TILE-LED36 Ref. Manual

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
	2019-Mar-25		tsp

Contents

1 Ref. Manual

Product	**TILE-LED36**
Rev	3.0
Date	2019-Apr-01
Author	tsp

2 Physical Properties

TILE-LED36 is a 6x6 RGB LED matrix based on the PYBD TILE form factor (12mm x 12mm).

It's 108 individual LEDs are controlled by a small MCU that also takes care of the host communication.

Each individual LED has it's individual PWM modulator and is capable of displaying more than 1000 brightness levels.

All 108 LEDs are activated about 100 times per second.

Software running on the MCU uses certain values out of the more than 1000 possible brightness levels to approach a perception of linear brightness in response to an 8-bit code provided by the user.

With this technique a flicker free (refresh rate > 100Hz) balanced color can be achieved.

At the same time the MCU optimizes current consumption in a way that allows to operate all 108 LEDs at maximum brightness at a total current consumption of less than 10mA.

Although the total brightness is limited due to the high multiplexing rate, the display is very well suitable for indoor use.

2.1 Key Features

- simple control via I2C
- user definable I2C address
- common broadcast address for multiple LED36 on a single I2C bus
- integrated font generator
- content scrolling in four directions
- one byte synchronization via I2C broadcast address

2.2 Operation Modes

The LED36 uses a single I2C interface and usually a 3.3 Volt power source.

Multiple LED36 devices can be operated with one four wire interface.

Each LED36 can be operated as an individual 6x6 RGB display.

Even in this mode multiple LED36 can be operated on one bus and still be addressed individually.

Multiple LED36 can be operated in linked mode.

Each of the individual tiles has its own I2C address.

The first tile uses the default address 60, other tiles, currently up to 16, use successive addresses.

A host controller can then send a text string via the broadcast address to all tiles in one operation.

A single bate broadcast to the I2C bus will then advance the display content across boarders of the available LED32 tiles.

2.3 I2C Status

The I2C communication port provides two addresses.

The primary address can be defined by the user, it's default value is 60.

A second, broadcast address is also avaialable at fixed address 1.

Any status information can only be read via the non broadcast address.

2.4 I2C Read Data

The LED36 output buffer can be read anytime.

The first byte received by the host contains the number of bytes available in the send queue of the LED36.

Reading just one byte results in getting the number of bytes in the output queue.

Reading N bytes still gives the number of available bytes in the first position followed by N-1 bytes from the output queue.

If N-1 exceeds the number of bytes available, trailing 0x00 bytes are transmitted.

To clear the LED36 output buffer data has to be read until the first byte received is less than the number of bytes requested.

3 Power Up

After power up LED36 waits for a SOH character (0x01) to start operation. It doesn't matter which of the two I2C addresses receives this character. All connected LED36 modules can be activated with a single byte sent to the I2C broadcast address 1.

4 Commands

All bytes entered through either I2C channel are treated as ASCII characters and silently ignored.

The host is responsible to avoid buffer overflow and protocol sequence errors in case more than one channel is used.

The available space of the input buffer can be polled.

The queue status is reported as one byte giving the available space (0..255).

If more than 253 bytes available, 254 is reported.

If the input queue is empty (eg. all commands processed) 255 is reported.

Note:

The queue status register is not yet implemented.

Commands are initiated by an STX character (0x02).

If enabled at compilation time, an ENQ character (0x05) enters debug mode.

The special one byte command (SOH, 0x01) is used to increment the sroll counter.

Screen coordinates are specified as uint8_t values.

A timeout of 5 s is applied for all composite commands. After timeout the parser falls back to ASCII mode.

Parameter values are of type uint8 except specified otherwise.

