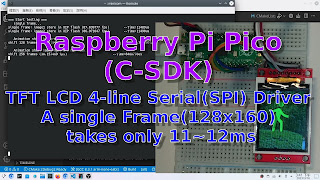
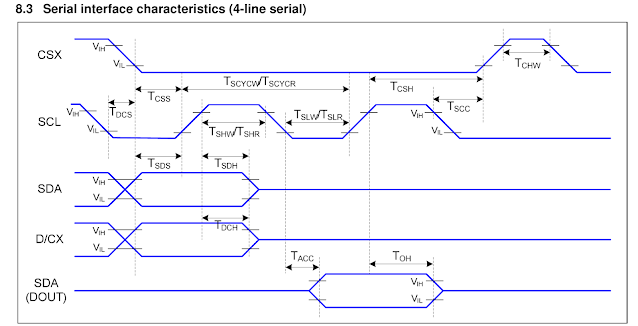
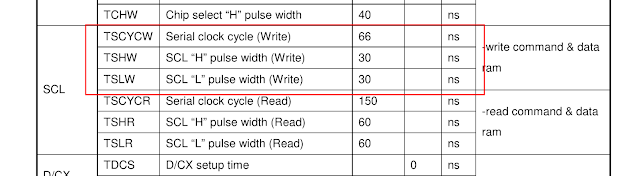
**-[Raspberry Pi Pico (c-sdk)] Display: Ep 5 :TFT LCD 4-lines Serial(SPI) Driver -- A single Frame(128x160) takes only 11~12ms**

本篇文章介紹使用RP2040 PIO(Raspberry Pi Pico開發板)功能，撰寫一個TFT LCD 4-line Serial Interface(SPI)驅動程式。有關8-bit parallel請參閱上篇文章[[Raspberry Pi Pico (c-sdk)] Display: Ep 2 : PIO TFT LCD ILI9341 8-bit parallel C-code Driver](https://rfwumcu.blogspot.com/2022/12/raspberry-pi-pico-c-sdk-display-ep-2.html)

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEg_NUE_fzlY-B_l0tUYkQ7vXhGNYJ1Tp53aDnUMgtITMmCJx2PENWnwjSTl-0A_0NhHB5G-8SR_85miWf6a8-SwBuWmyrlGjxKBfut8t-TxAjQMoD3u_rTTqnygjQwPQSxR9u0Jeix6XeUzYwwp3rgNMM10ktlvjcBRvtPKwNvJmmAX36PoCnhxN12x/s1920/yt-title.jpg)

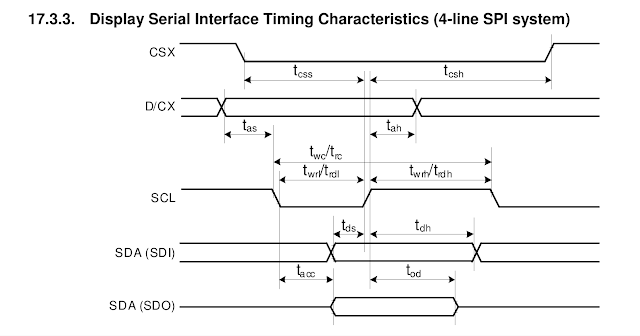
下圖示ST7735 datasheet中有關4-line interface的clock cycle。

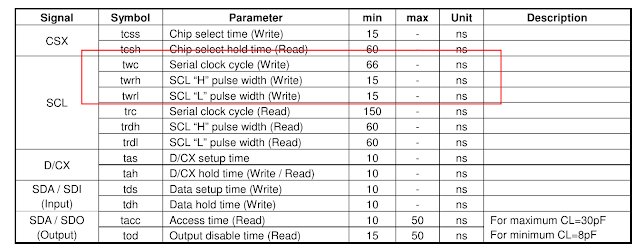
[](https://blogger.googleusercontent.com/img/a/AVvXsEhncT3Cg_U7Jny1NrnehHl6jSmDz8pxYAzZCFV3COg0Vqp3as-QkVuyZlcR_n3iapt5G9p073lHqhNaBpbU_s-qIv-TKjOuBLESHzIY2qTn10GBYnyMIWEwnHy0HFvXjeJrv0E6z9k7-mFua-KgQaR41QILTmc9M2h_MLbWg1nQ_WSI1N9sapKfSFJd)

[](https://blogger.googleusercontent.com/img/a/AVvXsEjsIFLlqBInaoNl4OMGh5YFL2Hoj2rbF-rtkno-RL-NBltP3zU2XJj0HorYiNTNoElN3gPX-PNtjPuB8lv6nEvsUEAmC5FGUttG3mZ35esb-_u0R186UtUi482wetbntF9QFrhLtR8Pyo8t9IccY_NKdO_e-iKAN7jz91sWZ8xr8ALYse53No9JbmZU)

SCL "H" and "L" pulse width MIN 為30ns。

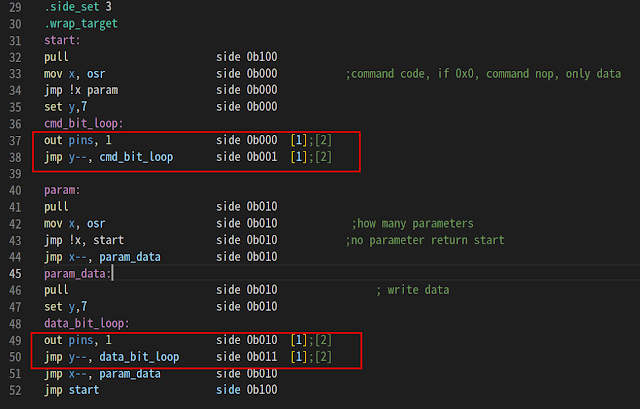
另外跟據ILI9486 Datasheet內容為：

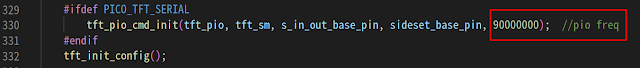
[](https://blogger.googleusercontent.com/img/a/AVvXsEisDssD0ERkMmDv05cyTRMYJKC_SIAshkZljoLr_xhiqECuwTXA0ewAcTHZe_IlwtJ-o7GHaXXP4oKm9EvpJheVZXS_oCvl6hUwC_WyyAogzW-SH2FD33a7ufr_Y1CYVmQqsAiNle3pQjbSfzSpLlqQWx5w6GnDwrmv61mFdu0pUefLOImzyh_PO5xo)

