..13 and GPIO22 are available on connector J7. Each IO has a 220Ω current-limiting resistor and a pair of diode clamps fitted for over- and reverse-voltage protection.

The basic circuit uses general-purpose 1n4148 diodes for this, which typically clamp the input voltages to within 0.6 – 0.7V of the supply rail/ground. For tighter voltage control, at the cost of an additional stock part, Schottky diodes (e.g. BAT43) will limit this to within approximately 0.4V of the supply rail/ground.

Even tighter protection can be achieved by replacing the GND-connected diodes (D9..D13) with 3.3V Zener diodes. These are fitted, diodes D4..D8 are not needed and can be omitted.

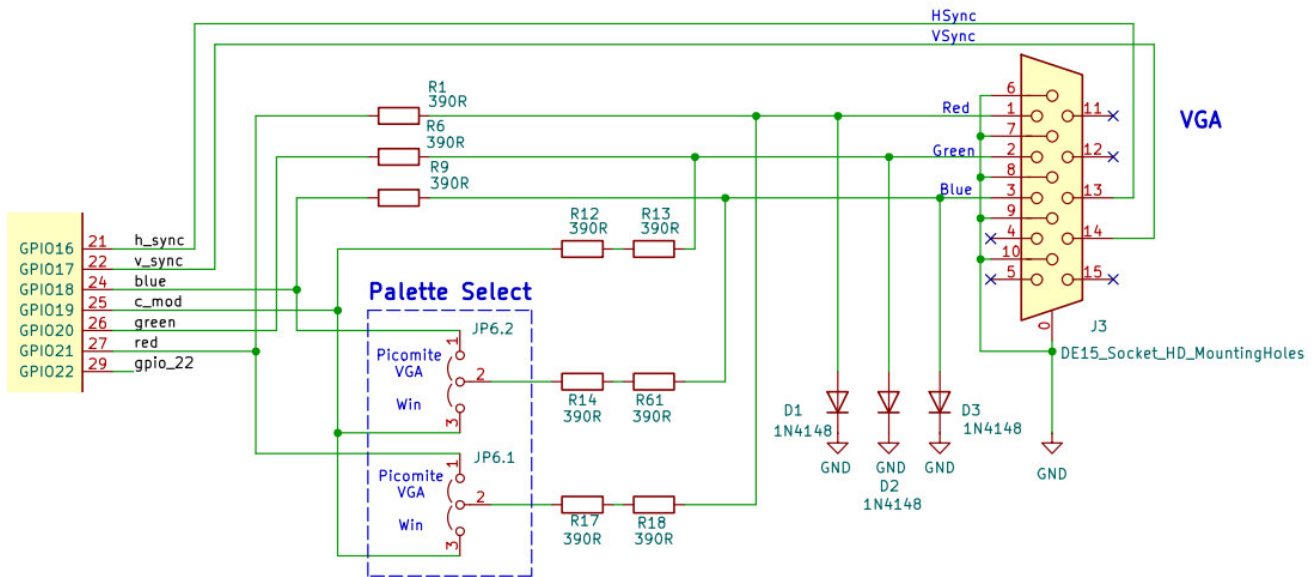
At the connector, the GPIO pins are interspersed with +3.3V and 0V power connections to allow simple wiring of switches (for example) using adjacent pins.

PicoMite Basic Configuration

No configuration is needed.

Refer to the PicoMite User Manual for details of reading and writing to the GPIO pins.

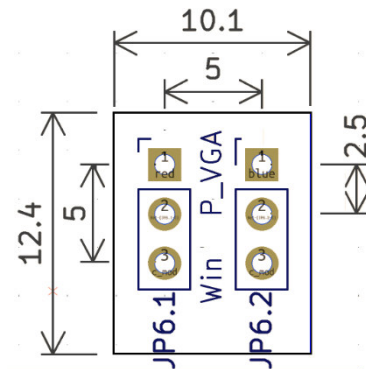
6 VGA Monitor Interface



Description

The 3.3V digital signals generated by the Raspberry Pi Pico are scaled to the 0 - 0.7V @75Ω signals expected by the VGA monitor by the resistor and diode networks shown.

The basic configuration produces the VGA colour palette described in the PicoMite manual. Here, a fourth signal “C_Mod” acts as a simple digital-analogue converter in conjunction with the GREEN signal to give four levels of green video output. An additional feature is to allow the “C_Mod” signal to adjust the RED and BLUE signals as well as the GREEN signal, which results in a palette closer to the Windows 16-colour scheme. A pair of 3-pin jumpers (JP6.1, JP6.2) are provided to swap between palettes. The jumper pins have been spaced so that a PCB pushbutton switch can be fitted instead, providing it fits the following footprint:

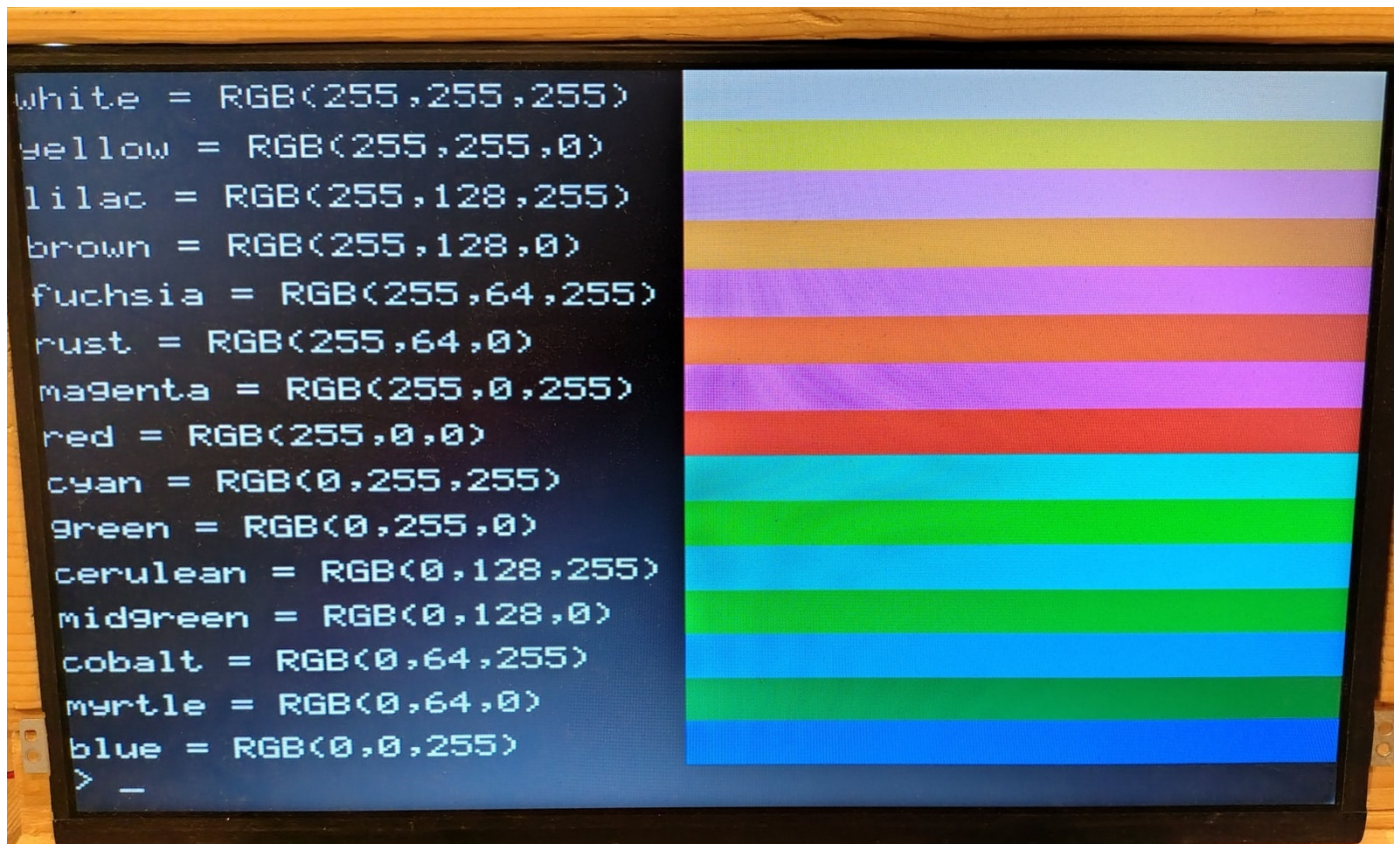


PicoMite Basic Configuration

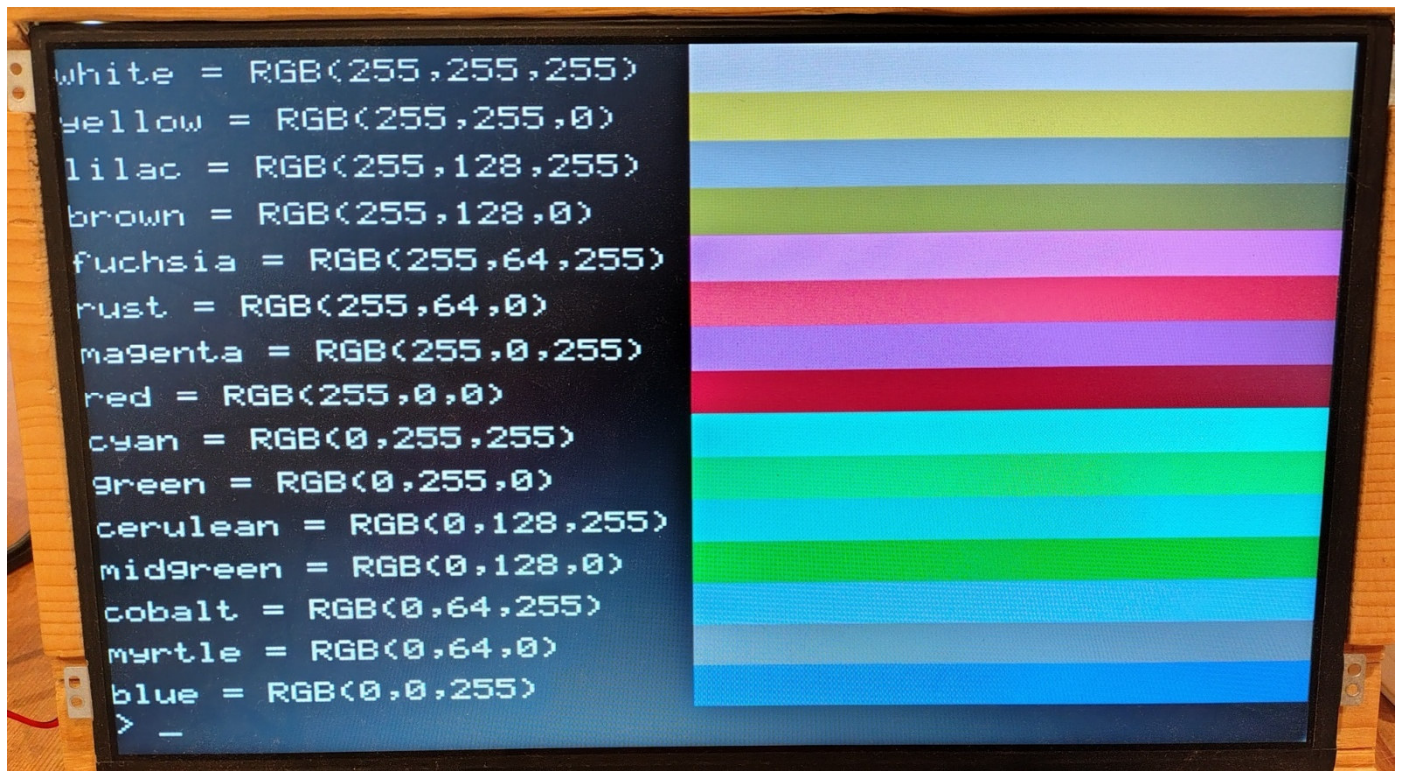
No configuration is needed in PicoMite Basic for the default VGA output. If an RP2350 is fitted then higher resolution modes are available, refer to the PicoMite manual for details of all VGA modes.

To change between standard PicoMite VGA colour palette and the “Win” palette, set the jumpers JP6.1 and JP6.2 as indicated on the PCB.

Standard PicoMite VGA 16-colour palette

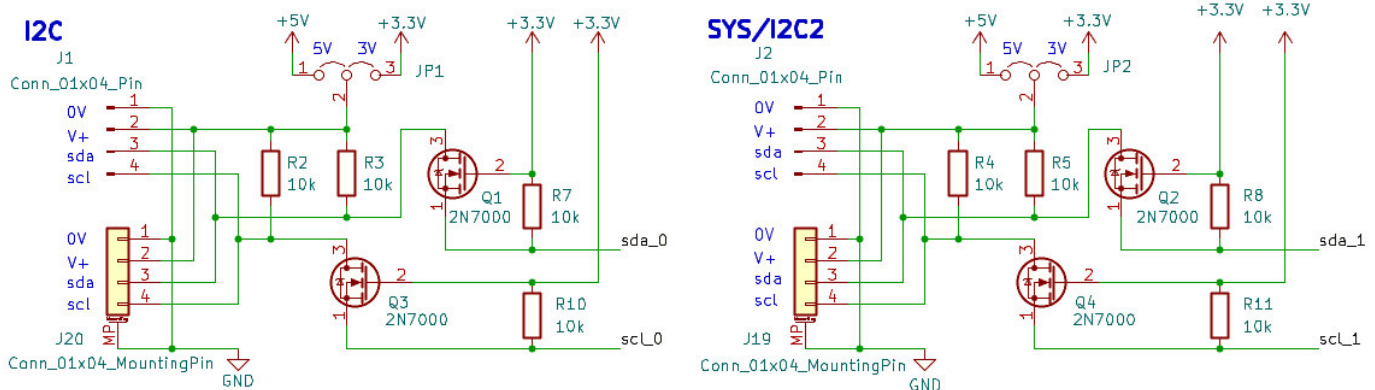


Modified Windows-style VGA 16-colour palette. Note that if names are used to refer to each colour in a program, the PicoMite names still apply (e.g. “lilac” refers to the darker grey).



7 Peripheral Expansion

I2C peripheral bus



Description

Two separate I2C buses are included for connecting external modules to the Pi-PicoBuddy. I2C is unused and fully available to the user. I2C2 is designated the system I2C bus and is shared with on-board I2C devices (LCD interface, PIO expansion).

Each external I2C bus connection can be configured for 5V or 3.3V devices using JP1 and JP2. Having two individual channels means that external modules with conflicting power requirements or addresses can be distributed between the two connections accordingly.

The system I2C bus is also available via a 4-pin header (J22) located bottom-right of the PCB. This is 3.3V-only and is intended for additional system modules that may form part of an expanded computer system e.g. real-time clocks, extra I/O interfaces.

PicoMite Basic Configuration

To enable the I2C bus, use the following commands:

To configure the hardware, use: **SET PIN GP0, GP1, I2C**

To open the channel: **I2C OPEN, 100, 1000**

To close the channel: **I2C CLOSE**

These commands do not force a reboot and can be incorporated into programs. Refer to the PicoMite User Manual for details of commands for the I2C bus.