4.1 Basic commands

Table 1: Basic Commands

Type	Cmd	nParam	Description	Notes	Result
1	-	-	trigger text buffer roll		
2	0x11	1	set scroll text		
2	0x14	1	set matrix rotation	0..3	
2	0x16	1	set brightness	0..255 0..255%	
2	0x1a	0	enter bootloader		
2	0x2e	0	roll textbuf		
2	A	3	set pixel	R G B	
2	X	2	pos	P1..P2: <x><y>	
2	Y	4	sysres	P1 0xdeadbeef:int32	
2	c	6	set_text_color	fgred fg_green fg_blue bg_red bg_green bg_blue	
2	f	1	save to flash	P1 *n*: saves NVRAM to flash, *N*: factory reset	
2	g	1	get status, P1: ['1','2',-]	'1': get SW version, '2': get serial & I2C, -: <w><h><hf>0x4c	31, 20 or 4 bytes
2	i	3	set matrix	R G B	
2	k	?	text_buf colorwheel	P1: is number of bytes to follow, reset text_buf index	
2	l	?	text_buf	P1: is number of bytes to follow, reset text_buf index	
2	m	?	fill matrix	P1: is number of bytes to follow (RGB gamma corrected)	
2	n	?	fill matrix raw	P1: is number of bytes to follow (RGB)	

4.2 Setup commands

Table 2: Setup commands

Type	Cmd	nParam	Description	Notes	Result
2	0x0e	4	set I2C address	0x49 0x32 0x43 (addr*2) & 0xfc. Requires saving nvram and reset.	
2	0x1c	1	set RGB mode	0: CREE, 1: WE	permanent after NVRAM save
2	f	1	save to flash	P1 *n*: saves NVRAM to flash, *N*: factory reset	

4.3 Advanced commands

Table 3: Advanced commands

Type	Cmd	nParam	Description	Notes	Result

4.4 Obsolete commands

These commands will be removed in future versions.

Table 4: Obsolete commands

Type	Cmd	nParam	Description	Notes	Result

5 Colors

LED36 operates in 24-bit RGB mode.

The 8-bit brightness values are selected out of more than 1000 internal brightness values to mimic gamma corrected visual perception.

There is a raw mode available that circumvents the intensity correction.

6 Fonts

LED36 has a builtin font.

Table 5: Available Fonts

Font	Size	Glyphs	Description
0	5x6	0x20..0x7e	tiny font

7 NVRAM Parameters

A number of parameters are saved in non-volatile RAM.

Table 6: NVRAM Parameters

Parameter	Format	Values	Default	Description	Notes
i2c address	uint8	0x00, 0x20..0xfc	0xc4	individual I2C address	default addresses are 60 and 1
brightness	uint8	0..255	100	default global brightness	default brighntess 100%

Note:

If NVRAM parameters are saved, default brightness will also be saved. If there was no explicit set brightness command, any value that had been saved previously will remain active. If this value happens to be zero (0 %), the LED36 will stay dark until brightness is set to a non-zero value.

8 Firmware Upgrade

LED36 firmware can be upgraded in the field through appropriate MicroPython scripts supplied by the manufacturer.

9 Technical Specifications

36 RGB LEDs in a 6x6 matrix 108 PWM channels with 10-bit resolution Single I2C interface, no extra control lines

9.1 Power Supply

- LED36 can be safely operated from 3.0 V to 3.6 V
- Typical power consumption is from 3mA to 10mA (peak)
- Power consumption depends on LED brightness

9.2 Ports

All I/O ports operate at 3.3 V. Due to the nature of I2C interfaces the I2C lines may be operated with signals up to 5V.

9.3 Connectors

LED36 uses standard TILE connectors (HIROSE DF40-20).

9.4 Communication

- Fast I2C interface (400 kHz), external pullup resistors required.