[](https://blogger.googleusercontent.com/img/a/AVvXsEhPR-eye-6QQBtedvYvLsbum_gcMluJQ4CtNsycJHp9-wkpvLJ_BOE1vjFh15tCTIymzmJ1_-qLqSJgd2XKKKfFDdos5tvHHCfcj-yTDESf6gsmI4Lo-7Y-oe8YdwrAHwB7raESebM0--ZjG6GZ-GnCe0gP9HmFXosDN23yiSXIHqlkMZALcGnkxLRB)

SCL "H" and "L" pulse width MIN 為15ns。

PIO程式碼設定約為22ns。如下圖所示：

[](https://blogger.googleusercontent.com/img/a/AVvXsEjMas3d1a8ZhUV_b6kkkhcykhtB5VRnMA9-tbm3GucYWum47Qwmc78UPXkVWPzldjTcnCtvnnGOIS2r_yeyhELPh5uWzPjizx_Kc1_YFcoLe_LSkHiH3KSB3TUOU-emHNECNPPGerumEZzd6MjWMTInDlHIQg2PBqECUNyPsLLOcoIpXvi8BV70g2kH)

[](https://blogger.googleusercontent.com/img/a/AVvXsEirBXpCZUsc52aE88khUAIeKH6EL5AYfNbBsHSBB3MAI8DSDzt7hWs5jH8yU1G5UGj0OclhkDuAJXYp6pKrkc3tvdbFAMkFeFsBmwUyY5vPxp6E0xEQtlQxC7pl09DDuq4JQd8jAL_0Ue_Kpq3dkPNhL9h9eUaSpcUdpRkpR2p1p09W096Akp8PGr2q)

以此設定輸出單一個128x160大小的frame 約需只要11~12ms，frame rate可達到80fps以上。

成果影片

程式碼：

pico\_tft.pio

.program tft\_pio\_parallel

; CSX, D/CX, WRX, RDX --> gpio 15, 14, 13, 12 (set pins)

.wrap\_target

start:

pull

set pins, 0b0011

mov x, osr ;command code, if 0x0, command nop, only data

jmp !x param

set pins, 0b0001

out pins, 8 ;[1]

set pins, 0b0011 ;[1]

param:

set pins, 0b0111

pull

mov x, osr ;how many parameters

jmp !x, start ;no parameter return start

jmp x--, param\_data

param\_data:

pull ; write data

set pins, 0b0101

out pins, 8 ;[1]

set pins, 0b0111 ;[1]

jmp x--, param\_data

set pins, 0b1111

jmp start

.program tft\_pio\_serial

; CSX, D/CX(A0), SCL --> gpio 15, 14, 13 (side-set pins)

.side\_set 3

.wrap\_target

start:

pull side 0b100

mov x, osr side 0b000 ;command code, if 0x0, command nop, only data

jmp !x param side 0b000

set y,7 side 0b000

cmd\_bit\_loop:

out pins, 1 side 0b000 [1];[2]

jmp y--, cmd\_bit\_loop side 0b001 [1];[2]

param:

pull side 0b010

mov x, osr side 0b010 ;how many parameters

jmp !x, start side 0b010 ;no parameter return start

jmp x--, param\_data side 0b010

param\_data:

pull side 0b010 ; write data

set y,7 side 0b010

data\_bit\_loop:

out pins, 1 side 0b010 [1];[2]

jmp y--, data\_bit\_loop side 0b011 [1];[2]

jmp x--, param\_data side 0b010

jmp start side 0b100

pico\_tft.c

#include "stdio.h"

#include "stdlib.h"

#include "pico/stdlib.h"

#include "hardware/clocks.h"

#include "string.h"

#include "registers.h"

#include "pico\_tft.pio.h"

#include "pico\_tft.h"

#include "fonts/font\_ubuntu\_mono\_24.h"

#include "hardware/dma.h"

#define MAX\_BYTE\_TRANS (TFT\_WIDTH\*TFT\_HEIGHT\*2)

#define PICO\_TFT\_SERIAL

//#define PICO\_TFT\_DMA

PIO tft\_pio = pio1;

uint tft\_sm=0;

uint in\_out\_base\_pin=4;

uint set\_base\_pin=12;

uint sideset\_base\_pin=11;

uint s\_in\_out\_base\_pin=10;

int tft\_dma\_channel;

/\*

\param frame\_buff: frame buffer

\param src: source image

\param x: start x of source image

\param y: start y of source image

\param fx: start x of drawing in frame buffer

\param fy: start y of drawing in frame buffer

\param ow: source image width

param oh: source image height

\*/

void fb\_draw\_image(uint8\_t\* frame\_buffer, uint8\_t \*src, uint16\_t x, uint16\_t y, uint16\_t fx, uint16\_t fy, uint32\_t ow, uint32\_t oh, bool wrapx, bool wrapy, bool enable\_transparent, uint16\_t transparent\_color) {

uint16\_t width = (ow > TFT\_WIDTH)? TFT\_WIDTH : ow;

uint16\_t height = (oh > TFT\_HEIGHT) ? TFT\_HEIGHT: oh;

if (!wrapx && (fx+width > TFT\_WIDTH)) width = TFT\_WIDTH-fx;

if (!wrapy && (fy+height) > TFT\_HEIGHT) height = TFT\_HEIGHT-fy;

uint32\_t src\_loc;

for (int j=0; j < height; j++) {

for (int i=0; i < width; i++) {

src\_loc=((x+i)%ow + ((y+j)%oh)\*ow)\*2;

if (!enable\_transparent || src[src\_loc]!=transparent\_color>>8 || src[src\_loc+1] != transparent\_color&0xff) {

frame\_buffer[((i+fx)+(j+fy)\*TFT\_WIDTH)\*2] = src[src\_loc];

frame\_buffer[((i+fx)+(j+fy)\*TFT\_WIDTH)\*2+1] = src[src\_loc+1];