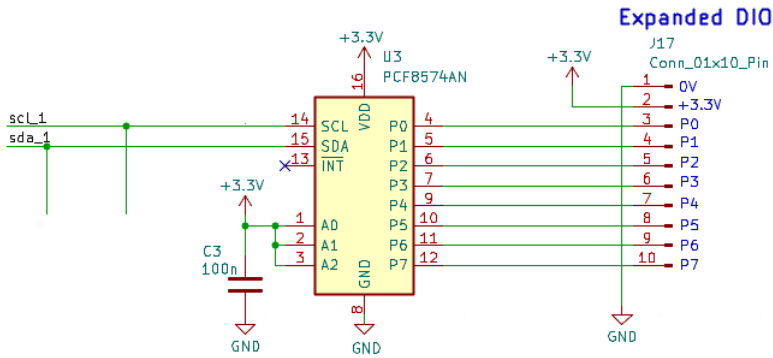
To enable the system/I2C2 bus, enter the following at the command line:

OPTION SYSTEM I2C GP2, GP3, SLOW

This will force a reboot, after which the 2nd I2C bus will be available to the user as “**I2C2**”. Once configured, **OPEN** and **CLOSE** commands are not needed to access I2C2.

Refer to the PicoMite User Manual for details of using devices with native support connected to the system I2C bus (e.g. Real-Time Clocks, temperature and humidity sensors etc.).

DIO Expander interface



Description

An additional 8-bit wide digital IO interface is provided onboard by an I2C to parallel expander IC (U3) connected to the I2C2 bus. This interface is mainly intended for in-system expansion (e.g. additional switches, LEDs) and does not feature any current-limit or over- under-voltage protection.

The PCF8574N was chosen as it can operate from a supply of 2.5 to 6V and is still available in a P-DIP through-hole package. Refer to the PCF8574N datasheet for details of its sink/source current capabilities etc.

The eight IO pins are available on connector J17, along with 3.3V and 0V supply lines.

If the expanded IO interface is not needed, U3, C3 and J17 can be omitted from the build.

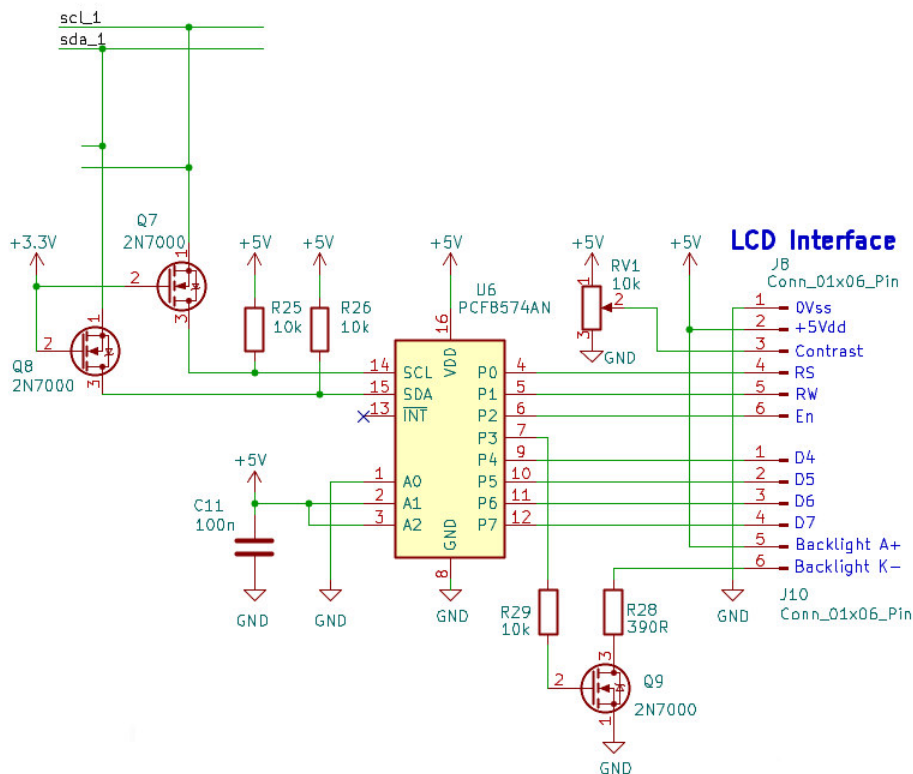
PicoMite Basic Configuration

I2C2 must be enabled in order to communicate with U3.

The U3 base address on the system I2C bus (I2C2) is located at 63_{10} , &H3F, 0b0011 1111.

Example code showing simple input and output operations is given in the Appendix.

8 LCD Character Display (PLCD)



Description

The on-board character LCD interface is intended to allow HD44780-compatible displays to be connected to the Pi-PicoBuddy. It has a separate addressable switch to turn the LCD backlight on/off (Q9). A potentiometer (RV1) sets the display viewing contrast.

An I2C-to-parallel interface IC (U6) directly drives the display pins, with Q7 and Q8 providing level translation to the 5V needed by the display. Commands must be sent to U6 to bit-bang the parallel data to the LCD. The LCD must be configured for 4-bit transfer mode. In 4-bit data transfer mode, bits D0 ..D3 are not used. All transfers are achieved using D4..D7 to transfer pairs of half-bytes.

The pinout of J8 is arranged so that it can be wired in the order shown to the lower six pins of the LCD connector. J10 carries the upper four data bits used for the half-byte interface mode, along with the switched backlight power. This pinout is also suitable for direct wiring to the upper six pins of most LCD connectors, providing the backlight pins are as shown. Check the LCD datasheet to confirm the pinout before fitting.

If an onboard LCD display is not needed, the components shown can be omitted from the build.

When using the LCD for the first time, the display will typically show two black and two empty lines (for a 4-line display). Adjust the contrast using RV1 until the characters can be seen clearly.

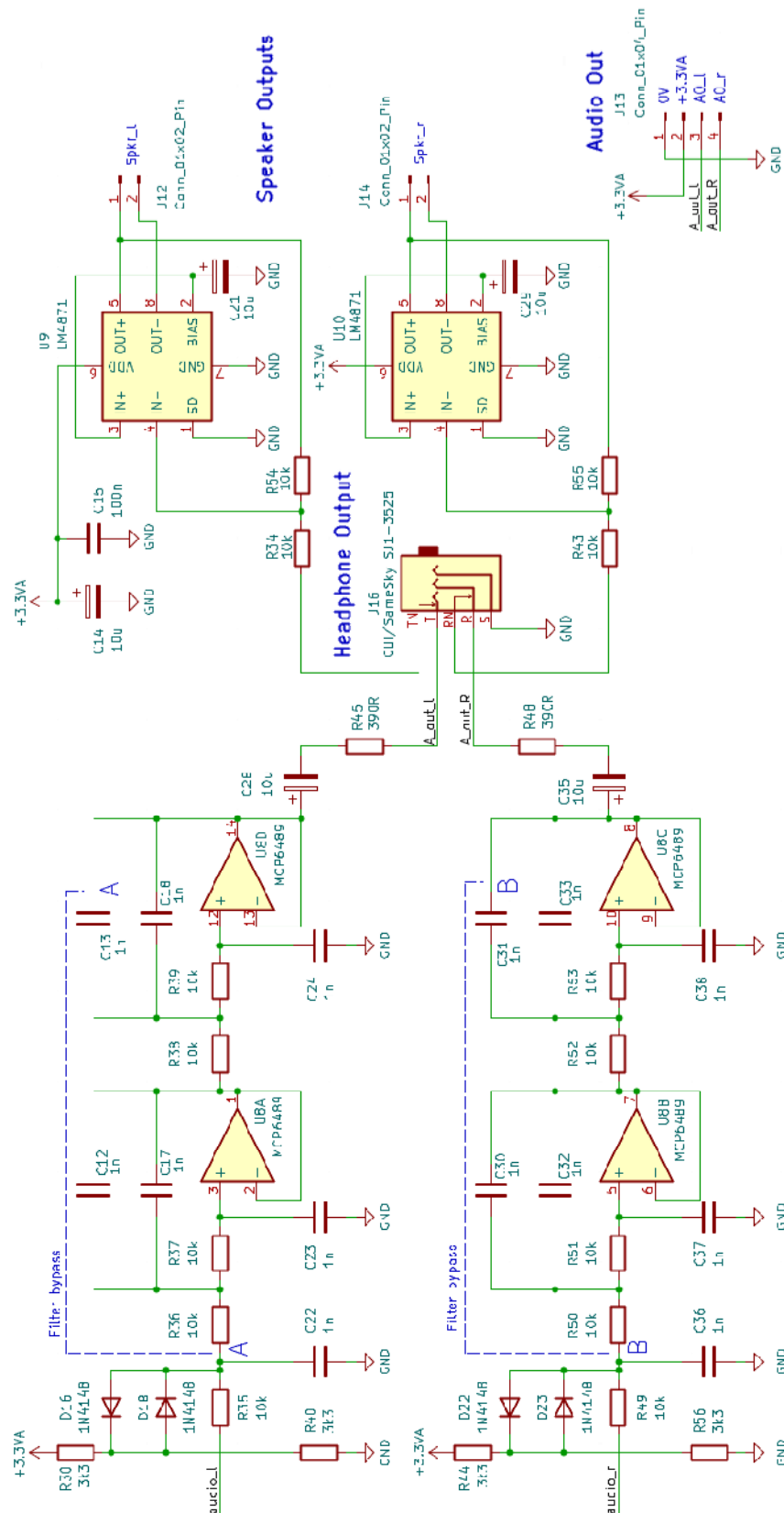
PicoMite Basic Configuration

I2C2 must be enabled in order to communicate with U6.

