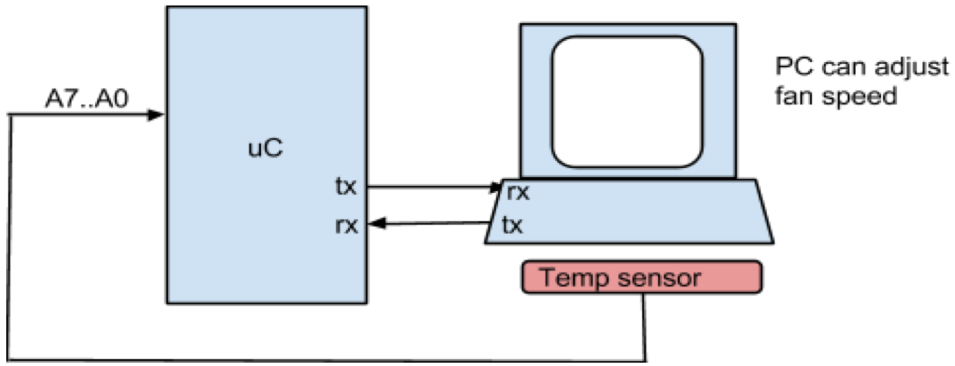
Homebrew Temperature Sensor

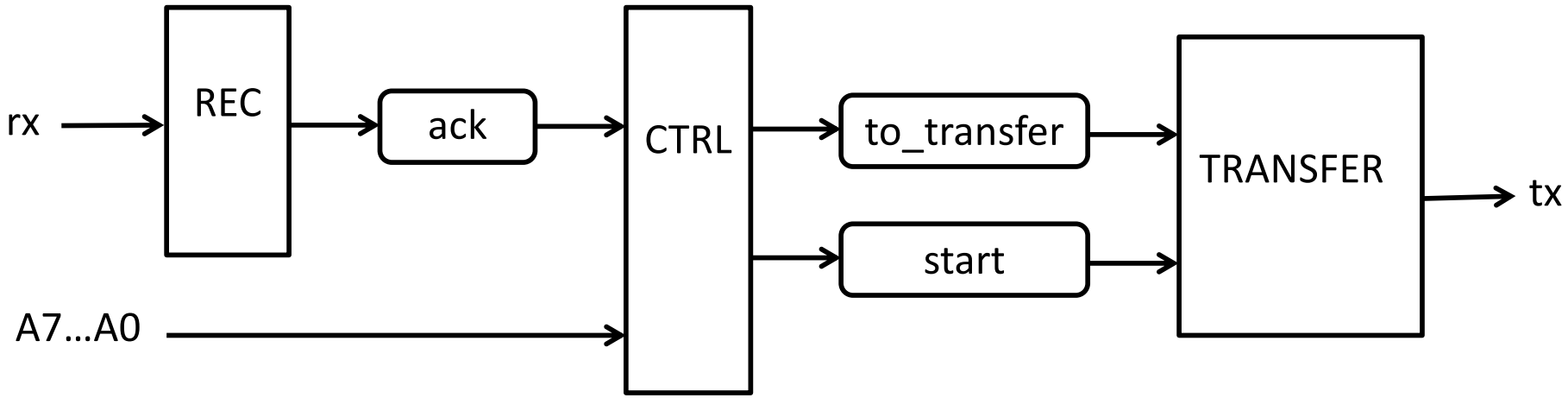
Solution

(It Burns!!!!!!!!!)



Solution #1

unsigned char ack, to_transfer, start;