9.5 Code Samples

```
"""
from LED36_examples import *
"""
from micropython import const
import machine
import pyb

LED_ADDR = const(60)        # default LED36 address

i2c = machine.I2C(1)        # select I2C1

def cyc(dt=250, addr=1):
    """ set all LEDs to black, red, green, yellow, blue, magenta, cayn and white for dt ms
        ramp up brightnes from 0 % to 100 %
    """
    while True:
        try:
            fill_rgb(100,100,100)
            break
        except:
            pyb.delay(100)
    for i in range(8):
        fill_rgb((i & 1)*255, ((i >> 1) & 1)*255, ((i >> 2) & 1)*255)
        pyb.delay(dt)
    for i in range(100):
        brightness(i)
        pyb.delay(20)

def save_nvram(addr=1):
    """ save NVRAM state
    """
    ba = bytearray(b'\x02fn')
    i2c.writeto(addr, ba)

def set_i2caddr(addr=60, oaddr=1):
    """ modify I2C address and save it to NVRAM """
    ba = bytearray(b'\x02\x0eI2C ')
    ba[-1] = (addr*2) & 0xfe
    i2c.writeto(oaddr, ba)
    save_nvram()

def brightness(b=100, addr=1):
    """ set brigntness
    """
    ba = bytearray(b'\x02\x16 ')
    ba[-1] = b & 0xff
    i2c.writeto(addr, ba)

def bloop(dt=100, maxv=100, inc=1):
    """ cycle through brigntness ramp
    """
    b = 0
    while True:
        print(b)
        brightness(b)
        b += inc
        b %= maxv
        pyb.delay(dt)
```

```
def pump(dt=10, maxv=100):
    """ cycle through brigntness modulation
    """
    import math
    sinar = []
    for i in range(90):
        sinar.append(int((math.sin(i*4/180*math.pi)+1)*maxv/2))
    i = 0
    while True:
        brightness(sinar[i])
        i += 1
        i %= len(sinar)
        pyb.delay(dt)

def fill_rgb(r=0x80, g=0, b=0, addr=1):
    """ fill LED array using set pixel command
    """
    i2c.writeto(addr, b'\x02X\x00\x00')
    buf = bytearray(b'\x02A   ')
    buf[2] = r
    buf[3] = g
    buf[4] = b
    for i in range(36):
        i2c.writeto(addr, buf)

def illu(r=0x80, g=0, b=0, addr=1):
    """ fill LED array using set illumination command
    """
    buf = bytearray(b'\x02i   ')
    buf[2] = r
    buf[3] = g
    buf[4] = b
    i2c.writeto(addr, buf)

def fill_frame(r=0x80, g=0, b=0, addr=1):
    """ fill LED array using fill frame command
    """
    i2c.writeto(addr, b'\x02ml')
    buf = bytearray(b'   ')
    buf[0] = r
    buf[1] = g
    buf[2] = b
    for i in range(36):
        i2c.writeto(addr, buf)

def set_dot(x=0, y=0, r=100, g=0, b=0, addr=1):
    """ set single LED color at position
    """
    buf = bytearray(b'\x02X  ')
    buf[2] = x
    buf[3] = y
    i2c.writeto(addr, buf)

    buf = bytearray(b'\x02A   ')
    buf[2] = r
    buf[3] = g
    buf[4] = b
    i2c.writeto(addr, buf)

def fill_raw(r=0x80, g=0, b=0, addr=1):
    """ fill LED array with raw values using fill frame command
    """
```