The U6 base address on the system I2C bus (I2C2) is located at 62₁₀, &H3E, 0b0011 1110.

Example code showing how to initialise the display, send strings, clear the display, and turn the backlight on/off is given in the Appendix.

9 Audio Output



Description

Audio output is generated using Pulse-Width Modulated (PWM) signals generated by the Raspberry Pi Pico. These varying pulse-width digital signals need to be filtered in order to produce decent quality sound, and then amplified so that they can drive headphones or loudspeakers.

The Pi-PicoBuddy uses a Sallen and Key op-amp low-pass filter circuit to remove the carrier noise from the PWM signal. The op-amp used in the circuit needs to have good gain-bandwidth properties for the low-pass

filter to operate correctly into the tens of kHz (a good rule of thumb is a gain-bandwidth product of 50x the cutoff frequency at least) as above a certain frequency the delay/phase-shift of the op-amp renders the filter unstable.

The 0 to 3.3V (typical) digital PWM signals from the Raspberry Pi Pico are first clipped to roughly $0.7 V_{pk-pk}$ and centred around the mid-supply voltage point by R30, R35, D16, D18 and R44, R49, R56, D22 and D23. Clipping helps to remove some of the power supply noise from the incoming signals (especially if the Raspberry Pi Pico's SMPS supply is being used), and limiting the signal swing to $0.7 V_{pk-pk}$ around the mid-supply point removes the need for rail-to-rail input op-amps if an alternative to the MCP6489 is available.

C22 and C36, in conjunction with R39 and R45 respectively, create a passive low-pass filter that suppresses signals that are high enough in frequency to cause the Sallen and Key filter to misbehave.

With the 10k Ω resistors and 1nF capacitors shown, the filter has a cutoff frequency of just over 11 kHz. Other bandwidths can be tuned by changing the R and C components from the standard values shown, with reference to the Sallen and Key filter design calculations (web-based calculators are available).

The outputs from the op-amp are sufficient to directly drive headphones connected to J16. The headphone output is also available, along with the +3.3V analogue supply, on a header connector (J13). This could be connected to an external amplifier, for example, but check the impedance matching is correct for your application.

If a jack plug is not inserted into the headphone connector, J16, it's internal switches route the signals to a pair of unity-gain class AB drive speaker amplifiers (U9, U10). LM4871M amplifiers have been chosen as they are readily available at low cost, give good quality sound, can operate from 3.3V supplies, and are able to directly drive 8 Ω loudspeakers. Class D amplifiers were tested during development for better power-efficiency, but they had the unfortunate side effect of reintroducing unacceptable levels of noise to the signal. The shutdown function of the LM4871M is not needed, as without any PWM input signal the balanced outputs remain in sync and do not drive current through the speakers. Sticking with the LM4871M devices is highly recommended for any build.

For best audio performance it is recommended that the power supply is configured to use one of the linear 3.3V analogue supply options, to avoid contamination of the sound by noise from the Raspberry Pi Pico SMPS.

PicoMite Basic Configuration

To enable PWM audio, enter the following command at the command prompt. This will force a reboot:

OPTION AUDIO GP14, GP15

This ties the audio generation functions to PWM Channel 7, with GP14 as the left channel and GP15 as the right channel.

Refer to the PicoMite User Manual for details of commands for generating sounds e.g.

PLAY WAV <wavfile>.WAV

PLAY VOLUME <nn>,<nn>

10 Installing PicoMite Basic

Description

The PicoMite firmware can be installed onto the Raspberry Pi Pico before or after assembly. Loading it before soldering the device to the circuit board is recommended as faulty devices can be identified early, or the device itself can be eliminated from enquiry if the finished Pi-PicoBuddy Computer board does not work following assembly.

There are several versions of the PicoMite firmware available, supporting either the RP2040 or RP2350 microcontrollers with VGA or HDMI, PS2 or USB support. The Pi-PicoBuddy Computer has been designed to use the basic PS/2 keyboard-only with VGA output. This version will work with the lowest-cost device, the RP2040 and has perhaps the least demand on the hardware. The Pi-PicoBuddy has also been tested successfully with the RP2350 and USB versions of the firmware.

At the time of writing, the firmware image to use is:

PicoMiteRP2040VGAV6.01.00.uf2

Instructions for programming the Raspberry Pi Pico 2040/2350 are given in the PicoMite User Manual

Configuration

A fully populated Pi-PicoBuddy Computer board needs the following commands entering at the command prompt to enable each subsystem (if installed). After each command is entered, the system will reboot and that option is committed to memory.

OPTION KEYBOARD UK (if using a UK keyboard)
OPTION SDCARD GP5, GP6, GP7, GP4
OPTION AUDIO GP14, GP15
OPTION SYSTEM I2C GP2, GP3, SLOW

To enable the general-purpose I2C bus, the following commands must be entered. They can be included in your Basic programs and will not require a reboot. Refer to the PicoMite User Manual for details of all the commands available for communicating with devices and modules connected to the I2C bus.

SET PIN GP0, GP1, I2C
I2C OPEN, 100, 1000
I2C CLOSE

To enable each of the analogue inputs the following commands must be entered. They can be included in your Basic programs and will not require a reboot.

To enable ADC0:	SETPIN 31, AIN	(referenced in PicoMite Basic as AIN_0)
To enable ADC1:	SETPIN 32, AIN	(referenced in PicoMite Basic as AIN_1)
To enable ADC2:	SETPIN 34, AIN	(referenced in PicoMite Basic as AIN_2)

