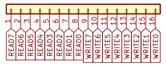
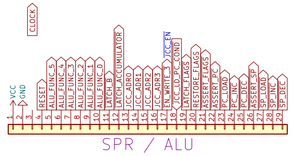
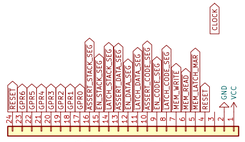
Control Unit – Microcode Instruction Decoding

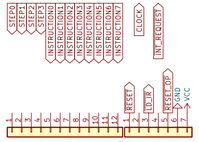
Microcode Control Words

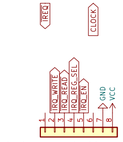
From EEPROMs







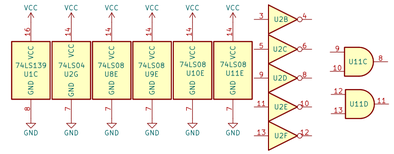




Write: (INC_PC, RESET OP, RESET) | Group | Module

Read: () | Group | Module

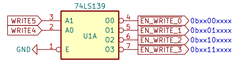
I should get HALT in here somewhere.
I think perhaps it should just be part of the Instruction Register Module with a control line here. But need to figure out how to "un halt"



Write Enable Decoder

0bxx11xxxx

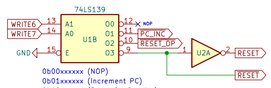
Write Enable Decoder is in charge of enabling other decoders which use WRITE[0..3] – They are all the control lines that perform any "write" operation



Special Write Decoder

0b11xxxxxx

Special Write Decoder is able to add an additional active control line at the same time as another Write operation. For example, this allows the CPU to increment the PC at the same time as latching the instruction register.

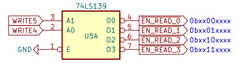


0b00xxxxxx (NOP)
0b01xxxxxx (Increment PC)
0b10xxxxxx (Reset Op Counter)
0b11xxxxxx (Reset CPU)

Read Decoder

0bxx11xxxx

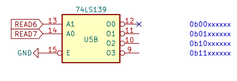
Read Enable Decoder is in charge of enabling other decoders which use READ[0..3] – They are all the control lines that perform any "read" operation



Read Decoder

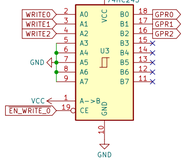
0b11xxxxxx

Special Read Decoder is able to add an additional active control line at the same time as another Read operation. For example, this allows the CPU to ....[todo]...



Writes: 0bxx_00_xxxx

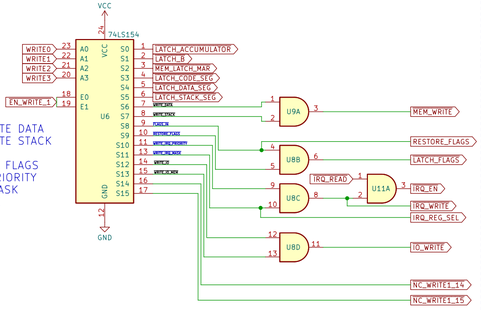
General Purpose Registers



0bxx00x000: (0x0 | 0x0) 0x0 NOP
0bxx00x001: (0x0 | 0x1) 0x1 LDC
0bxx00x010: (0x0 | 0x2) 0x2 LDD
0bxx00x011: (0x0 | 0x3) 0x3 LDS1
0bxx00x100: (0x0 | 0x4) 0x4 LDS2
0bxx00x101: (0x0 | 0x5) 0x5 LDF
0bxx00x110: (0x0 | 0x6) 0x6 LDS3
0bxx00x111: (0x0 | 0x7) 0x7 NOP

Writes: 0bxx_01_xxxx

Accumulator, Operand, Memory, Interrupts, I/O

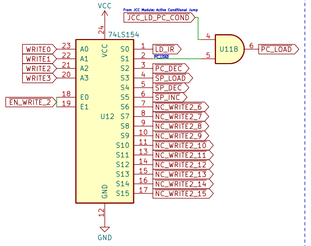


0bxx010000: (0x10 | 0x0) 0x10 LDA
0bxx010001: (0x10 | 0x1) 0x11 LDB
0bxx010010: (0x10 | 0x2) 0x12 LDMAR
0bxx010011: (0x10 | 0x3) 0x13 LDCS
0bxx010100: (0x10 | 0x4) 0x14 LDDS
0bxx010101: (0x10 | 0x5) 0x15 LDSS
0bxx010110: (0x10 | 0x6) 0x16 MEM WRITE DATA
0bxx010111: (0x10 | 0x7) 0x17 MEM WRITE STACK
0bxx011000: (0x10 | 0x8) 0x18 LDFLAGS
0bxx011001: (0x10 | 0x9) 0x19 RESTORE FLAGS
0bxx011010: (0x10 | 0xA) 0x1A LDIRQ PRIORITY
0bxx011011: (0x10 | 0xB) 0x1B LDIRQ MASK
0bxx011100: (0x10 | 0xC) 0x1C IO WRITE
0bxx011101: (0x10 | 0xD) 0x1D n.c.
0bxx011110: (0x10 | 0xE) 0x1E n.c.
0bxx011111: (0x10 | 0xF) 0x1F n.c.

