Built In Self-Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Out High | Out Low | Failed address | Reason |
| Test 1: Shift Left (8: ASL) | B4 | 01 | 0x0C | Result is not D0 |
|  | 68 |  |  |  |
|  | D0 |  |  |  |
| Test 2: Rotate Left (9: ROL) | 68 | 02 | 0x1A | Carry is not set |
|  | D0 |  | 0x1F | Result is not 0x41 |
|  | A0 |  |  |  |
|  | 41 |  |  |  |
| Test 3: Logic SR (A: LSR) | 43 | 03 | 0x29 | Carry is not set |
|  | 21 |  | 0x2D | Result is not 0x21 |
| Test 4: Arithmetic SR (B: ASR) | C3 | 04 | 0x37 | Carry is not set |
|  | E1 |  | 0x3B | Result is not 0xE1 |
| Test 5: Add (1234 + 6DCD) = 0x8001 | 01 | 05 | 0x43 | Carry is not set |
|  | 80 |  | 0x49 | Carry is set |
|  |  |  | 0x4D | Result is not 0x80 |
| Test 6: Sub (A765 – 34AB) = 0x72BA | BA | 06 | 0x55 | Carry is set |
|  | 72 |  | 0x5B | Carry is not set |
|  |  |  | 0x5F | Result is not 0x72 |
| Test 7: Rotate Right (C: ROR) | 1B | 07 | 0x70 | Carry is not set |
|  | 0D |  | 0x74 | Result is not 0x61 |
|  | 86 |  |  |  |
|  | C3 |  |  |  |
|  | 61 |  |  |  |
| Test 8: SRAM Imediate Load Store  LMA / SMA | 33 | 08 |  |  |
|  | 22 |  |  |  |
|  | 00 |  | 0x86 | Result was not 00 |
| Test 9: SRAM Register Load/Store  LMR / SMR | 5,4,3,2,1 | 09 | 0x9A | Last Read was not 5 |
|  | 24,1 |  |  |  |
|  | 23,2 |  |  |  |
|  | 22,3 |  |  |  |
|  | 21,4 |  |  |  |
|  | 20,5 |  |  |  |
| END |  | 0A | 0x9D | PASS |

{ // Display | High | Low

0x03, 0x01, // 0: LDR R0, 0x01 --------|------|------

0x18, 0x00, // 1: OUT R0->L (0x01)

// Test 1: Shift Left (8: ASL)

0x83, 0x5A, // 2: LDR 02, 0x5A

0x82, 0x28, // 3: ASL R2, R2

0x28, 0x20, // 4: OUT Reg 2 (0xB4)

0x82, 0x28, // 5: ASL R2, R2

0x28, 0x20, // 6: OUT Reg 2 (0x68) + Carry

0x82, 0x28, // 7: ASL R2, R2 (Unlike ROL, ASL does not take the carry in back)

0x28, 0x20, // 8: OUT Reg 2 (0xD0)

0xC3, 0xD0, // 9: LDR 03, D0

0xC2, 0xB7, // A: SUB R3, R3, R2

0x67, 0x01, // B: BEQ RELATIVE JUMP

0x01, 0x00, // C: BRK, Test 1 failed

// Test 2: Rotate Left (9: ROL)

0x00, 0x00, // D: NOP

0x06, 0x16, // E: INC R0

0x18, 0x00, // F: OUT R0->L (0x02)

0x83, 0x34, // 10: LDR 02, 0x34

0x82, 0x29, // 11: ROL R2, R2 // 34 -> 68

0x28, 0x20, // 12: OUT Reg 2 (0x68)

0x82, 0x29, // 13: ROL R2, R2 // 68 -> D0

0x28, 0x20, // 14: OUT Reg 2 (0xD0)

0x82, 0x29, // 15: ROL R2, R2 // D0 -> A0 + C

0x28, 0x20, // 16: OUT Reg 2 (0xA0)

0x82, 0x29, // 17: ROL R2, R2 // C+A0 -> 41 + C

0x28, 0x20, // 18: OUT Reg 2 (0x41)

