Code flowchart

Understand the flow of code within your system using our code flowchart template. Identify start/stop points, conditions, inputs/outputs and decisions to showcase how a system connects.

How to use this template

1. Choose your stop and start points.
2. Insert any inputs or outputs and processing points.
3. Insert any decisions.
4. Connect the system with true/false flows.

Quick tips

Start with the toolbar
It's at the bottom of your screen, with stickies, stamps, and anything you need.

Move and zoom
Use the hand tool to pan around. Zoom controls are in the top right corner.

But first: get your team in here
FigJam's better with others. Click the **Share** button above to invite your team.

Explore more templates

FigJam Templates | 100+ Curated Resources
Add superpowers to your online remote meeting w...
figma.com

Start_Level.asm

DrawMap

x = 70
y = 20

```
lda #0
sta MAP_X_DELTA
sta MAP_Y_DELTA
stx MAP_X_POS
sty MAP_Y_POS
```

ZEROPAGE_POINTER_4 = MAP File

```
ldx MAP_Y_POS
lda MAP_LINE_LOOKUP_LO,x
sta ZEROPAGE_POINTER_4
lda MAP_LINE_LOOKUP_HI,x
sta ZEROPAGE_POINTER_4 + 1
```

; Draw line by line
inc ZEROPAGE_POINTER_4

```
clc
lda ZEROPAGE_POINTER_4
adc MAP_X_POS
sta ZEROPAGE_POINTER_4
lda ZEROPAGE_POINTER_4+ 1
adc #0
sta ZEROPAGE_POINTER_4 + 1
```

Map Data (ZP_4) copied to MAP_POS_ADDRESS

```
copyPointer
ZEROPAGE_POINTER_4, MAP_POS_ADDRESS
```

Draw map to screen:
lda (MAP_LINE_LOOKUP),y

```
ldy #0
ldx #0
```

@loop

```
lda (ZEROPAGE_POINTER_4),y
jsr DrawTile
iny
```

Draw 10 tiles across (horizontally)

```
lda
(ZEROPAGE_POINTER_4),y
cpy #10
bne @loop
```

Move down one line after counting over 100 tiles in the map file

```
addPointer
ZEROPAGE_POINTER_4, 100
ldy #0
inx
```

Go down 6 tiles

```
cpx #6
bne @loop
```

RTS

DrawTile

PARAM1 = Tile Number
PARAM2 = X position
PARAM3 = Y position

```
sta PARAM1
sty PARAM2
stx PARAM3
saveRegs
```

Calculate 4 × 4 tiles using PARAM3 (Y position)

```
lda PARAM3
asl
asl
tax
```

Read correct screen address from table lookup

```
jsr GetScreenLineAddress
lda COLOR_LINE_OFFSET_TABLE_LO,x
sta ZEROPAGE_POINTER_3
lda COLOR_LINE_OFFSET_TABLE_HI,x
sta ZEROPAGE_POINTER_3 + 1
```

Calculate 4 × 4 tiles using PARAM2 (X position)

```
lda PARAM2
asl
asl
tax
clc
```

Screen address now exists in ZEROPAGE_POINTER_1 when we executed "GetScreenLineAddress"

```
adc ZEROPAGE_POINTER_1
sta ZEROPAGE_POINTER_1
lda ZEROPAGE_POINTER_1 + 1
adc #0
sta ZEROPAGE_POINTER_1 + 1
```

```
txa
clc
```

Color ram exists in ZEROPAGE_POINTER_3

```
adc ZEROPAGE_POINTER_3
sta ZEROPAGE_POINTER_3
lda ZEROPAGE_POINTER_3+ 1
adc #0
sta ZEROPAGE_POINTER_3 + 1
```

Tile data exists in ZEROPAGE_POINTER_2

```
ldx PARAM1
lda TILE_NUMBER_LOOKUP_LO,x
sta ZEROPAGE_POINTER_2
lda TILE_NUMBER_LOOKUP_HI,x
sta ZEROPAGE_POINTER_2 + 1
```

```
ldy #0
```

@drawloop

Color attribute data is read into ZEROPAGE_POINTER_3

```
lda (ZEROPAGE_POINTER_2),y
sta (ZEROPAGE_POINTER_1),y
tax
lda ATTRIBUTE_MEM,x
sta (ZEROPAGE_POINTER_3),y
```

Keep drawing until all attributes are completed

```
cpy #15
beq @DONE
tya
iny
```

Set count 0-3 to draw a row of tiles (use the and mask for this)

```
and #%00000011
cmp #15
bne @drawloop
```

Draw map tiles to the screen going down 40 characters

```
clc
lda ZEROPAGE_POINTER_1
adc #40 - 4
sta ZEROPAGE_POINTER_1
lda ZEROPAGE_POINTER_1 + 1
adc #0
sta ZEROPAGE_POINTER_1 + 1
```

Draw color ram to the screen going down 40 characters

```
clc
lda ZEROPAGE_POINTER_3
adc #40 - 4
sta ZEROPAGE_POINTER_3
lda ZEROPAGE_POINTER_3+ 1
adc #0
sta ZEROPAGE_POINTER_3 + 1
jmp @drawloop
```

@done

```
restoreRegs
rts
```