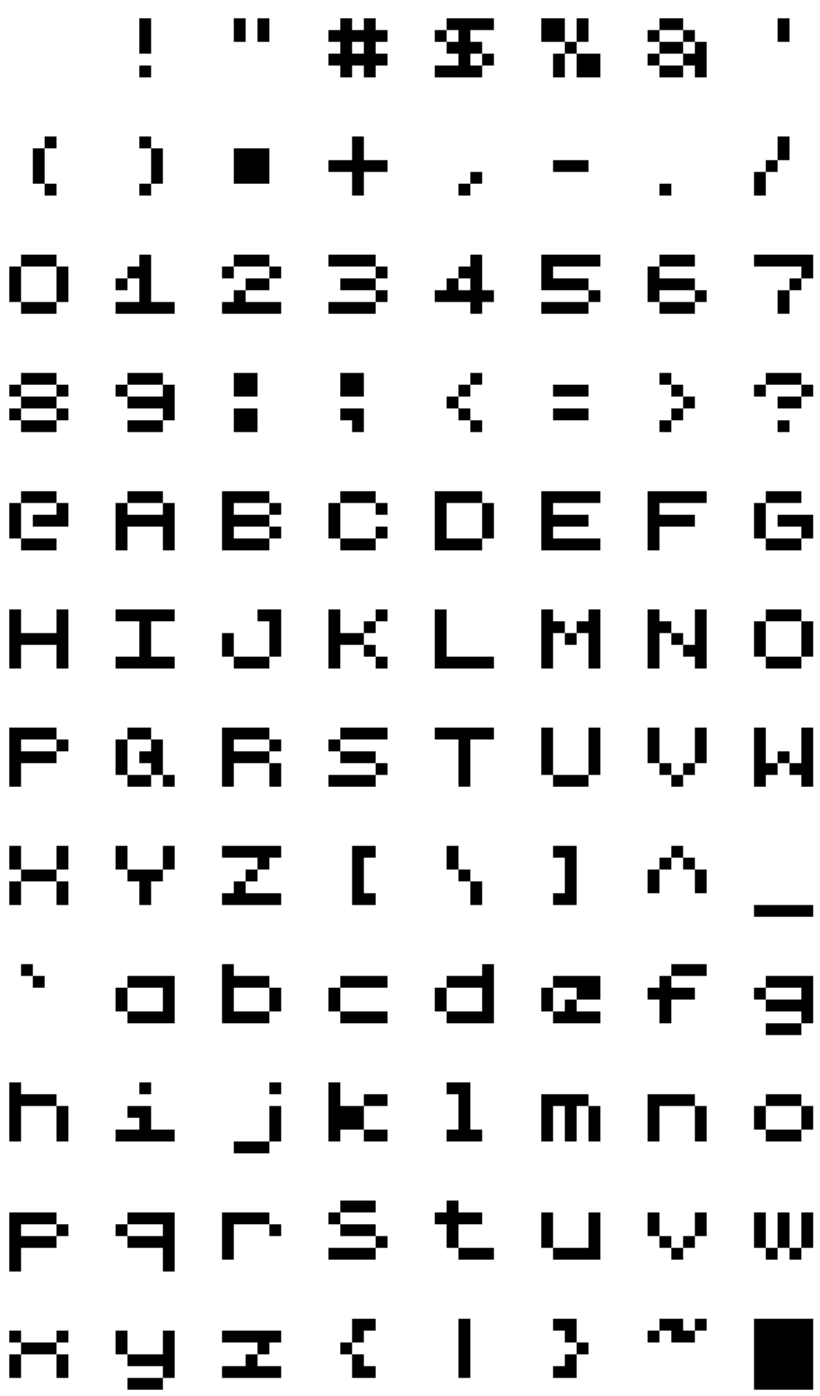
```
    i2c.writeto(addr, b'\x02nl')
    buf = bytearray(b'   ')
    buf[0] = r
    buf[1] = g
    buf[2] = b
    for i in range(36):
        i2c.writeto(addr, buf)

def ledpins(v=0, addr=1):
    """ permute LED colors (use with care)
    """
    buf = bytearray(b'\x02\x1c\x00')
    buf[-1] = v & 3
    i2c.writeto(addr, buf)

def random_dots(dt=10, addr=1):
    """ set random colors at random positions
    """
    while True:
        rn = pyb.rng()
        r = rn & 0xff
        g = (rn >> 8) & 0xff
        b = (rn >> 16) & 0xff
        x = (rn >> 24) % 36
        y = x // 6
        x %= 6
        set_dot(x, y, r, g, b, addr)
        pyb.delay(dt)
```

9.6 Font

The following are the glyphs used in the internal font.



10 Drawings

o.o.p.s.