0x00, 0x00, // 19: NOP

0x47, 0x01, // 1A: BCS RELATIVE JUMP

0x01, 0x00, // 1B: BRK, Test 2 failed: Carry was not set

0xC3, 0x41, // 1C: LDR 03, 41

0xC2, 0xB7, // 1D: SUB R3, R3, R2

0x67, 0x01, // 1E: BEQ RELATIVE JUMP

0x01, 0x00, // 1F: BRK, Test 2 failed: Result is not 0x41

// Test 3: Logic Shift Right (A: LSR)

0x00, 0x00, // 20: NOP

0x06, 0x16, // 21: INC R0

0x18, 0x00, // 22: OUT R0->L (0x03)

0x83, 0x87, // 23: LDR 02, 0x87

0x82, 0x2A, // 24: LSR R2, R2 // 87 -> 43 + Carry

0x28, 0x20, // 25: OUT Reg 2 (0x43)

0x82, 0x2A, // 26: LSR R2, R2 // 43 -> 21 + Carry

0x28, 0x20, // 27: OUT Reg 2 (0x21)

0x47, 0x01, // 28: BCS Relative

0x01, 0x00, // 29: BRK, Test 3 failed: Carry was not set

0xC3, 0x21, // 2A: LDR 03, 21

0xC2, 0xB7, // 2B: SUB R3, R3, R2

0x67, 0x01, // 2C: BEQ relative

0x01, 0x00, // 2D: BRK, Test 3 failed: Result is not 0x21

// Test 4: Arithmetic Shift Right (B: ASR)

0x00, 0x00, // 2E: NOP

0x06, 0x16, // 2F: INC R0

0x18, 0x00, // 30: OUT R0->L (0x04)

0x83, 0x87, // 31: LDR 02, 0x87

0x82, 0x2B, // 32: ASR R2, R2 // 87 -> C3 + Carry

0x28, 0x20, // 33: OUT Reg 2 (0xC3)

0x82, 0x2B, // 34: ASR R2, R2 // C3 -> E1 + Carry

0x28, 0x20, // 35: OUT Reg 2 (0xE1)

0x47, 0x01, // 36: BCS Relative

0x01, 0x00, // 37: BRK, Test 4 failed: Carry was not set

0xC3, 0xE1, // 38: LDR 03, E1

0xC2, 0xB7, // 39: SUB R3, R3, R2

0x67, 0x01, // 3A: BEQ Relative

0x01, 0x00, // 3B: BRK, Test 3 failed: Result is not 0xE1

// Test 5: Add 0x1234 + 0x6DCD = 0x8001

0x06, 0x16, // 3C: INC R0

0x18, 0x00, // 3D: OUT R0->L (0x05)

0x83, 0x34, // 3E: LDR 02, 0x34

0x43, 0xCD, // 3F: LDR 01, 0xCD

0xC2, 0x66, // 40: ADD R3, R1, R2 // 34 + CD = 0x01 + Carry

0x28, 0x30, // 41: OUT high Reg 3 (0x01)

0x47, 0x01, // 42: BCS Relative

0x01, 0x00, // 43: BRK, Test 5 failed: Carry was not set

0x83, 0x12, // 44: LDR 02, 0x12

0x43, 0x6D, // 45: LDR 01, 0x6D

0xC2, 0x6E, // 46: A

0x28, 0x30, // 47: OUT high Reg 3 (0x80)

0xC7, 0x01, // 48: BCC Relative

0x01, 0x00, // 49: BRK, Test 5 failed: Carry was set

0x83, 0x80, // 4A: LDR 02, 0x80

0xC2, 0xB7, // 4B: SUB R3, R3, R2

0x67, 0x01, // 4C: BEQ Relative

0x01, 0x00, // 4D: BRK, Test 5 failed: Result is not 0x80

// Test 6: Sub (0xA765 – 0x34AB) = 0x72BA

0x06, 0x16, // 4E: INC R0

0x18, 0x00, // 4F: OUT R0->L (0x06)

0x83, 0x65, // 50: LDR 02, 0x65

0x43, 0xAB, // 51: LDR 01, 0xAB

0xC2, 0x67, // 52: SUB R3, R2, R1 // 65 - AB = 0x + Carry

0x28, 0x30, // 53: OUT high Reg 3 (0xBA) (No carry, means borrow)