}

}

}

}

/\*

\param src: source image

\param dest: dest image buffer

\param x: source image x, coordinates

\param y: source image y coordinates

\param ow: source image width

param oh: source image height

\param width: dest image width

\param height: dest image height

\*/

void tft\_fill\_partial\_image(uint8\_t \*src, uint8\_t \*dest,uint16\_t x, uint16\_t y, uint32\_t ow, uint32\_t oh,uint16\_t width, uint16\_t height) {

uint32\_t src\_loc;

for (int j=0; j < height; j++) {

for (int i=0; i < width; i++) {

src\_loc=((x+i)%ow + ((y+j)%oh)\*ow)\*2;

dest[(i+j\*width)\*2] = src[src\_loc];

dest[(i+j\*width)\*2+1] = src[src\_loc+1];

}

}

}

/\*

\param src: source image

\param x: start x position

\param y: start y position

\param ow: source image width

\param oy: source image height

\param wrapx: warp x

\param warpy: warp y

\*/

void tft\_draw\_frame(uint8\_t \*src, uint16\_t x, uint16\_t y, uint16\_t tx, uint16\_t ty, uint32\_t ow, uint32\_t oh) {

uint16\_t width = (tx+ow > TFT\_WIDTH)? TFT\_WIDTH-tx : ow;

uint16\_t height = (ty+oh > TFT\_HEIGHT) ? TFT\_HEIGHT-ty: oh;

uint32\_t src\_loc;

uint32\_t count = width\*height\*2;

tft\_set\_address\_window (tx, ty, tx+width-1, ty+height-1);

pio\_sm\_restart(tft\_pio, tft\_sm);

#ifdef PICO\_TFT\_SERIAL

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, TFT\_MEMORYWRITE << 24);

#endif

#ifdef PICO\_TFT\_PARALLEL

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, TFT\_MEMORYWRITE);

#endif

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, count);

for (int j=0; j < height; j++) {

for (int i=0; i < width; i++) {

src\_loc=((x+i)%ow + ((y+j)%oh)\*ow)\*2;

#ifdef PICO\_TFT\_SERIAL

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, src[src\_loc]<<24);

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, src[src\_loc+1]<<24);

#endif

#ifdef PICO\_TFT\_PARALLEL

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, src[src\_loc]);

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, src[src\_loc+1]);

#endif

}

}

}

void tft\_draw\_image\_with\_trasnparent\_color(uint16\_t x, uint16\_t y, const tImage \*bitmap, uint32\_t transparent\_color) {

uint16\_t width, height;

if (x+bitmap->width > TFT\_WIDTH) width = TFT\_WIDTH-x; else width = bitmap->width;

if (y+bitmap->height > TFT\_HEIGHT) height = TFT\_HEIGHT-y; else height = bitmap->height;

uint16\_t \*color=(uint16\_t\*)bitmap->data;

for (int j=0; j < height;j++) {

for (int i=0; i < width;i++) {

if (color[i+j\*bitmap->width] != transparent\_color) {

tft\_draw\_pixel(x+i,y+j,color[i+j\*(bitmap->width)]);

}

}

}

}

void tft\_cmd(uint32\_t cmd, uint32\_t count, uint8\_t \*param)

{

pio\_sm\_restart(tft\_pio, tft\_sm);

#ifdef PICO\_TFT\_SERIAL

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, cmd << 24);

#endif

#ifdef PICO\_TFT\_PARALLEL

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, cmd);

#endif

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, count);

for (int i = 0; i < count; i++)

{

#ifdef PICO\_TFT\_SERIAL

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, param[i]<<24);

#endif

#ifdef PICO\_TFT\_PARALLEL

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, param[i]);

#endif

}

}

#ifdef PICO\_TFT\_DMA

void tft\_cmd\_dma(uint32\_t cmd, uint32\_t count, uint8\_t \*param)

{

#ifdef PICO\_TFT\_SERIAL

tft\_cmd(cmd, count, param);

return;

#endif

pio\_sm\_restart(tft\_pio, tft\_sm);

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, cmd);

pio\_sm\_put\_blocking(tft\_pio, tft\_sm, count);

dma\_channel\_set\_trans\_count(tft\_dma\_channel, count >> DMA\_SIZE\_8, false);

dma\_channel\_set\_read\_addr(tft\_dma\_channel, param, false);

dma\_channel\_start(tft\_dma\_channel);

dma\_channel\_wait\_for\_finish\_blocking(tft\_dma\_channel);

}

#endif

void tft\_pio\_cmd\_init(PIO pio, uint sm, uint in\_out\_base, uint set\_sideset, uint32\_t freq) {

uint offset=0;

pio\_sm\_config c;