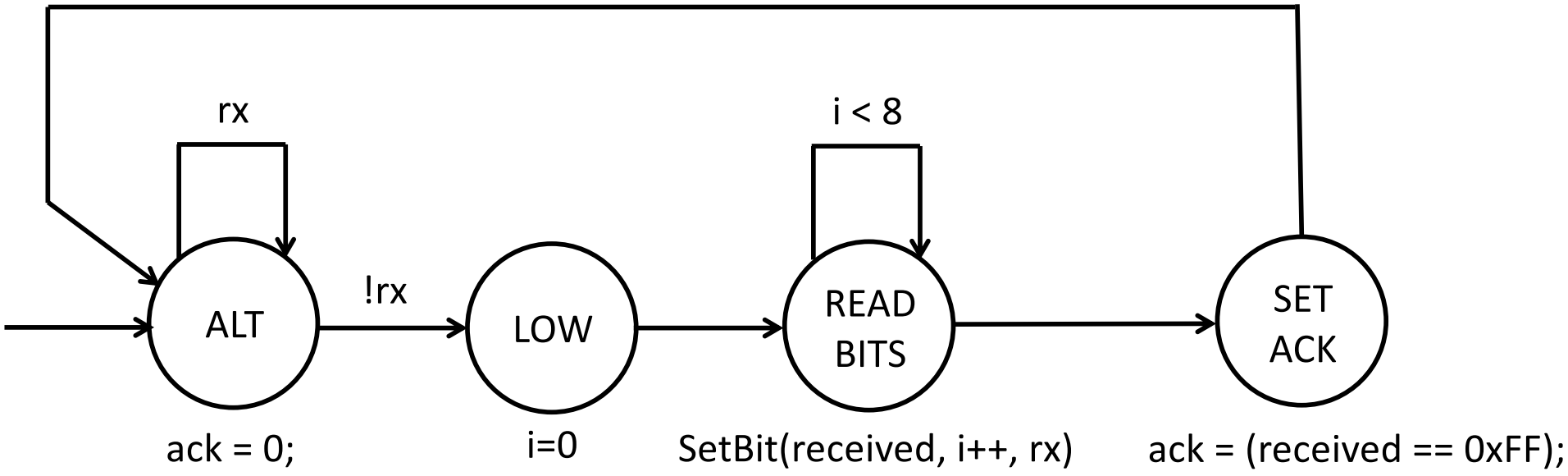
Assumptions

- All synchSMs have the same period (16.7ms ≈ 60bps)
- Ignore glitches
- 1 second limit starts at the “start” flag being set, not after completing sending
- 16.7ms x 60 = 1.002s ≈ 1s
- After 1s has passed, rx will not send a messageThe PC transmits a value of 1 to the microcontroller (read from rx) to indicate that the channel is idle. It will lower the value to 0 just before transmitting data.

REC

Period = 16.7ms

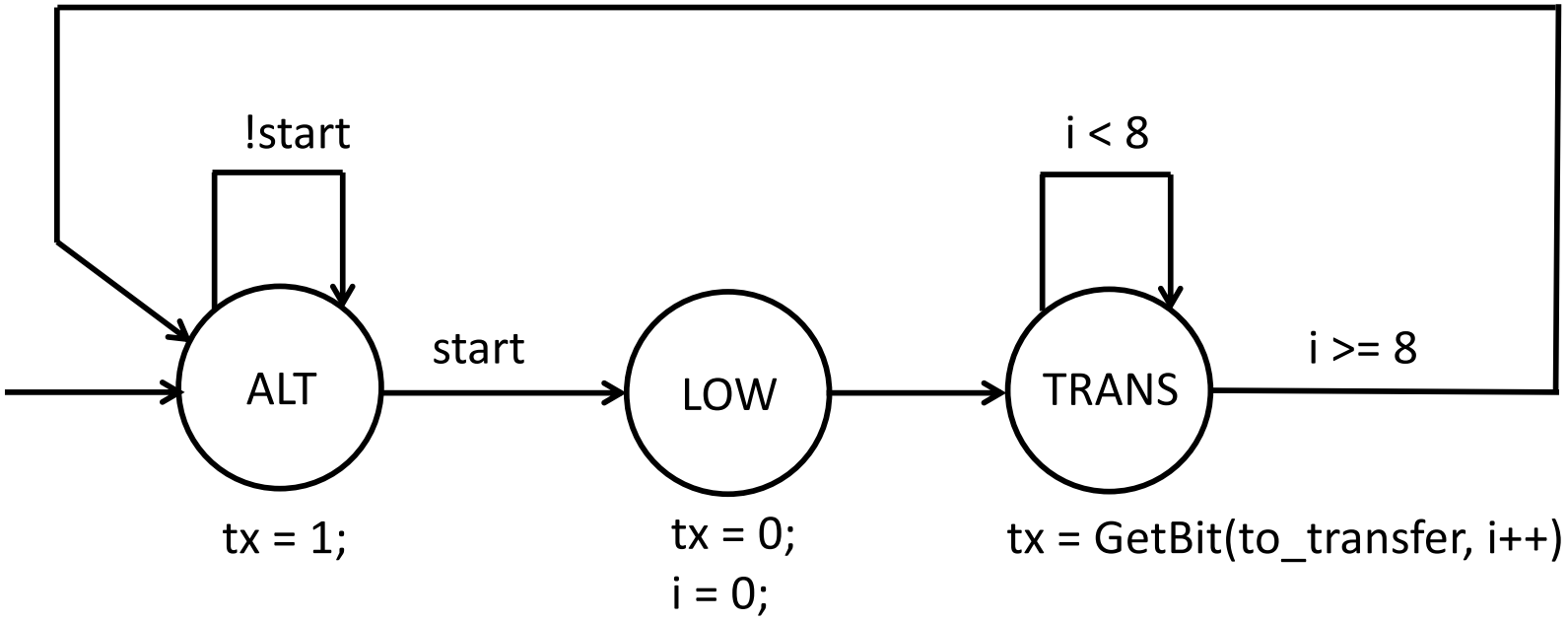
unsigned char received, i;