Writes: 0bxx_10_xxxx

Instruction Register, Stack Pointer, Program Counter

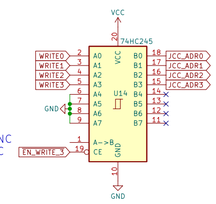
WHY DO I NEED PC_LOAD HERE?
WHY NOT JUST USE 0b0000 with the JCC Module?



0bxx100000: (0x20 | 0x0) 0x20 LDIR
0bxx100001: (0x20 | 0x1) 0x21 LDPC
0bxx100010: (0x20 | 0x2) 0x22 DEC PC
0bxx100011: (0x20 | 0x3) 0x23 LDSP
0bxx100100: (0x20 | 0x4) 0x24 DEC SP
0bxx100101: (0x20 | 0x5) 0x25 INC SP
0bxx100110: (0x20 | 0x6) 0x26 n.c.
0bxx100111: (0x20 | 0x7) 0x27 n.c.
0bxx101000: (0x20 | 0x8) 0x28 n.c.
0bxx101001: (0x20 | 0x9) 0x29 n.c.
0bxx101010: (0x20 | 0xA) 0x2A n.c.
0bxx101011: (0x20 | 0xB) 0x2B n.c.
0bxx101100: (0x20 | 0xC) 0x2C n.c.
0bxx101101: (0x20 | 0xD) 0x2D n.c.
0bxx101110: (0x20 | 0xE) 0x2E n.c.
0bxx101111: (0x20 | 0xF) 0x2F n.c.

Writes: 0bxx_11_xxxx

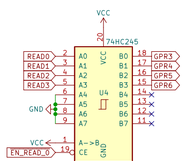
Conditional Jumps



0bxx110000: (0x30 | 0x0) 0x30 Jump
0bxx110001: (0x30 | 0x1) 0x31 JLE, JNG
0bxx110010: (0x30 | 0x2) 0x32 JG, JNLE
0bxx110011: (0x30 | 0x3) 0x33 JGE, JNL
0bxx110100: (0x30 | 0x4) 0x34 JL, JNGE
0bxx110101: (0x30 | 0x5) 0x35 JA, JNBE
0bxx110110: (0x30 | 0x6) 0x36 JBE, JNA
0bxx110111: (0x30 | 0x7) 0x37 JNB, JAE, JNC
0bxx111000: (0x30 | 0x8) 0x38 JB, JNAE, JC
0bxx111001: (0x30 | 0x9) 0x39 JNE, JNZ
0bxx111010: (0x30 | 0xA) 0x3A JE, JZ
0bxx111011: (0x30 | 0xB) 0x3B JNS
0bxx111100: (0x30 | 0xC) 0x3C JS
0bxx111101: (0x30 | 0xD) 0x3D JNO
0bxx111110: (0x30 | 0xE) 0x3E JO
0bxx111111: (0x30 | 0xF) 0x3F NOP

Reads: 0bxx_00_xxxx

General Purpose Registers

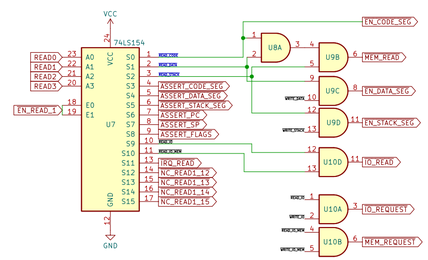


0bxx000000: (0x0 | 0x0) 0x0 NOP
0bxx000001: (0x0 | 0x1) 0x1
0bxx000010: (0x0 | 0x2) 0x2
0bxx000011: (0x0 | 0x3) 0x3
0bxx000100: (0x0 | 0x4) 0x4
0bxx000101: (0x0 | 0x5) 0x5
0bxx000110: (0x0 | 0x6) 0x6
0bxx000111: (0x0 | 0x7) 0x7

0bxx000111: (0x0 | 0x8) 0x8
0bxx000111: (0x0 | 0x9) 0x9
0bxx000111: (0x0 | 0xA) 0xA
0bxx000111: (0x0 | 0xB) 0xB
0bxx000111: (0x0 | 0xC) 0xC
0bxx000111: (0x0 | 0xD) 0xD
0bxx000111: (0x0 | 0xE) 0xE
0bxx000111: (0x0 | 0xF) 0xF

Writes: 0bxx_01_xxxx

Special Purpose Registers & Memory



Writes: 0bxx_11_xxxx

Arithmetic Logic Unit