#ifdef PICO\_TFT\_PARALLEL

offset = pio\_add\_program(pio, &tft\_pio\_parallel\_program);

c = tft\_pio\_parallel\_program\_get\_default\_config(offset);

for (int i=0; i < 8; i++) pio\_gpio\_init(pio, in\_out\_base+i);

for (int i=0; i < 4; i++) pio\_gpio\_init(pio, set\_sideset+i);

pio\_sm\_set\_consecutive\_pindirs(pio, sm, in\_out\_base, 8, true);

pio\_sm\_set\_consecutive\_pindirs(pio, sm, set\_base, 4, true);

sm\_config\_set\_in\_pins(&c, in\_out\_base);

sm\_config\_set\_out\_pins(&c, in\_out\_base, 8);

sm\_config\_set\_set\_pins(&c, set\_sideset, 4);

sm\_config\_set\_out\_shift(&c, true, false, 8);

sm\_config\_set\_in\_shift(&c, false, false, 8);

#endif

#ifdef PICO\_TFT\_SERIAL

offset = pio\_add\_program(pio, &tft\_pio\_serial\_program);

c = tft\_pio\_serial\_program\_get\_default\_config(offset);

pio\_gpio\_init(pio, in\_out\_base);

for (int i=0; i < 3; i++) pio\_gpio\_init(pio, set\_sideset+i);

pio\_sm\_set\_consecutive\_pindirs(pio, sm, in\_out\_base, 1, true);

pio\_sm\_set\_consecutive\_pindirs(pio, sm, set\_sideset, 3, true);

sm\_config\_set\_in\_pins(&c, in\_out\_base);

sm\_config\_set\_out\_pins(&c, in\_out\_base, 1);

sm\_config\_set\_sideset\_pins(&c, set\_sideset);

sm\_config\_set\_out\_shift(&c, false, false, 8);

sm\_config\_set\_in\_shift(&c, true, false, 8);

#endif

//sm\_config\_set\_fifo\_join(&c, PIO\_FIFO\_JOIN\_TX);

float div = clock\_get\_hz(clk\_sys)/freq;

sm\_config\_set\_clkdiv(&c, div);

#ifdef PICO\_TFT\_DMA

/\* DMA \*/

tft\_dma\_channel = dma\_claim\_unused\_channel(true);

dma\_channel\_config dc = dma\_channel\_get\_default\_config(tft\_dma\_channel);

channel\_config\_set\_write\_increment(&dc, false);

channel\_config\_set\_read\_increment(&dc, true);

channel\_config\_set\_dreq(&dc, pio\_get\_dreq(pio, sm, true));

channel\_config\_set\_transfer\_data\_size(&dc, DMA\_SIZE\_8); //DMA\_SIZE\_8,16,32

dma\_channel\_configure(tft\_dma\_channel, &dc, (void\*) (PIO1\_BASE+PIO\_TXF0\_OFFSET),

NULL, MAX\_BYTE\_TRANS>> DMA\_SIZE\_8, false); //DMA\_SIZE\_8 or 16 or 32

/\* DMA \*/

#endif

pio\_sm\_init(pio, sm, offset, &c);

pio\_sm\_set\_enabled(pio, sm, true);

}

/\* ili3941 draw functions\*/

uint16\_t tft\_color\_565RGB(uint8\_t R, uint8\_t G, uint8\_t B) {

uint16\_t c;

c = (((uint16\_t)R)>>3)<<11 | (((uint16\_t)G)>>2) << 5 | ((uint16\_t)B)>>3;

return c;

}

void tft\_memory\_write\_window(uint16\_t x1, uint16\_t y1, uint16\_t x2, uint16\_t y2)

{

uint8\_t addr[4];

addr[0]=(uint8\_t)(x1 >> 8);

addr[1]= (uint8\_t)(x1 & 0xff);

addr[2]= (uint8\_t)(x2 >> 8);

addr[3]= (uint8\_t)(x2 & 0xff);

tft\_cmd(TFT\_COLADDRSET, 4, addr);

addr[0]=(uint8\_t)(y1 >> 8);

addr[1]= (uint8\_t)(y1 & 0xff);

addr[2]= (uint8\_t)(y2 >> 8);

addr[3]= (uint8\_t)(y2 & 0xff);

tft\_cmd(TFT\_PAGEADDRSET, 4, addr );

tft\_cmd(TFT\_MEMORYWRITE, 0, NULL);

}

void tft\_set\_address\_window(uint16\_t x1, uint16\_t y1, uint16\_t x2, uint16\_t y2)

{

uint8\_t addr[4];

addr[0]=(uint8\_t)(x1 >> 8);

addr[1]= (uint8\_t)(x1 & 0xff);

addr[2]= (uint8\_t)(x2 >> 8);

addr[3]= (uint8\_t)(x2 & 0xff);

tft\_cmd(TFT\_COLADDRSET, 4, addr);

addr[0]=(uint8\_t)(y1 >> 8);

addr[1]= (uint8\_t)(y1 & 0xff);

addr[2]= (uint8\_t)(y2 >> 8);

addr[3]= (uint8\_t)(y2 & 0xff);

tft\_cmd(TFT\_PAGEADDRSET, 4, addr );

}

/\* put color at point\*/

void tft\_draw\_pixel(uint16\_t x, uint16\_t y, uint16\_t color)

{

if ( x < 0 || x > TFT\_WIDTH-1 || y < 0 || y > TFT\_HEIGHT-1) return;

tft\_set\_address\_window(x,y,x,y);

tft\_cmd(TFT\_MEMORYWRITE, 2, (uint8\_t[2]){(uint8\_t)(color >> 8), (uint8\_t)color});

}

void tft\_fill\_rect(uint16\_t x, uint16\_t y, uint16\_t width, uint16\_t height, uint16\_t color) {

if (x < 0) x=0;

if (y < 0) y=0;

if (x+width > TFT\_WIDTH) width = TFT\_WIDTH-x;

if (y+height > TFT\_HEIGHT) height = TFT\_HEIGHT-y;

tft\_memory\_write\_window(x,y, x+width-1, y+height-1);

for (int j = y; j < y+height; j++)

for (int i = x; i< x+width;i++)

tft\_cmd(TFT\_NOP, 2, (uint8\_t[2]){(uint8\_t)(color >> 8), (uint8\_t)(color&0xff)});

