4.

Parameters required when calling drawPixel

- Position X

- Position Y

- Colour

Values of X and Y formulas:

position Y = Y \* SCREEN\_X \* BITS\_PER\_PIXEL / BITS PER BYTE

position X = X \* BITS\_PER\_PIXEL / BITS PER BYTE

5.

Code which controls the calling of drawPixel:

bl drawpixel

What that code is doing:

It adds values to r1, and r2, and takes the value of color (r6) to add to r3. And then it

runs itself.

9.

;r0 = pointer + x \* BITS\_PER\_PIXEL/8 + y \* SCREEN\_X \* BITS\_PER\_PIXEL/8

format binary as 'img'

;constants

;memory addresses of BASE

BASE = $FE000000 ; $3F000000 for 2B, 3B, 3B+

org $8000

mov sp,$1000

;set up GPIOs

GPIO\_OFFSET = $200000

mov r10,BASE

orr r10,GPIO\_OFFSET ;Base address of GPIO

ldr r8,[r10,#4] ;read function register for GPIO 10 - 19

bic r8,r8,#27 ;bit clear 27 = 9 \* 3 = read access

str r8,[r10,#4];10 input

;set up input

mov r8,#1

lsl r8,#10 ;bit 10 to enable input GPIO10

mov r0,BASE

bl FB\_Init

;r0 now contains address of screen

;SCREEN\_X and BITS\_PER\_PIXEL are global constants in FB\_Init

and r0,$3FFFFFFF ; Convert Mail Box Frame Buffer Pointer From BUS Address To Physical Address ($CXXXXXXX -> $3XXXXXXX)

str r0,[FB\_POINTER] ; Store Frame Buffer Pointer Physical Address

mov r7,r0 ;back-up a copy of the screen address

; Setup Characters

CHAR\_X = 8

CHAR\_Y = 8

loop$:

;read first block of GPIOs

ldr r9,[r10,#52] ;read gpios 0-31

tst r9,#1024 ; use tst to check bit 10

bne red ;if ==0

bl setup\_chars

adr r2,Text ; R2 = Text Offset "Open"

DrawChars:

mov r4,CHAR\_Y ; R4 = Character Row Counter

ldrb r5,[r2],1 ; R5 = Next Text Character

add r5,r1,r5,lsl 6 ; Add Shift To Correct Position In Font (\* 64)

bl DrawChar

subs r3,1 ; Subtract Number Of Text Characters To Print

subne r0,SCREEN\_X \* CHAR\_Y ; Jump To Top Of Char

addne r0,CHAR\_X ; Jump Forward 1 Char

bne DrawChars ; IF (Number Of Text Characters != 0) Continue To Print Characters

b cont

red:

bl setup\_chars

adr r2,Text2 ; R2 = Text Offset "Closed"

DrawChars2:

mov r4,CHAR\_Y ; R4 = Character Row Counter

ldrb r5,[r2],1 ; R5 = Next Text Character

add r5,r1,r5,lsl 6 ; Add Shift To Correct Position In Font (\* 64)

bl DrawChar

subs r3,1 ; Subtract Number Of Text Characters To Print

subne r0,SCREEN\_X \* CHAR\_Y ; Jump To Top Of Char

addne r0,CHAR\_X ; Jump Forward 1 Char

bne DrawChars2 ; IF (Number Of Text Characters != 0) Continue To Print Characters

cont:

;call timer (stop keybounce)

;push {r0-r11}

;mov r0,BASE

;mov r1,$0A100

;orr r1,$00020 ;TIMER\_MICROSECONDS = 40,000

;bl TIMER

;pop {r0-r11}

b loop$

setup\_chars:

; Setup Characters

mov r0,r7

mov r1,SCREEN\_X

lsl r1,r1,5 ;32

orr r1,#256

add r0,r1 ; Place Text At XY Position 256,32

adr r1,Font ; R1 = Characters

mov r3,#6 ; R3 = Number Of Text Characters To Print

bx lr

include "FBinit8.asm"

include "timer2\_2Param.asm"

include "DrawChar.asm"

align 4

Text:

db " Open!"

align 4

Text2:

db "Closed"

align 4

Font:

include "Font8x8.asm"