Appendix A Power Supply Configurations

Supply configuration			Supply	Supply Rails		External Supply Rating			Pi Pico SMPS or Linear 3.3 V regulation (note 1)		3.3V System Rail			3.3V Analogue Rail			Components Needed					
From Pi Pico USB only	Basic	Input	5V	3.3V system	3.3V analogue	JP4		Ext 5V	Ext >5V	JP5.1	JP5.2	Pi Pico SMPS enable	3.3V Lin	JP3		JP7		Connectors / lumpsers (note 2)	Capacitors	Diodes	Regulators (note 3)	Other
						Ext 5V	Ext >5V							3.3V Lin	3.3V SMPS	Separate 3.3V linear rail	-3.3V system rail					
		Pi Pico USB	Pi Pico 5V	Pico SMPS	n/a	n/a	n/a	X	Link	Link	n/a	n/a	Link	Link	Link	Link	Link	JP3, JP5.2	C1, C7, C8, C10	D14	n/a	n/a
		Pi Pico USB	Pi Pico 5V	Pico SMPS	n/a	n/a	n/a	X	Link	Link	Link	Link	Link	Link	Link	Link	Link	JP3, JP5.2, JP7	C1, C4, C5, C7, C8, C10	D14	n/a	n/a
Split		Pi Pico USB	Pi Pico 5V	Linear U5	n/a	n/a	n/a	Link	X	Link	Link	X	Link	Link	Link	Link	Link	JP3, JP5.2	C1, C7, C8, C10	D14	U5	n/a
		Pi Pico USB	Pi Pico 5V	Pico SMPS	Linear U4	n/a	n/a	X	Link	Link	Link	Link	X	Link	Link	Link	Link	JP3, JP5.2, JP7	C1, C4, C5, C7, C8, C9, C10	D14	U4	n/a
		Pi Pico USB	Pi Pico 5V	Linear U5	Linear U5	n/a	n/a	Link	X	Link	Link	X	Link	Link	Link	Link	Link	JP3, JP5.2, JP7	C1, C4, C5, C7, C8, C9, C10	D14	U5	n/a
		Pi Pico USB	Pi Pico 5V	Linear U5	Linear U4	n/a	n/a	Link	X	Link	Link	X	Link	Link	Link	Link	Link	JP3, JP5.2, JP7	C1, C4, C5, C7, C8, C9, C10	D14	U4, U5	n/a
From External 5V regulated supply																						
Basic		Ext 5V	Ext 5V	Pico SMPS	n/a	n/a	Link	n/a	Link	X	Link	Link	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C6, C7, C9, C10	D14, D25	n/a	F1, R59 (opt)
		Ext 5V	Ext 5V	Pico SMPS	n/a	n/a	Link	n/a	Link	X	Link	Link	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25	n/a	F1, R59 (opt)
Split		Ext 5V	Ext 5V	Linear U5	n/a	n/a	Link	n/a	Link	X	Link	X	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C6, C7, C9, C10	D14, D25	U5	F1, R59 (opt)
		Ext 5V	Ext 5V	Pico SMPS	Linear U4	n/a	Link	n/a	Link	X	Link	Link	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25	U4	F1, R59 (opt)
		Ext 5V	Ext 5V	Linear U5	Linear U5	n/a	Link	n/a	Link	X	Link	X	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25	U5	F1, R59 (opt)
		Ext 5V	Ext 5V	Linear U5	Linear U4	n/a	Link	n/a	Link	X	Link	X	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25	U4, U5	F1, R59 (opt)
From External >5V supply																						
Basic		Ext >5V	Linear U1	Pico SMPS	n/a	n/a	n/a	Link	Link	X	Link	Link	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C6, C7, C9, C10	D14, D25, D26	U1	F1, R59 (opt)
		Ext >5V	Linear U1	Pico SMPS	n/a	n/a	n/a	Link	Link	X	Link	Link	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25, D26	U1	F1, R59 (opt)
Split		Ext >5V	Linear U1	Linear U5	n/a	n/a	n/a	Link	Link	Link	X	Link	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C6, C7, C9, C10	D14, D25, D26	U5, U1	F1, R59 (opt)
		Ext >5V	Linear U1	Pico SMPS	Linear U4	n/a	n/a	Link	Link	X	Link	Link	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25, D26	U4, U1	F1, R59 (opt)
		Ext >5V	Linear U1	Linear U5	Linear U5	n/a	n/a	Link	Link	Link	X	Link	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25, D26	U5, U1	F1, R59 (opt)
		Ext >5V	Linear U1	Linear U5	Linear U4	n/a	n/a	Link	Link	Link	X	Link	Link	Link	Link	Link	Link	JP3, JP4, JP5.2, J9, J21	C1, C4, C5, C6, C7, C8, C9, C10	D14, D25, D26	U4, U5, U1	F1, R59 (opt)

Notes:
1 JP5.1 and JP5.2 **MUST NOT** be linked at the same time, otherwise the Pi Pico SMPS will conflict with U5's output
2 J9 can be used to input a supply from an external connector, or to tap off the supply from the barrel jack to a peripheral e.g. pass-through supply to a low-voltage LCD monitor. If not needed, it can be omitted.
3 Linear regulator U1 may be replaced by a "7805" style DC-DC converter module for better efficiency/lower heat dissipation.
4 R59 can be omitted if Barrel Jack switch detection is not needed

Appendix B Full Parts List

MINIMAL VERSION: VGA, PS/2 KEYBOARD						
Reference	Qty	Value	Description	Digikey part number	Alt. part?	Mount
A1	1	RaspberryPi_Pico	MCU		Check op	TPH
C1,C10	2	100n	Supply decoupling	445-173588-1-ND	Y	TPH
C7,C9	2	10u	Supply line smoothing	399-18269-1-ND	Y	TPH
D1,D2,D3	3	1N4148	VGA RGB output level setting	1N4148FS-ND	Y	TPH
J18	1	Conn_01x02_Pin	RUN switch	S1022EC-40-ND	Y	TPH
JP3	1	Link pin2-pin3	Select 3.3V supply from Pico SMPS	S1022EC-40-ND	Y	TPH
JP6.1,JP6.2	1	Link pin2-pin3	Select Picomite or Win palette	S1022EC-40-ND	Y	TPH
J3	1	DE15_Socket_HD_MountingHoles	VGA output connector	L77HDE15SD1CH4FVGA-ND	Check FP	TPH
J6	1	Mini-DIN-6	PS/2 keyboard connector	CP-2260-ND	Check FP	TPH
JP5	1	Conn_01x02_Pin	JP5.1-open JP5.2-linked. Raspberry Pi Pico SMPS 3.3V supply enabled	S1022EC-40-ND	Y	TPH
Q5,Q6	2	2N7000	PS/2 level shifting	2N7000BU-ND	Y	TPH
R1,R6,R9,R12,R13,R14,R17,R18,R61,R62	10	390R	VGA RGB output level setting. SMPS enable	13-MFR-25FTE52-390RCT-ND	Y	TPH
R15,R16,R19,R20,R57,R60	10	10k	PS/2 keyboard i/face. Raspberry Pi Pico SMPS enable & RUN	13-MFR-25FRF52-10KCT-ND	Y	TPH

OPTIONS AND ALTERNATIVES						
I/O OPTIONS						
Digital GPIO						
Reference	Qty	Value	Description	Digikey part number		
D4,D5,D6,D7,D8,D9,D10,D11,D12,D13	10	1N4148	General-purpose signal/rectifier diode	1N4148FS-ND	Y	TPH
J7	1	Conn_01x10_Pin	Digital I/O connector	S1022EC-40-ND	Y	TPH
R21,R22,R23,R24,R27	5	220R	Current limit resistors	13-MFR-25FRF52-220RCT-ND	Y	TPH

Analogue Input						
Reference	Qty	Value	Description	Digikey part number		
C16,C25,C34,C40	4	1n	Input R-C filter capacitors. Vref feedback	R82EC1100SH50J	Y	TPH
C19	1	10u	Supply line smoothing for op-amp	399-18269-1-ND	Y	TPH
C20,C39	2	100n	Supply and Vref decoupling	445-173588-1-ND	Y	TPH
D15,D17,D19,D20,D21,D24	4	1N4148	Analogue input voltage limiters	1N4148FS-ND	Y	TPH
J11	1	Conn_01x08_Pin	Analogue input connector	S1022EC-40-ND	Y	TPH
R31,R32,R33,R41,R42,R46,R47	7	3k3	Input R-C filter resistors. Op-amp to ADC inputs. Vref feedback	13-MFR-25FTE52-3K3CT-ND	Y	TPH
R58	1	220R	Vref output current limiter	13-MFR-25FTE52-470RCT-ND	Y	TPH
U7	1	MCP6489	Quad op-amp. Analog input and ADC voltage reference output buffer	150-MCP6489-E/SL-ND	Y	SMT

Alternative: I/O protection Schottky diodes						
Reference	Qty	Value	Description	Digikey part number		
D4,D5,D6,D7,D8,D9,D10,D11,D12,D13,D15,D17,D19,D20,D21,D24	16	BAT43	General-purpose Schottky diodes. Lower Vf for better voltage limiting	112-BAT43-TAPCT-ND	Y	TPH