}

void tft\_draw\_bitmap(uint16\_t x, uint16\_t y, const tImage \*bitmap)

{

uint32\_t width = 0, height = 0;

width = bitmap->width;

height = bitmap->height;

uint32\_t total\_pixels = width \* height\*2;

tft\_set\_address\_window (x, y, x + width-1, y + height-1);

tft\_cmd\_dma(TFT\_MEMORYWRITE, total\_pixels, (uint8\_t\*)(bitmap->data));

}

void tft\_init\_config() {

tft\_cmd(TFT\_SOFTRESET, 0, NULL);

sleep\_ms(150);

tft\_cmd(TFT\_DISPLAYOFF, 0, NULL);

sleep\_ms(150);

tft\_cmd(TFT\_PIXELFORMAT, 1, (uint8\_t[1]){0x55}); //0x55

tft\_cmd(TFT\_POWERCONTROL1, 1, (uint8\_t[1]){0x05}); // 0x05 :3.3V

tft\_cmd(TFT\_POWERCONTROL2, 1, (uint8\_t[1]){0x10});

tft\_cmd(TFT\_VCOMCONTROL1, 2, (uint8\_t[2]){0x3E, 0x28});

tft\_cmd(TFT\_VCOMCONTROL2, 1, (uint8\_t[1]){0x86});

tft\_cmd(TFT\_MADCTL, 1, (uint8\_t[1]){0x08}); //MY,MX,MV,ML,BRG,MH,0,0(40)

tft\_cmd(TFT\_FRAMECONTROL, 2, (uint8\_t[2]){0b00, 0x1B}); // Default 70Hz:0x1B

tft\_cmd(TFT\_DISPLAYFUNC, 4, (uint8\_t[4]){0x0A, 0x82, 0x27, 0x04}); //0a,a2,27,04

tft\_cmd(TFT\_GAMMASET, 1, (uint8\_t[1]){0x01});

//tft\_cmd(tft\_PGAMCOR, 15, 0xFF, (uint8\_t[15]){ 0x0f, 0x31, 0x2b, 0x0c, 0x0e, 0x08, 0x4e, 0xf1, 0x37, 0x07, 0x10, 0x03, 0x0e, 0x09, 0x00 });

//tft\_cmd(tft\_NGAMCOR, 15,0xFF, (uint8\_t[15]){ 0x00, 0x0e, 0x14, 0x03, 0x11, 0x07, 0x31, 0xc1, 0x48, 0x08, 0x0f, 0x0c, 0x31, 0x36, 0x0f });

tft\_cmd(TFT\_SLEEPOUT, 0, NULL);

sleep\_ms(150);

tft\_cmd(TFT\_DISPLAYON, 0, NULL);

sleep\_ms(500);

}

void tft\_init() {

#ifdef PICO\_TFT\_PARALLEL

tft\_pio\_cmd\_init(tft\_pio, tft\_sm, in\_out\_base\_pin, set\_base\_pin, 70000000); //pio freq

#endif

#ifdef PICO\_TFT\_SERIAL

tft\_pio\_cmd\_init(tft\_pio, tft\_sm, s\_in\_out\_base\_pin, sideset\_base\_pin, 90000000); //pio freq

#endif

tft\_init\_config();

}

void tft\_rotate\_image(const uint16\_t \*src, uint16\_t \*dest, uint16\_t width, uint16\_t height, uint16\_t deg) {

double sinx = sin((deg)/180.0 \* M\_PI);

double cosx = cos((deg)/180.0 \* M\_PI);

int xCenter = height/2; // Rotate image by its center.

int yCenter = width/2;

int xt,yt,xRotate,yRotate;

for(int x=0; x<height; x++) {

xt = x - xCenter;

double xt\_cosx = xt\*cosx;

double xt\_sinx = xt\*sinx;

for(int y=0; y<width; y++) {

yt = y - yCenter;

xRotate = (int)(lround(xt\_cosx - (yt\*sinx)) + xCenter);

yRotate = lround((yt\*cosx) + xt\_sinx) + yCenter;

if( (xRotate >= 0) && (xRotate < height) && (yRotate >= 0) && (yRotate < width) ) {

dest[x\*width+y] = src[xRotate\*width+yRotate];

} else {

dest[x\*width+y]=0xffff;

}

}

}

}

pico\_tft.h

#ifndef \_TFT\_H\_

#define \_TFT\_H\_

#define TFT\_WIDTH 128

#define TFT\_HEIGHT 160

#include "pico/stdlib.h"

#include "fonts/bitmap\_typedefs.h"

#include "hardware/pio.h"

#include "math.h"

void tft\_init();

void tft\_init\_config();

void tft\_draw\_pixel(uint16\_t x, uint16\_t y, uint16\_t color);

void tft\_set\_address\_window(uint16\_t x1, uint16\_t y1, uint16\_t x2, uint16\_t y2);

void tft\_cmd(uint32\_t cmd, uint32\_t count, uint8\_t \*param);

void tft\_cmd\_dma(uint32\_t cmd, uint32\_t count, uint8\_t \*param);

uint16\_t tft\_color\_565RGB(uint8\_t R, uint8\_t G, uint8\_t B);

void tft\_fill\_rect(uint16\_t x, uint16\_t y, uint16\_t width, uint16\_t height, uint16\_t color);

void tft\_draw\_bitmap(uint16\_t x, uint16\_t y, const tImage \*bitmap);

void tft\_pio\_cmd\_init(PIO pio, uint sm, uint out\_base, uint set\_base, uint32\_t freq);

void tft\_rotate\_image(const uint16\_t \*src, uint16\_t \*dest, uint16\_t width, uint16\_t height, uint16\_t deg);

void tft\_fill\_partial\_image(uint8\_t \*src, uint8\_t \*dest,uint16\_t x, uint16\_t y, uint32\_t ow, uint32\_t oh,uint16\_t width, uint16\_t height);

void tft\_draw\_frame(uint8\_t \*src, uint16\_t x, uint16\_t y, uint16\_t tx, uint16\_t ty, uint32\_t ow, uint32\_t oh);

void tft\_draw\_image\_with\_trasnparent\_color(uint16\_t x, uint16\_t y, const tImage \*bitmap, uint32\_t transparent\_color);

void fb\_draw\_image(uint8\_t\* frame\_buffer, uint8\_t \*src, uint16\_t x, uint16\_t y, uint16\_t fx, uint16\_t fy, uint32\_t ow, uint32\_t oh, bool wrapx, bool wrapy, bool enable\_transparent, uint16\_t transparent\_color);