0xC7, 0x01, // 54: BCC Relative

0x01, 0x00, // 55: BRK, Test 6 failed: Carry was set

0x83, 0xA7, // 56: LDR 02, 0xA7

0x43, 0x34, // 57: LDR 01, 0x34

0xC2, 0x6F, // 58: SBC R3, R1, R2 // R2 - R1 + Cin = A7 - 34 + Cin = 72 + Cout

0x28, 0x30, // 59: OUT high Reg 3 (0x72)

0x47, 0x01, // 5A: BCS Relative

0x01, 0x00, // 5B: BRK, Test 6 failed: Carry was not set

0x83, 0x72, // 5C: LDR 02, 0x72

0xC2, 0xB7, // 5D: SUB R3, R3, R2

0x67, 0x01, // 5E: BEQ Relative

0x01, 0x00, // 5F: BRK, Test 6 failed: Result is not 0x72

// Test 7: Rotate Right (C: ROR)

0x00, 0x00, // 60: NOP

0x06, 0x16, // 61: INC R0

0x18, 0x00, // 62: OUT R0->L (0x07)

0x83, 0x36, // 63: LDR 02, 0x36

0x82, 0x2C, // 64: ROR R2, R2 // 36 -> 1B

0x28, 0x20, // 65: OUT Reg 2 (0x1B)

0x82, 0x2C, // 66: ROR R2, R2 // 1B -> 0D + Carry

0x28, 0x20, // 67: OUT Reg 2 (0x0D)

0x82, 0x2C, // 68: ROR R2, R2 // 0D -> 86 + Carry

0x28, 0x20, // 69: OUT Reg 2 (0x86)

0x82, 0x2C, // 6A: ROR R2, R2 // 86 -> C3

0x28, 0x20, // 6B: OUT Reg 2 (0xC3)

0x82, 0x2C, // 6C: ROR R2, R2 // C3 -> 61 + Carry

0x28, 0x20, // 6D: OUT Reg 2 (0x61)

0x00, 0x00, // 6E: NOP

0x47, 0x01, // 6F: BCS RELATIVE JUMP

0x01, 0x00, // 70: BRK, Test 7 failed: Carry was not set

0xC3, 0x61, // 71: LDR 03, 61

0xC2, 0xB7, // 72: SUB R3, R3, R2

0x67, 0x01, // 73: BEQ RELATIVE JUMP

0x01, 0x00, // 74: BRK, Test 7 failed: Result is not 0x61

// Test 8: SRAM Load Store LMA / SMA

0x06, 0x16, // 75: INC R0

0x18, 0x00, // 76: OUT R0->L (0x08)

0x43, 0x22, // 77: LDR R1, #11

0x4A, 0x10, // 78: SMA R1, #10 // Store 11 in address 10

0x83, 0x92, // 79: LDR R2, #22

0x8A, 0x20, // 7A: SMA R2, #20 // Store 22 in address 20

0xC3, 0xDE, // 7B: LDR R3, #33

0xCA, 0x30, // 7C: SMA R3, #30 // Store 33 in address 30

0xC9, 0x20, // 7D: LMA R3, #20 // Load value at address 20 into R3 (R3 = 22)

0x89, 0x10, // 7E: LMA R2, #10 // Load value at address 10 into R2 (R2 = 11)

0x49, 0x30, // 7F: LMA R1, #30 // Load value at address 30 into R1 (R1 = 33)

0x28, 0x10, // 80: OUT R1 High (0x33)

0x42, 0x97, // 81: SUB R1, R1, R2

0x28, 0x10, // 82: OUT R1 High (0x22)

0x42, 0xD7, // 83: SUB R1, R1, R3

0x28, 0x10, // 84: OUT R1 High (0x00)

0x67, 0x01, // 85: BEQ RELATIVE JUMP

0x01, 0x00, // 86: BRK, Test 8 failed: Result is not 0x00

// Test 9: SRAM Register Load/Store LMR / SMR

0x06, 0x16, // 87: INC R0

0x18, 0x00, // 88: OUT R0->L (0x09)