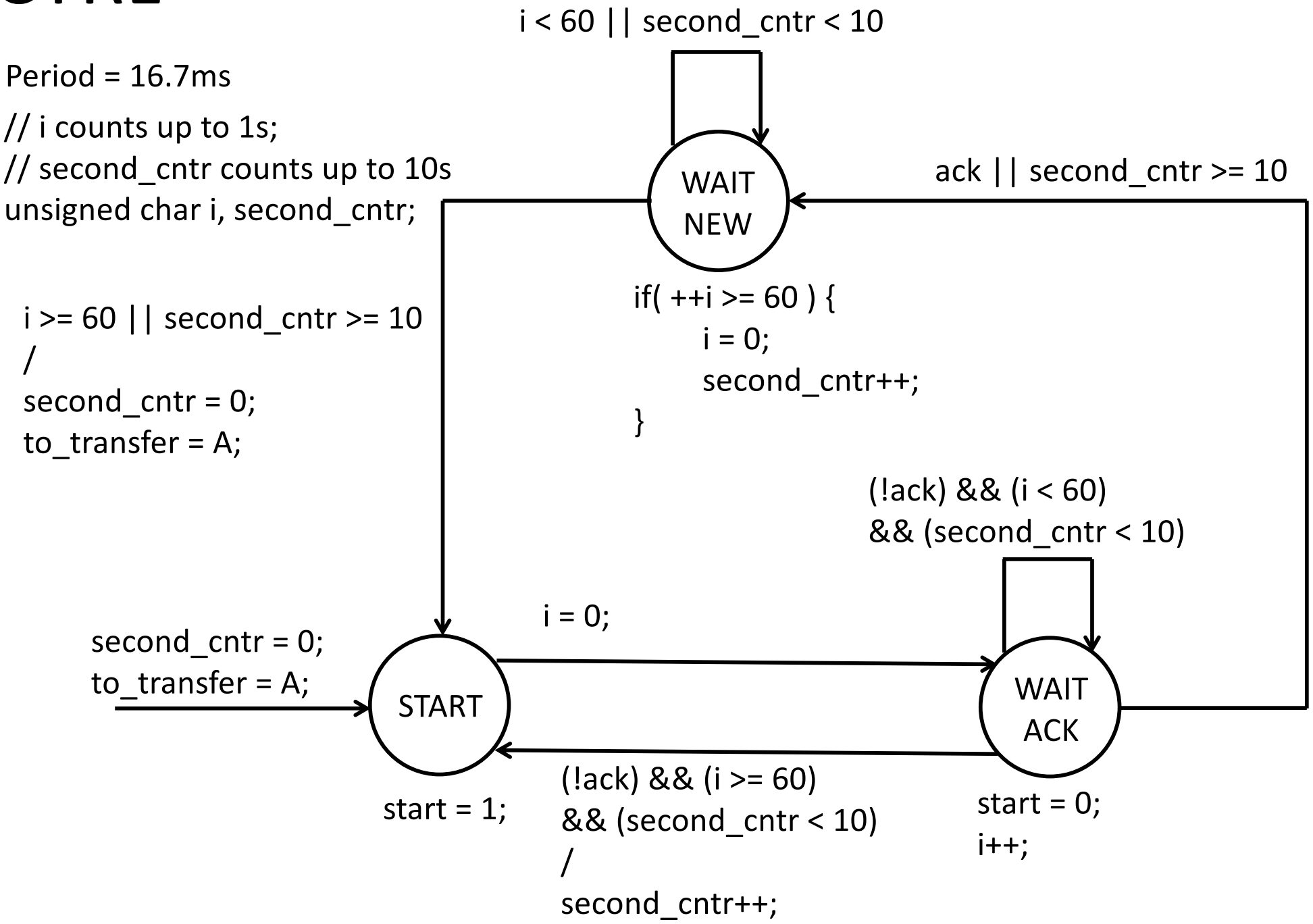
TRANSFER

Period = 16.7ms

unsigned char i;



CTRL

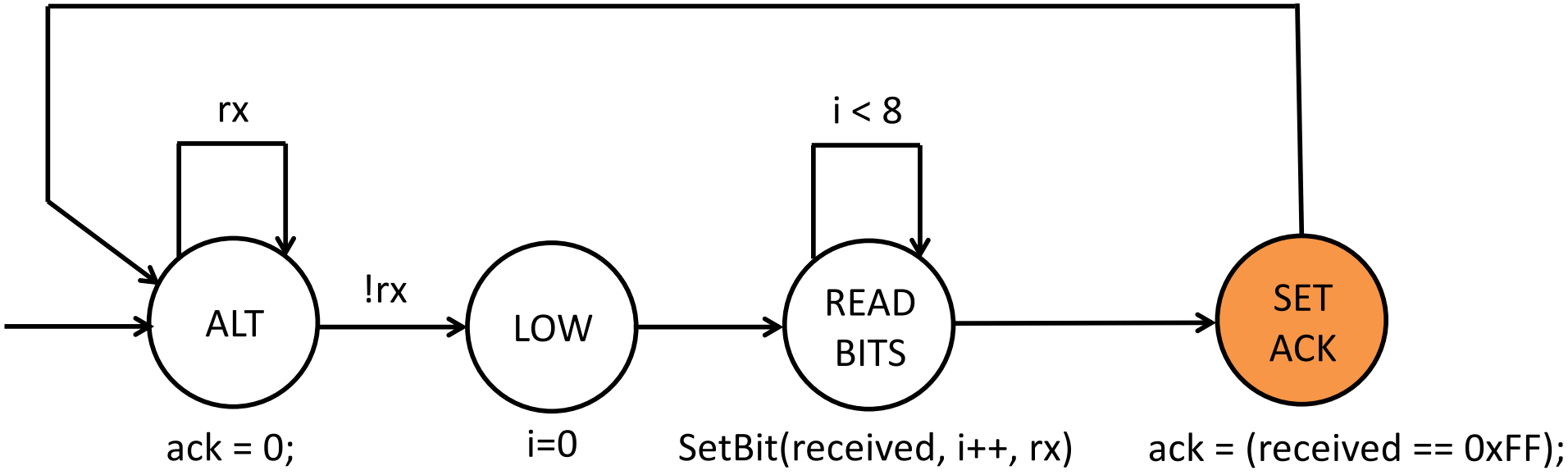

i < 60 || second_cntr < 10
Period = 16.7ms
// i counts up to 1s;
// second_cntr counts up to 10s
unsigned char i, second_cntr;
WAIT NEW
ack || second_cntr >= 10
if(++i >= 60) {
i = 0;
second_cntr++;
}
i >= 60 || second_cntr >= 10
/
second_cntr = 0;
to_transfer = A;
(!ack) && (i < 60)
&& (second_cntr < 10)
i = 0;
second_cntr = 0;
to_transfer = A;
START
WAIT ACK
start = 1;
(!ack) && (i >= 60)
&& (second_cntr < 10)
/
second_cntr++;
start = 0;
i++;