#endif

測試主程式

#include <stdio.h>

#include "pico/stdlib.h"

#include "hardware/dma.h"

#include "hardware/pio.h"

#include "pico\_tft.h"

#include "tft\_string.h"

#include "fonts/font\_ubuntu\_mono\_24.h"

#include "registers.h"

#include "time.h"

#include "stdlib.h"

#include "string.h"

#include "fonts/img1.h"

#include "fonts/img2.h"

#include "fonts/gear.h"

#include "pico/multicore.h"

#include "fonts/bg.h"

#include "fonts/w1.h"

#include "fonts/w2.h"

#include "fonts/w3.h"

typedef struct \_rotate\_obj\_t {

uint8\_t \*srcimg;

uint8\_t \*destimg;

uint16\_t width;

uint16\_t height;

uint16\_t angle;

} rotate\_obj\_t;

void core1\_rotate() {

while(1) {

rotate\_obj\_t \*rimg = (rotate\_obj\_t\*)multicore\_fifo\_pop\_blocking();

tft\_rotate\_image((uint16\_t\*)rimg->srcimg,(uint16\_t\*)rimg->destimg, rimg->width, rimg->height, rimg->angle);

multicore\_fifo\_push\_blocking(1);

}

}

int main()

{

stdio\_init\_all();

tft\_init();

multicore\_launch\_core1(core1\_rotate);

rotate\_obj\_t \*rotate\_img = (rotate\_obj\_t\*) calloc(1, sizeof(rotate\_obj\_t));

uint8\_t \*read\_buff = (uint8\_t\*) calloc(TFT\_WIDTH\*TFT\_HEIGHT\*2, sizeof(uint8\_t)); //frame buffer

if (!read\_buff) {

printf("alloc memory error\n");

return 0;

}

uint8\_t \*write\_buff = (uint8\_t\*) calloc(TFT\_WIDTH\*TFT\_HEIGHT\*2, sizeof(uint8\_t)); //frame buffer

if (!write\_buff) {

printf("alloc memory error\n");

return 0;

}

uint8\_t \*tmp\_buff;

absolute\_time\_t t1, t2;

/\* start testing \*/

// single frame =============================

tft\_fill\_rect(0,0, TFT\_WIDTH, TFT\_HEIGHT, 0xffff);

tft\_draw\_string(20,20,"Starting", 0x001f, &font\_ubuntu\_mono\_24);

printf("\n=== Start testing ===\n");

sleep\_ms(1000);

printf(" single frame...\n");

t1 = get\_absolute\_time();

tft\_set\_address\_window (0, 0, TFT\_WIDTH-1, TFT\_HEIGHT-1);

tft\_cmd(TFT\_MEMORYWRITE, TFT\_WIDTH\*TFT\_HEIGHT\*2, (uint8\_t\*)img1.data);

t2=get\_absolute\_time();

printf("single frame: image1 store in XIP flash (%f fps) --time:%"PRIu64"us\n",(1000000.0)/absolute\_time\_diff\_us(t1, t2),absolute\_time\_diff\_us(t1, t2));

sleep\_ms(1000);

t1 = get\_absolute\_time();

tft\_set\_address\_window (0, 0, TFT\_WIDTH-1, TFT\_HEIGHT-1);

tft\_cmd(TFT\_MEMORYWRITE, TFT\_WIDTH\*TFT\_HEIGHT\*2, (uint8\_t\*)img2.data);

t2=get\_absolute\_time();

printf("single frame: image2 store in XIP flash (%f fps) --time:%"PRIu64"us\n",(1000000.0)/absolute\_time\_diff\_us(t1, t2),absolute\_time\_diff\_us(t1, t2));

sleep\_ms(1000);

// Animation shift background ========

printf("\n Animation shift background...\n");

t1 = get\_absolute\_time();

for (int i =0; i < bg.width; i+=2) {

fb\_draw\_image(read\_buff, (uint8\_t\*)bg.data, i,60, 0,0,bg.width, bg.height, false, false, false, 0x0000);

tft\_set\_address\_window (0, 0, TFT\_WIDTH-1, TFT\_HEIGHT-1);

tft\_cmd(TFT\_MEMORYWRITE, TFT\_WIDTH\*TFT\_HEIGHT\*2, read\_buff);

}

t2=get\_absolute\_time();

printf("shift %u frames (%f fps) --time:%"PRIu64"us\n", bg.width/2, (float)bg.width/2\*1000000/absolute\_time\_diff\_us(t1, t2),absolute\_time\_diff\_us(t1, t2));

// animation shift background and man walk ========

printf("\n Animation shift background and man walk...\n");

