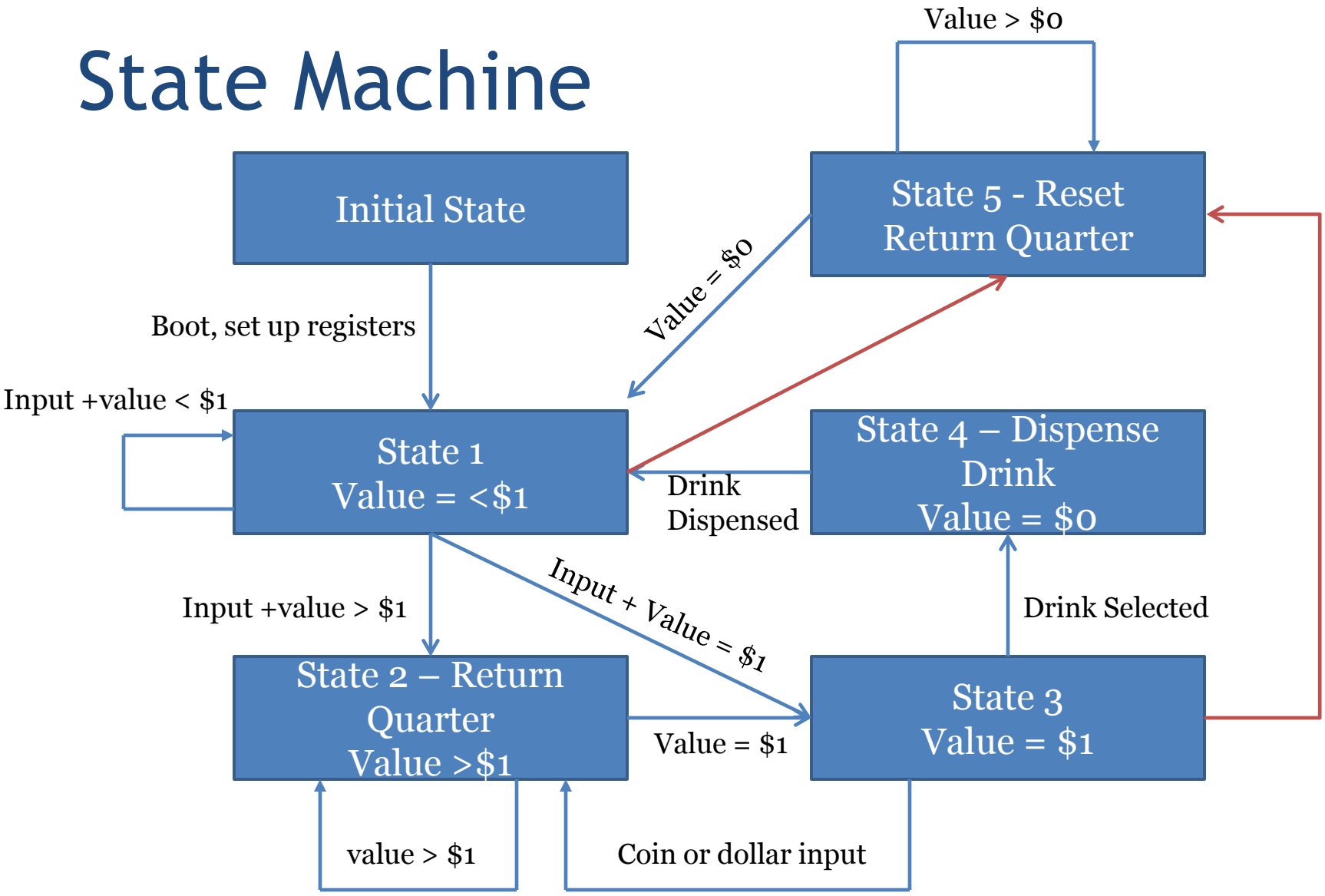
State Machine Example

State Machine with AVR Assembly

Example – Vending Machine

- Assume we have a vending machine with the following use case:
  - A drink cost $1.00
  - The machine only accepts $1 bills or quarters
    - Inserting a quarter makes PA0 go high
    - Inserting a dollar bill makes PA1 go high
  - A change return button gives back in quarters the current value
    - Return button is connected to PA2
    - Output PA3 high to return a coin
  - The machine accepts at max $1.
    - i.e. if there is $0.75 in the machine and $1 bill is inserted, it will return $0.75 to the user
  - There are 8 drink choices, and choice is made after $1 is inserted
    - These 8 choices are connected to 8 pins on Port B

State Machine



Assembler Set-Up

```
.include “m169pdef.inc”
.DEF value=R16  ; Define registers for state machine
.DEF state=R17    ;
.DEF temp=R18    ;
LDI R0, HIGH(RAMEND); Set up stack pointer
OUT SPH,R0;
LDI R0, LOW(RAMEND);
OUT SPL,R0;
LDI value, 0; Set initial value to 0
LDI state, 1; Set initial state to 1
LDI temp, 0b00000000;
OUT DDRB, temp; set Port B to all input
LDI temp, 0b11111000;
OUT DDRA, temp; set Port A0-2 to input
MAIN: (Continue from main loop here)
```

Assembler Main Loop (input)

```
MAIN:IN temp, PINA; Check input from PortA
SBIC temp, 0; Skip if quarter not inserted
ADDI value,1; Add 1 to value for quarter
SBIC temp,1; Skip if dollar not inserted
ADDI value,4; Add 4 to value for dollar
SBIC temp,2; Skip if reset not pressed
JMP RESET; Jump to reset code
```

Assembler Main Loop (State Machine)

```
MOV temp, value; Copy value to temp
SUBI temp,4; Subtract a dollar from temp
BRMI S1; Jump to S1 if less than $1
BRNE S3; Jump to state 3 if exactly $1
LDI state, 2;
JMP: OVERFLOW; Return all excess quarters
S1: LDI state, 1;
JMP MAIN; Jump back to main
S3:LDI state, 3;
IN temp, PINB; Load Port B to temp
ORI temp, 8b00000000;
BREQ MAIN; No drink selection, back to main
OUT temp, PORTC; Put drink selection out to portC
LDI value, 0; Set value back to 0
JMP MAIN; Go back to main
```