Alternative: I/O protection zeners						
Reference	Qty	Value	Description	Digikey part number		
D9,D10,D11,D12,D13,D17,D20,D24	8	1N5226BTR	3.3 V Zener diode	1N5226BFSCCT-ND	Y	TPH
D4,D5,D6,D7,D8,D15,D19,D21	8	DNF		DNF	Y	

POWER SUPPLY OPTIONS

Select External supply or the Internal supply from Pico USB

External +5V PSU

Reference	Qty	Value	Description	Digikey part number		
C6	1	10u	Supply line smoothing, audio output coupling	399-18269-1-ND	Y	TPH
D14	1	5.6V 1N5338	5.6 V Zener Supply voltage protection	1N5339BRLGOSCT-ND	Y	TPH
D25	1	1N5818	Schottky Supply polarity protection	497-4548-1-ND	Y	TPH
F1	1	1.1A 60R110XU	Overcurrent protection	F2010-ND	Y	TPH
J9	1	Conn_01x02_Pin	External PSU wired connector	S1022EC-40-ND	Y	TPH
J21	1	Barrel_Jack_Switch	DC power supply connector	732-5930-ND	Check FP	SMT
JP4	1	Jumper_3_Open	Link pins 1-2 for 5V supply	S1022EC-40-ND	Y	TPH
R59	1	3k3	DC power connector switch sense	13-MFR-25FTE52-3K3CT-ND	Y	TPH

External >5V PSU

Reference	Qty	Value	Description	Digikey part number		
C6	1	10u	Supply line smoothing, audio output coupling	399-18269-1-ND	Y	TPH
D14	1	5.6V 1N5338	5.6 V Zener Supply voltage protection	1N5339BRLGOSCT-ND	Y	TPH
D25, D26	2	1N5818	Schottky Supply polarity protection	497-4548-1-ND	Y	TPH
F1	1	1.1A 60R110XU	Overcurrent protection	F2010-ND	Y	TPH
J9	1	Conn_01x02_Pin	External PSU wired connector	S1022EC-40-ND	Y	TPH
J21	1	Barrel_Jack_Switch	DC power supply connector	732-5930-ND	Check FP	SMT
JP4	1	Jumper_3_Open	Link pins 2-3 for >6V supply	S1022EC-40-ND	Y	TPH
R59	1	3k3	DC power connector switch sense	13-MFR-25FTE52-3K3CT-ND	Y	TPH
U1	1	LM7805_TO220	5 V regulator. Input power regulation	497-1443-5-ND	Check op	TPH

+3.3VA Linear analogue supply

Reference	Qty	Value	Description	Digikey part number		
C5	1	100n	Supply decoupling	445-173588-1-ND	Y	TPH
C4	1	10u	Supply line smoothing, audio output coupling	399-18269-1-ND	Y	TPH
JP3	1	Jumper_3_Open	System 3.3V supply voltage select: Link 1-2: U5 Linear 3.3V supply Link 2-3: Shared with Pi Pico SMPS 3.3V supply	S1022EC-40-ND	Y	TPH
JP7	1	Jumper_3_Open	Analogue 3.3V supply voltage select: Link 1-2: U4 Linear 3.3V supply Link 2-3: Shared with System 3.3V supply	S1022EC-40-ND	Y	TPH
U4	1	LD1117V33C	3.3 V linear regulator. Audio output/Analogue input circuit supply rail	497-1492-5-ND	Check op	TPH

+3.3V Linear digital system supply

Reference	Qty	Value	Description	Digikey part number		
JP3	1	Jumper_3_Open	Link pins 1-2 for 3.3V linear 3.3V system supply	S1022EC-40-ND	Y	TPH
JP5	1	Conn_01x02_Pin	Pi Pico 3.3V supply select: Link JP5.1: Linear 3.3V supply, Pi Pico SMPS disabled Link JP5.2: Pi Pico 3.3V SMPS enabled	S1022EC-40-ND	Y	TPH
U5	1	LD1117V33C	3.3 V linear regulator. Internal 3.3V system supply rail	497-1492-5-ND	Check op	TPH

+5V Tap

Reference	Qty	Value	Description	Digikey part number		
J15	1	Conn_01x02_Pin	External PSU wired connector	S1022EC-40-ND	Y	TPH

I2C INTERFACE OPTIONS

External I2C channel “I2C”						
Reference	Qty	Value	Description	Digikey part number		
R2,R3,R7,R10	4	10k	SCL and SDA pull-up resistors	13-MFR-25FRF52-10KCT-ND	Y	TPH
J1	1	Conn_01x04_Pin	Offboard bare connector	S1022EC-40-ND	Y	TPH
J20	1	Conn_01x04_MountingPin	Offboard I2C plug-in module connector	455-SM04B-SRSS-TBCT-ND	Y	SMT
JP1	1	Jumper_3_Open	External I2C voltage select: Link 1-2: 5V Link 2-3: 3.3V	S1022EC-40-ND	Y	TPH
Q1,Q3	2	2N7000	I2C level shifting. LCD backlight switch	2N7000BU-ND	Y	TPH

Internal system/External I2C channel “I2C2” (needed for onboard LCD, Expanded DIO)						
Reference	Qty	Value	Description	Digikey part number		
R4,R5,R8,R11	4	10k	SCL and SDA pull-up resistors	13-MFR-25FRF52-10KCT-ND	Y	TPH
J2	1	Conn_01x04_Pin	Offboard bare connector	S1022EC-40-ND	Y	TPH
J19	1	Conn_01x04_MountingPin	Offboard I2C plug-in module connector	455-SM04B-SRSS-TBCT-ND	Y	SMT
J22	1	Conn_01x04_Pin	Internal I2C system bus bare connector	S1022EC-40-ND	Y	TPH
JP2	1	Jumper_3_Open	External I2C voltage select: Link 1-2: 5V Link 2-3: 3.3V	S1022EC-40-ND	Y	TPH
Q2,Q4	2	2N7000	I2C level shifting. LCD backlight switch	2N7000BU-ND	Y	TPH

Expanded DIO (System I2C2 Addr 63)						
Reference	Qty	Value	Description	Digikey part number		
C3	1	100n	Supply decoupling	445-173588-1-ND	Y	TPH
J7	1	Conn_01x10_Pin	Expanded Digital I/O connector	S1111EC-40-ND	Y	TPH
U3	1	PCF8574AN	I2C system Digital I/O	296-13106-5-ND	Check op	TPH

Onboard I2C LCD interface (System I2C2 Addr 62)						
Reference	Qty	Value	Description	Digikey part number		
C11	1	100n	Supply decoupling	445-173588-1-ND	Y	TPH
J8,J10	2	Conn_01x06_Pin	LCD connectors	S1022EC-40-ND	Y	TPH
Q7,Q8,Q9	3	2N7000	I2C level shifting to 5V. LCD backlight switch	2N7000BU-ND	Y	TPH
R25,R26,R29	3	10k	I2C signal pullup resistors. Backlight enable	13-MFR-25FRF52-10KCT-ND	Y	TPH
R28	1	390R	LCD backlight LED bias	13-MFR-25FTE52-390RCT-ND	Y	TPH
RV1	1	10k	LCD contrast control	1993-1116-ND	Y	TPH
U6	1	PCF8574AN	I2C system LCD interface.	296-13106-5-ND	Check op	TPH

AUDIO OPTIONS

Audio option 1: Minimal audio (R-C filter to off-board pin connector)

Reference	Qty	Value	Description	Digikey part number		
C22,C36	2	1n	Audio filter capacitors	R82EC1100SH50J	Y	TPH
C28,C35	2	10u	Audio output coupling	399-18269-1-ND	Y	TPH
J13	1	Conn_01x04_Pin	Audio output expansion	S1022EC-40-ND	Y	TPH
R35,R49	2	10k	Audio filter resistors	13-MFR-25FRF52-10KCT-ND	Y	TPH
R45,R48	2	Wire Link,	Headphone output resistor bypass			TPH
Link1	1	Wire link, A-A	Filter bypass link audio_l			TPH
Link2	1	Wire link, B-B	Filter bypass link audio_r			TPH