//for (int j=0; j < bg.height-TFT\_HEIGHT; j+=60){

t1= get\_absolute\_time();

for (int i =0; i < bg.width; i++) {

fb\_draw\_image(read\_buff, (uint8\_t\*)bg.data, i,0, 0,0,bg.width, bg.height, false, false, false, 0x0000);

if (i%18< 6) fb\_draw\_image(read\_buff, (uint8\_t\*)w1.data, 0,0,30,30, w1.width, w1.height, false, false, true, 0x0000);

if (i%18>=6 && i%18 <12) fb\_draw\_image(read\_buff, (uint8\_t\*)w2.data, 0,0,30,30, w2.width, w2.height, false, false, true, 0x0000);

if (i%18>=12) fb\_draw\_image(read\_buff, (uint8\_t\*)w3.data, 0,0,30,30, w3.width, w3.height, false, false, true, 0x0000);

tft\_set\_address\_window (0, 0, TFT\_WIDTH-1, TFT\_HEIGHT-1);

tft\_cmd(TFT\_MEMORYWRITE, TFT\_WIDTH\*TFT\_HEIGHT\*2, read\_buff);

}

t2=get\_absolute\_time();

printf("shift %u frames (%f fps) --time:%"PRIu64"us\n", bg.width, (float)bg.width\*1000000/absolute\_time\_diff\_us(t1, t2),absolute\_time\_diff\_us(t1, t2));

t1= get\_absolute\_time();

for (int i =0; i < bg.width; i++) {

fb\_draw\_image(read\_buff, (uint8\_t\*)bg.data, i,i, 0,0,bg.width, bg.height, true, true, false, 0x0000);

if (i%18< 6) fb\_draw\_image(read\_buff, (uint8\_t\*)w1.data, 0,0,30,30, w1.width, w1.height, false, false, true, 0x0000);

if (i%18>=6 && i%18 <12) fb\_draw\_image(read\_buff, (uint8\_t\*)w2.data, 0,0,30,30, w2.width, w2.height, false, false, true, 0x0000);

if (i%18>=12) fb\_draw\_image(read\_buff, (uint8\_t\*)w3.data, 0,0,30,30, w3.width, w3.height, false, false, true, 0x0000);

tft\_set\_address\_window (0, 0, TFT\_WIDTH-1, TFT\_HEIGHT-1);

tft\_cmd(TFT\_MEMORYWRITE, TFT\_WIDTH\*TFT\_HEIGHT\*2, read\_buff);

}

t2=get\_absolute\_time();

printf("shift %u frames (%f fps) --time:%"PRIu64"us\n", bg.width, (float)bg.width\*1000000/absolute\_time\_diff\_us(t1, t2),absolute\_time\_diff\_us(t1, t2));

//}

// random 50 rectangles in a frame================

printf("\n Random 50 rectangles in a frame\n");

time\_t t;

srand((unsigned) time(&t));

char src\_buf[100\*100\*2];

uint8\_t \*spt;

int x,y,width,color;

int boxs[100][4]; //(x,y,width, color-h, cololr-l)

//==== 100 random rectangles

for (int i=0; i < 100; i++) {

y=rand()%TFT\_HEIGHT;

x=rand()%TFT\_WIDTH;

width=rand()%40;

if (y+width >=TFT\_HEIGHT) y=TFT\_HEIGHT-y;

if (x+width >=TFT\_WIDTH) x=TFT\_WIDTH-x;

boxs[i][0]=x;

boxs[i][1]=y;

boxs[i][2]=width;

boxs[i][3]=rand()%0xff;

}

int p;

int i, j, b; // repeat test times

for (int rep=0; rep < 2; rep++) {

t1= get\_absolute\_time();

for (i=0; i < 100; i++) {

memset(write\_buff, 0, TFT\_WIDTH\*TFT\_HEIGHT\*2);

for (j=0; j < 50; j++) {

p = rand()%100;

spt= (write\_buff+(boxs[p][0]+boxs[p][1]\*TFT\_WIDTH)\*2);

for (int k =0; k < boxs[p][2]; k++){

memset((spt+k\*TFT\_WIDTH\*2), boxs[p][3], boxs[p][2]\*2);

}

}

b=j;

tft\_set\_address\_window (0, 0, TFT\_WIDTH-1, TFT\_HEIGHT-1);

tft\_cmd(TFT\_MEMORYWRITE, TFT\_WIDTH\*TFT\_HEIGHT\*2, write\_buff);

}

t2=get\_absolute\_time();

printf("%d frames(%d Boxs) in RAM) --time:%"PRIu64"us\n",i, j, absolute\_time\_diff\_us(t1, t2));

printf("Frame Rate: %ffps, pause 1 second to start next test\n\n", (float)i/(absolute\_time\_diff\_us(t1, t2))\*1000000);

sleep\_ms(1000);

}

// coror frame test

printf("\n Color frame test...\n");

t1 = get\_absolute\_time();

tft\_fill\_rect(0,0, TFT\_WIDTH, TFT\_HEIGHT, 0xf800);

tft\_draw\_string(20,20,"Red", 0xffff,&font\_ubuntu\_mono\_24);

t2=get\_absolute\_time();

printf("1 color frame(with text) --time:%"PRIu64"us\n",absolute\_time\_diff\_us(t1, t2));

printf("Frame Rate: %ffps\n\n", 1000000.0/(absolute\_time\_diff\_us(t1, t2)));

sleep\_ms(1000);

t1 = get\_absolute\_time();

tft\_set\_address\_window (0, 0, TFT\_WIDTH-1, TFT\_HEIGHT-1);

tft\_cmd(TFT\_MEMORYWRITE, TFT\_WIDTH\*TFT\_HEIGHT\*2, (uint8\_t\*)image\_data\_img1);

t2=get\_absolute\_time();

printf("1 image frame --time:%"PRIu64"us\n",absolute\_time\_diff\_us(t1, t2));

printf("Frame Rate: %ffps\n\n", 1000000.0/(absolute\_time\_diff\_us(t1, t2)));

sleep\_ms(1000);

t1 = get\_absolute\_time();

tft\_fill\_rect(0,0, TFT\_WIDTH, TFT\_HEIGHT, 0x7E0);