0x43, 0x05, // 89: LDR R1, 0x05

0x83, 0x20, // 8A: LDR R2, #20

0x4C, 0x20, // 8B: SMR R1, R2 // Store 5 in address 20

0x28, 0x10, // 8C: OUT High, R1 // (20:0x05 , 21:0x04, 22:0x03, 23:0x02, 24:0x01)

0x86, 0x16, // 8D: INC R2

0x46, 0x17, // 8E: DEC R1

0xE7, 0xFB, // 8F: BNE back to Store

0x43, 0x05, // 90: LDR R1, 0x05

0x83, 0x24, // 91: LDR R2, #24

0xCB, 0x20, // 92: LMR R3 from @(R2)

0x28, 0x30, // 93: OUT High, R3 (0x01, 0x02, 0x03, 0x04, 0x05)

0x28, 0x20, // 94: OUT High, R2 (0x24, 0x23, 0x22, 0x21, 0x20)

0x86, 0x17, // 95: DEC R2

0x46, 0x17, // 96: DEC R1

0xE7, 0xFA, // 97: BNE back to Store

0xC6, 0x57, // 98: SUB R3 = R3 - #5

0x67, 0x01, // 99: BEQ RELATIVE JUMP

0x01, 0x00, // 9A: BRK, Test 9 failed: R3 != #5

// End of BIST

0x06, 0x16, // 87: INC R0

0x18, 0x00, // 88: OUT R0->L (0x09)

0x01, 0x00 // 89: BRK (end of BIST)

};

# // ROBERTSON'S MULTIPLICATION SIGNED version 3 (with rotate right)

byte robertsonv3[] = {

0x03, 0x00, // 0 LDR R0, #0

0x43, 0x80, // 1 LDR R1, #80 (-128)

0x83, 0x01, // 2 LDR R2, #1 (1) --> Expect 0x0080 //--> Expect 0xF71D (-2275)

0xc3, 0x07, // 3 LDR R3, #7 (0x07)

0x46, 0x0a, // 4 LSR R1 (Shift right using immediate opcode R1[0] -> Carry)

0xc7, 0x01, // 5 BNC + 1 (relative jump over 1 instruction)

0x02, 0x26, // 6 ADD R0, R0, R2

0x46, 0x08, // 7 LSL R1 (resume R1 position)

0x06, 0x0b, // 8 ASR R0 (Shift Right: R0 -> Carry -> R1)

0x46, 0x0c, // 9 ROR R1

0x28, 0x00, // A OUT R0 H

0x18, 0x10, // B OUT R1 L

0xc6, 0x17, // C SUB R3, #1

0xe7, 0xf6, // D BNZ 4 (relative jump back 9 instructions by substracting 10 from PC)

0x46, 0x0a, // E LSR R1 (Shift right using immediate opcode R1[0] -> Carry)

0xc7, 0x01, // F BNC + 1 (relative jump over 1 instruction)

0x02, 0x87, // 10 SUB R0, R0, R2 R0 = R0 - R2

0x46, 0x08, // 11 LSL R1 (resume R1 position)

0x06, 0x0b, // 12 ASR R0 (Shift Right: R0 -> Carry -> R1)

0x46, 0x0c, // 13 ROR R1

0x28, 0x00, // 14 OUT R0 H

0x18, 0x10, // 15 OUT R1 L

0x01, 0x00 // 16 BRK

};

# // Hierarchical Loops

byte TwoLoops[] = {

0x03, 0x01, // 0: LDR R0, 0x01

0x43, 0x00, // 1: LDR R1, 0x00

0x83, 0x00, // 2: LDR R2, 0x00

0xC3, 0x00, // 3: LDR R3, 0x00

0xC2, 0xC6, // 4: ADD R3, R3, R0

0x18, 0x30, // 5: OUT Low, R3

0xE5, 0x04, // 6: Branch Not Zero, 4

0x82, 0x86, // 7: ADD R2, R2, R0

0x28, 0x20, // 8: OUT HIGH R2

0xE5, 0x04, // 9: Branch Not Zero, 4

0x01, 0x00 // 10: Break

};