Audio option 2: Filtered and amplified audio (S&K filter to off-board pin connector, headphone connector and speaker amp)

Reference	Qty	Value	Description	Digikey part number		
C12,C13,C17,C18,C22,C23,C24,C30,C31,C32,C33,C36,C37,C38	14	1n	Audio filter capacitors	R82EC1100SH50J	Y	TPH
C14,C21,C26,C28,C29,C35	6	10u	Supply line smoothing, audio output coupling	399-18269-1-ND	Y	TPH
C15,C27	2	100n	Supply decoupling	445-173588-1-ND	Y	TPH
D16,D18,D22,D23	4	1N4148	PWM signal clipping diode	1N4148FS-ND	Y	TPH

J12,J14	2	Conn_01x02_Pin	Speaker output connectors	S1022EC-40-ND	Y	TPH
J13	1	Conn_01x04_Pin	Audio output expansion	S1022EC-40-ND	Y	TPH
J16	1	CUI/SameSky SJ1-3525	Stereo headphone output connector	CP1-3525NG-GR	Check FP	TPH
R30,R40,R44,R56	4	3k3	PWM signal bias	13-MFR-25FTE52-3K3CT-ND	Y	TPH
R34,R35,R36,R37,R38,R39,R43,R49,R50,R51,R52,R53,R54,R55	14	10k	Audio filter resistors	13-MFR-25FRF52-10KCT-ND	Y	TPH
R45,R48	2	390R	Headphone output resistors	13-MFR-25FTE52-390RCT-ND	Y	TPH
U8	1	MCP6489	Quad op-amp. Audio output filter.	150-MCP6489-E/SL-ND	Check op	SMT
U9,U10	2	LM4871	Audio class AB output amplifier	LM4871MX/NOPBCT-ND	Check op	SMT
MISC						
Reference	Qty	Value	Description	Digikey part number		
Hardwired PS/2 keyboard						
J4	1	Conn_01x04_Pin	Off-board connectors and jumpers	S1022EC-40-ND		TPH
SD Card						
Reference	Qty	Value	Description	Digikey part number		
C2	1	100n	Supply decoupling	445-173588-1-ND	Y	TPH
J5	1	Micro_SD_Card_Det_Hirose_DM3AT	SD card memory storage	26-DM3AT-SF-PEJM5(11)CT-ND	Check FP	SMT
Analogue Reference Voltage						
Reference	Qty	Value	Description	Digikey part number		
U2	1	LM4040LP-3	Analog voltage reference	296-39020-ND	Check op	THP
Miscellaneous						
Reference	Qty	Value	Description	Digikey part number		
Jumper links		Link	Jumper settings	A860AR-ND	Y	THP

Notes:

- Pin headers are cut from 40-wide strips
- Pin headers marked in RED can be changed for right-angled versions e.g. Digikey part S1111EC-40-ND if preferred for off-board connections
- Alternative parts can be used as per following key:

Y	- The same value and footprint should be used, but alternatives are possible
Check FP	- Check that any alternative component has the same footprint as the one used on the board
Check op	- Check that any proposed alternative has equivalent specifications and footprint before using

Appendix C Code Samples

To start with a disclaimer, I make no claims to be much of a programmer. I have attempted to make these samples as easy to read as possible, rather than pretty. There is plenty of scope to make the code slicker and more efficient.

PBC DIO test.bas

This program reads the five data IO pins. If wired with pushbuttons that connect each GPIO pin to 0V, this demonstrates a simple game controller. You may need to add pull-up resistors to the switches, as the Raspberry Pi Pico pullups are fairly weak, or if the switches are to permanent then swap the DIO diodes for pullup- or pulldown resistors as required by your application.

```
const gpio_0 = 14
const gpio_1 = 15
const gpio_2 = 16
const gpio_3 = 17
const gpio_4 = 29

'Set up the DIO pins, enable the internal weak pullup resistors
setpin gpio_0, din, pullup
setpin gpio_1, din, pullup
setpin gpio_2, din, pullup
setpin gpio_3, din, pullup
setpin gpio_4, din, pullup

print "Press any key to START. Once started press any key to STOP"
do
loop until inkey$ <> ""

do
  if pin(gpio_0) = 0 then
    print "left:";
  end if

  if pin(gpio_1) = 0 then
    print "down:";
  end if

  if pin(gpio_2) = 0 then
    print "up:";
  end if

  if pin(gpio_3) = 0 then
    print "right:";
  end if

  if pin(gpio_4) = 0 then
    print "FIRE!"
  end if

  print
loop until inkey$ <> ""

end
```

PBC Check AIN.bas

This example reads all three ADC pins and displays a value for each. Connect one of the fixed terminals of a potentiometer to the Vref pin and the other fixed terminal to 0V. Connect the wiper to any of the ADC pins and read off the values as the potentiometer is adjusted.

'Set up the three analog input pins

setpin 31, ain

setpin 32, ain

setpin 34, ain

print "Connect a pot(s) between VRef and GND, with wiper(s) to AIN0..AIN2"

print "Press any key to start"

do

loop until inkey\$ <> ""

print "Press any key to end"

pause 1000

do

ain_0 = (int(pin(31)*100))/100 **'round each value to 2 decimal places**

ain_1 = (int(pin(32)*100))/100

ain_2 = (int(pin(34)*100))/100

print "AIN_0=";ain_0;" AIN_1=";ain_1;" AIN_2=";ain_2

pause 200

loop until inkey\$ <> ""

end

PBC I2C scan.bas

This example steps through each address on both I2C buses and indicates if there is a response from a connected device. Useful for checking whether a module is connected and working properly e.g. the correct supply voltage configuration has been selected.

'This assumes the System I2C has been configured to use GP2 and GP3

'by using the OPTION SYSTEM I2C GP2, GP3, SLOW command

'The system I2C bus is referenced as I2C2 in programs, as GP2 and GP3 are

'tied to I2C2 identities

'Addresses of some known modules have been singled out

SetPin gp0,gp1,i2c

I2C open 100,1000

For n = 0 To 127

I2C check n

If MM.I2C = 0 Then

Print "I2C Addr: "; n; " present"

```
End If
Pause 10
Next n
```

```
'If the system I2C bus has not already been OPTION'd, use these to enable I2C2
'SetPin gp2,gp3,i2c2
'I2C2 open 100,1000
```

```
For n = 0 To 127
```

```
  I2C2 check n
```

```
  If MM.I2C = 0 Then
```

```
    select case n
```

```
      case 32
```

```
        print "I2C2 addr: 32 PCF8575 DIO expander present"
```

```
      case 48
```

```
        print "I2C2 addr: 48 NEO Slider present"
```

```
      case 52
```

```
        print "I2C2 addr: 52 TCA8418 keyboard matrix present"
```

```
      case 56
```

```
        print "I2C2 addr: 56 AHT20 temp & humidity sensor present"
```

```
      case 62
```

```
        print "I2C2 addr: 62 Onboard I2C LCD interface present"
```

```
      case 63
```

```
        print "I2C2 addr: 64 Onboard IO Expander present"
```

```
      case 94
```

```
        print "I2C2 addr: 94 TLV493D magnetometer present"
```

```
      case 98
```

```
        print "I2C2 addr: 98 MSA311 3-axis acceleratometer present"
```

```
      case 104
```

```
        print "I2C2 addr: 104 RTC present"
```

```
      case else
```

```
        print "I2C2 addr: ";n;" unknown device"
```

```
    end select
```

```
  End if
```

```
  pause 1
```

```
next n
```


PBC I2C expander.bas

This example toggles all the pins of the Onboard DIO Expander and reads back the pin value. Pin values are written to and read from I2C2 bus address 63₁₀, &H3F, 0b0011 1111.