tft\_draw\_string(20,20,"Green", 0xf800, &font\_ubuntu\_mono\_24);

t2=get\_absolute\_time();

printf("1 color frame(with text) --time:%"PRIu64"us\n",absolute\_time\_diff\_us(t1, t2));

printf("Frame Rate: %ffps\n\n", 1000000.0/(absolute\_time\_diff\_us(t1, t2)));sleep\_ms(1000);

t1 = get\_absolute\_time();

tft\_set\_address\_window (0, 0, TFT\_WIDTH-1, TFT\_HEIGHT-1);

tft\_cmd(TFT\_MEMORYWRITE, TFT\_WIDTH\*TFT\_HEIGHT\*2, (uint8\_t\*)image\_data\_img2);

t2=get\_absolute\_time();

printf("1 image frame --time:%"PRIu64"us\n",absolute\_time\_diff\_us(t1, t2));

printf("Frame Rate: %ffps\n\n", 1000000.0/(absolute\_time\_diff\_us(t1, t2)));

sleep\_ms(1000);

t1 = get\_absolute\_time();

tft\_fill\_rect(0,0, TFT\_WIDTH, TFT\_HEIGHT, 0x001f);

tft\_draw\_string(20,20,"Blue", 0xffff, &font\_ubuntu\_mono\_24);

t2=get\_absolute\_time();

printf("1 color frame(with text) --time:%"PRIu64"us\n",absolute\_time\_diff\_us(t1, t2));

printf("Frame Rate: %ffps\n\n", 1000000.0/(absolute\_time\_diff\_us(t1, t2)));

sleep\_ms(1000);

tft\_fill\_rect(0,0, TFT\_WIDTH, TFT\_HEIGHT, 0xffff);

// image rotation test

printf("\n Image rotation test...\n");

t1 = get\_absolute\_time();

for (int i =0; i < 360; i+=2) {

tft\_rotate\_image((uint16\_t\*)image\_data\_gear, (uint16\_t\*)write\_buff, 100, 100, i);

tft\_set\_address\_window (13, 29, 112, 128);

tft\_cmd(TFT\_MEMORYWRITE, 100\*100\*2, (uint8\_t\*)write\_buff);

}

t2=get\_absolute\_time();

printf("rotation 360 degrees step 2 degree(100\*100 image in flash) --time:%"PRIu64"us\n",absolute\_time\_diff\_us(t1, t2));

printf("Frame Rate: %ffps\n\n", 180.0/(absolute\_time\_diff\_us(t1, t2))\*1000000);

memcpy(src\_buf, image\_data\_gear, 100\*100\*2);

t1 = get\_absolute\_time();

for (int i =360; i>0; i-=2) {

tft\_rotate\_image((uint16\_t\*)src\_buf, (uint16\_t\*)write\_buff, 100, 100, i);

tft\_set\_address\_window (13, 29, 112, 128);

tft\_cmd(TFT\_MEMORYWRITE, 100\*100\*2, (uint8\_t\*)write\_buff);

}

t2=get\_absolute\_time();

printf("rotation 360 degree step 2 degree(100\*100 source image in RAM)--time:%"PRIu64"us\n",absolute\_time\_diff\_us(t1, t2));

printf("Frame Rate: %ffps\n\n", 180.0/(absolute\_time\_diff\_us(t1, t2))\*1000000);

/\* core1 rotate\*/

rotate\_img->srcimg = (uint8\_t\*)image\_data\_gear;

rotate\_img->width=100;

rotate\_img->height=100;

memcpy(write\_buff, image\_data\_gear, 100\*100\*2);

t1 = get\_absolute\_time();

for (int i =0; i < 360; i+=2) {

rotate\_img->angle=i;

rotate\_img->destimg = read\_buff;

multicore\_fifo\_push\_blocking((uint32\_t)rotate\_img);

tft\_set\_address\_window (13, 29, 112, 128);

tft\_cmd(TFT\_MEMORYWRITE, 100\*100\*2, (uint8\_t\*)write\_buff);

uint32\_t r = multicore\_fifo\_pop\_blocking();

tmp\_buff = read\_buff;

read\_buff= write\_buff;

write\_buff=tmp\_buff;

}

t2=get\_absolute\_time();

printf("rotation 360 degree step 2 degree(100\*100) multicore --time:%"PRIu64"us\n",absolute\_time\_diff\_us(t1, t2));

printf("Frame Rate: %ffps\n\n", 180.0/(absolute\_time\_diff\_us(t1, t2))\*1000000);

memcpy(src\_buf, image\_data\_gear, 100\*100\*2);

tft\_rotate\_image((uint16\_t\*)src\_buf, (uint16\_t\*)write\_buff, 100, 100, 10);

t1 = get\_absolute\_time();

for (int i =0; i < 360; i++) {

tft\_set\_address\_window (13, 29, 112, 128);

tft\_cmd(TFT\_MEMORYWRITE, 100\*100\*2, (uint8\_t\*)write\_buff);

//tft\_set\_address\_window (13, 29, 112, 128);

tft\_cmd(TFT\_MEMORYWRITE, 100\*100\*2, (uint8\_t\*)src\_buf);

}

t2=get\_absolute\_time();

float factor = (100.0\*100.0)/(128\*160);

printf("rotation forward and backward 10 degree(double buffers in RAM)--time:%"PRIu64"us\n",absolute\_time\_diff\_us(t1, t2));

printf("Frame Rate: %ffps\n\n", 360.0\*2\*factor/(absolute\_time\_diff\_us(t1, t2))\*1000000);

printf("The End\n");

tft\_fill\_rect(0,0, TFT\_WIDTH, TFT\_HEIGHT, 0xffff);

tft\_draw\_string(20,20,"The End", 0x001f, &font\_ubuntu\_mono\_24);

while(1) {

tight\_loop\_contents();

}

puts("Hello, world!");

return 0;

}