'Include these if the system I2C bus has not been established as I2C2

'SetPin gp2,gp3,i2c2

'I2C2 open 100,1000

AllOn = &Hff

AllOff = &H00

do

I2C2 write 63,0,1,AllOn

pause 10

I2C2 read 63,0,1,InByte

print "In byte at 63 is: ";InByte

pause 500

I2C2 write 63,0,1,AllOff

pause 10

I2C2 read 63,0,1,InByte

print "In byte at 63 is: ";InByte

pause 500

loop until inkey\$ <> ""

end

PBC plcd demo.bas

This example demonstrates various routines needed to drive an LCD character display connected to the PLCD port. Communication is achieved by bit-banging half-byte data transfers to the upper four pins of the LCD 8-bit data lines using the dedicated I2C port expander (U6). Best practice recommends checking the “busy” status of the LCD between executing consecutive commands, however in practice the delay caused by the relatively slow I2C interface has been sufficient to allow the LCD functions used below to execute without problems.

Pin values are written to I2C2 bus address 62₁₀, &H3E, 0b0011 1110.

'I2c2 System I2C bus must be established

'The SUBS can be added to Basic programs to mimic the LCD print commands in PicoMite Basic.

'They could also be adapted and added to the Library e.g. PLCD line, pos, data\$

l_data = 0

cursorline = 0

cursorpos = 0

charloc = 0

textstring\$ = ""

do **'List menu options**

cls

print "1 - Initialise LCD"

Print "2 - Backlight on"

Print "3 - Backlight off"

print "4 - Clear display"

print "5 - Send character"

print "6 - Home cursor"

print "7 - Set cursor position"

print "8 - Print string"

Print "0 - Quit"

Input selection

select case selection

case 1

print "initialising PLCD"

send_ctrl(3) 'force 0011 to set 8-bit i/face 3 times as 4-bit transfers

send_ctrl(3)

send_ctrl(3)

send_ctrl(2) 'send 0010 to set 4-bit i/face once as final 4-bit t

send_ctrl(2) 'send 0010 10xx for Function Set

send_ctrl(8) '4-bit i/face (undisturbed), 2 lines, 5x8 chars

send_ctrl(0) 'send 0000 1100 for Display Control

send_ctrl(12) 'display on, cursor off, blink off

send_ctrl(0) 'send 0000 0001 Clear Display

send_ctrl(1)

send_ctrl(0) 'send 0000 0111 Increment DDRAM, no shift

send_ctrl(6)

```

        send_ctrl(0)      'send 0000 0010 Home cursor
        send_ctrl(2)

case 2
    PLCD_lon()           'turn on backlight

case 3
    PLCD_loff()          'turn off backlight

case 4
    send_ctrl(0)         'clear display
    send_ctrl(1)

    pause 10

    send_ctrl(0)
    send_ctrl(2)

case 5
    send_char("H")       'Sends a single character "H" to the display

case 6
    send_ctrl(0)         'moves the cursor to 1st position, 1st line
    send_ctrl(2)

case 7
    input "Enter Position: ", cursorpos
    move_cursor(cursorpos)

case 8
    'this mimics the LCD command in MMBasic
    input "Enter line: ", cursorline

    if cursorline > 3 then
        cursorline = 3
    end if

    input "Enter start position (0-19): ", cursorpos
    if cursorpos > 19 then
        cursorpos = 19
    end if

    input "Enter string: ", textstring$
    select case cursorline
        case 1
            cursorpos = cursorpos + &h40 'start posn 1st row in hex
        case 2
            cursorpos = cursorpos + &h14 'start posn 2nd row in hex
        case 3
            cursorpos = cursorpos + &h54 'start posn 3rd row in hex
        case else
            'leave posn of 0th row
    end select

    move_cursor(cursorpos)

    for charloc = 1 to len(textstring$) 'relies on LCD autoincrementing cursor posn
        send_char(mid$(textstring$,charloc,1))
    next charloc

```

```

    case 0
        'End demo
    end

end select

loop

Sub PLCD_on()
    I2C2 write 62,0,1,8
End Sub

Sub PLCD_loff()
    I2C2 write 62,0,1,0
End Sub

sub send_ctrl(l_data)
    local lcd_temp
    i2c2 read 62,0,1,lcd_temp
    lcd_temp = lcd_temp AND &B00001110 'mask the 4 control bits, clearing RS to
    l_data = l_data<<4      'shift data to upper 4 bits
    l_data = l_data OR lcd_temp
    write_plcd_ctrl(l_data)
    l_data = 0
end sub

sub send_char(l_data$)
    local lcd_temp = 0
    lcd_temp = asc(l_data$)
    write_plcd_data (lcd_temp)
    l_data$ = ""
end sub

sub write_plcd_ctrl(plcd_data)
    plcd_data = plcd_data AND &B11111000 'clear EN, R|/W and RS to CTRL register
    i2c2 write 62,0,1,plcd_data
    pause 1
    plcd_data = plcd_data OR &B00000100 'toggle E pin to write, 1 ms puls
    i2c2 write 62,0,1,plcd_data
    pause 1
    plcd_data = plcd_data AND &B11111000 'keeping b0-data off
    i2c2 write 62,0,1,plcd_data
    pause 1
end sub

sub write_plcd_data(plcd_data) 'takes 8-bit data, slices in to 2 x 4bits

    local lcd_u_nibble, lcd_l_nibble 'upper and lower nibbles

    i2c2 read 62,0,1,lcd_temp      'read current pin states on port

```

```

lcd_temp = lcd_temp AND &B00001001 'clear EN and R|/W and set RS to DATA
lcd_temp = lcd_temp OR &B00000001 'ensure set RS to DATA register

lcd_u_nibble = plcd_data and &B11110000 'mask off upper data nibble
lcd_l_nibble = plcd_data << 4

plcd_data = lcd_u_nibble OR lcd_temp 'prep upper nibble
i2c2 write 62,0,1,plcd_data

plcd_data = plcd_data OR &B00000100 'toggle E pin to write, 1 ms puls
i2c2 write 62,0,1,plcd_data

plcd_data = plcd_data AND &B11111001 'keeping b0-data on
i2c2 write 62,0,1,plcd_data

plcd_data = lcd_l_nibble OR lcd_temp 'prep lower nibble
i2c2 write 62,0,1,plcd_data

plcd_data = plcd_data OR &B00000100 'toggle E pin to write, 1 ms puls
i2c2 write 62,0,1,plcd_data

plcd_data = plcd_data AND &B11111001 'keeping b0-data on
i2c2 write 62,0,1,plcd_data

    pause 1 'better to check LCD busy flag, but this works OK
end sub

sub move_cursor(cursorpos)
    cursorposlow = cursorpos and &b00001111
    cursorposhigh = cursorpos or &B10000000 'set ctrl bit to DDRAM addr reg
    cursorposhigh = cursorposhigh >>4 'send upper nibble to lower nibble

    send_ctrl(cursorposhigh)
    send_ctrl(cursorposlow)
end sub

```

Real-Time Clock

These examples demonstrate using a Real-Time Clock connected to the system I2C.

PBC Set RTC.bas

This example prompts the user to configure the time on the RTC device. It then prompts the user to commit that time to the Raspberry Pi Pico.

```
input "Enter year (e.g. 2025): ",yr_val
input "Enter month (e.g. 09 for September): ",mo_val
input "Enter date (e.g. 05 for 5th): ",dy_val
input "Enter hour (e.g. 16 for 4pm): ",hr_val
input "Enter minute (0 .. 60): ",mn_val
input "Enter second (0 .. 59): ",sc_val

RTC settime yr_val, mo_val, dy_val, hr_val, mn_val, sc_val

print "Commit this time to the PI? (Y/N)"

do
  response$ = ucase$(inkey$)
  if response$ = "Y" then
    RTC gettime
    response$ = "N"
  endif
loop until response$ = "N"

end
```

PBC Read RTC.bas

This example asks the user whether they need to update the time registered on the Raspberry Pi Pico with the current data from RTC, before displaying the date and time currently held in Raspberry Pi Pico.

```
print "Update PI from RTC? (Y/N)"

do
  response$ = ucase$(inkey$)
  if response$ = "Y" then
    RTC gettime
    response$ = "N"
  endif
loop until response$ = "N"

print Date$
print Time$
```