Solution #1 – Key Issues

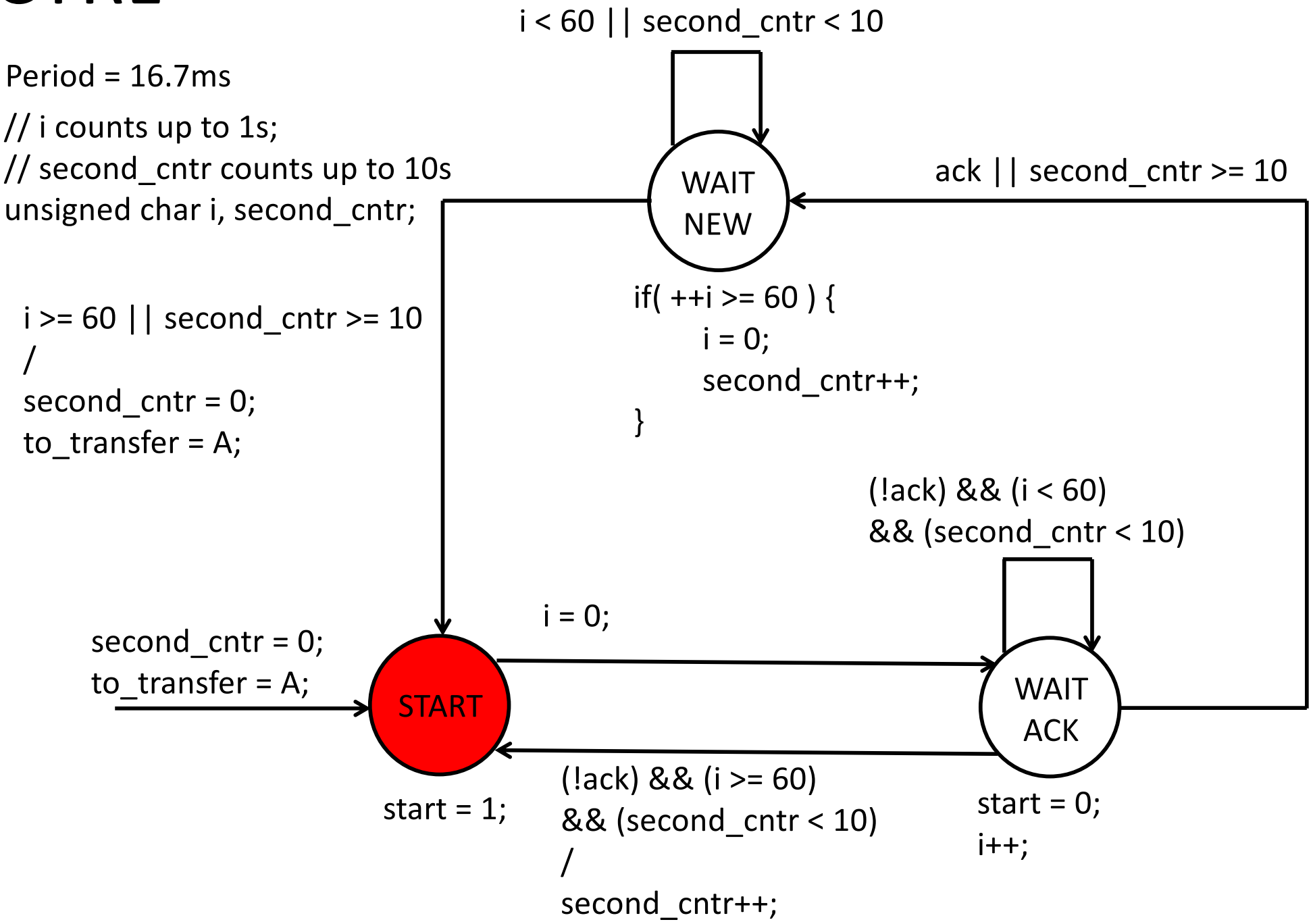
- REC and TRANSFER tasks are simple and reasonable
- CTRL task is unnecessarily complex
  - The logic for counting up to 1s and to 10s and processing ACKs is completely intertwined
  - The logic for retransmission and transmitting the next sensed value (after 10s) is completely intertwined

Solution #1 – Subtle Bug in REC Task

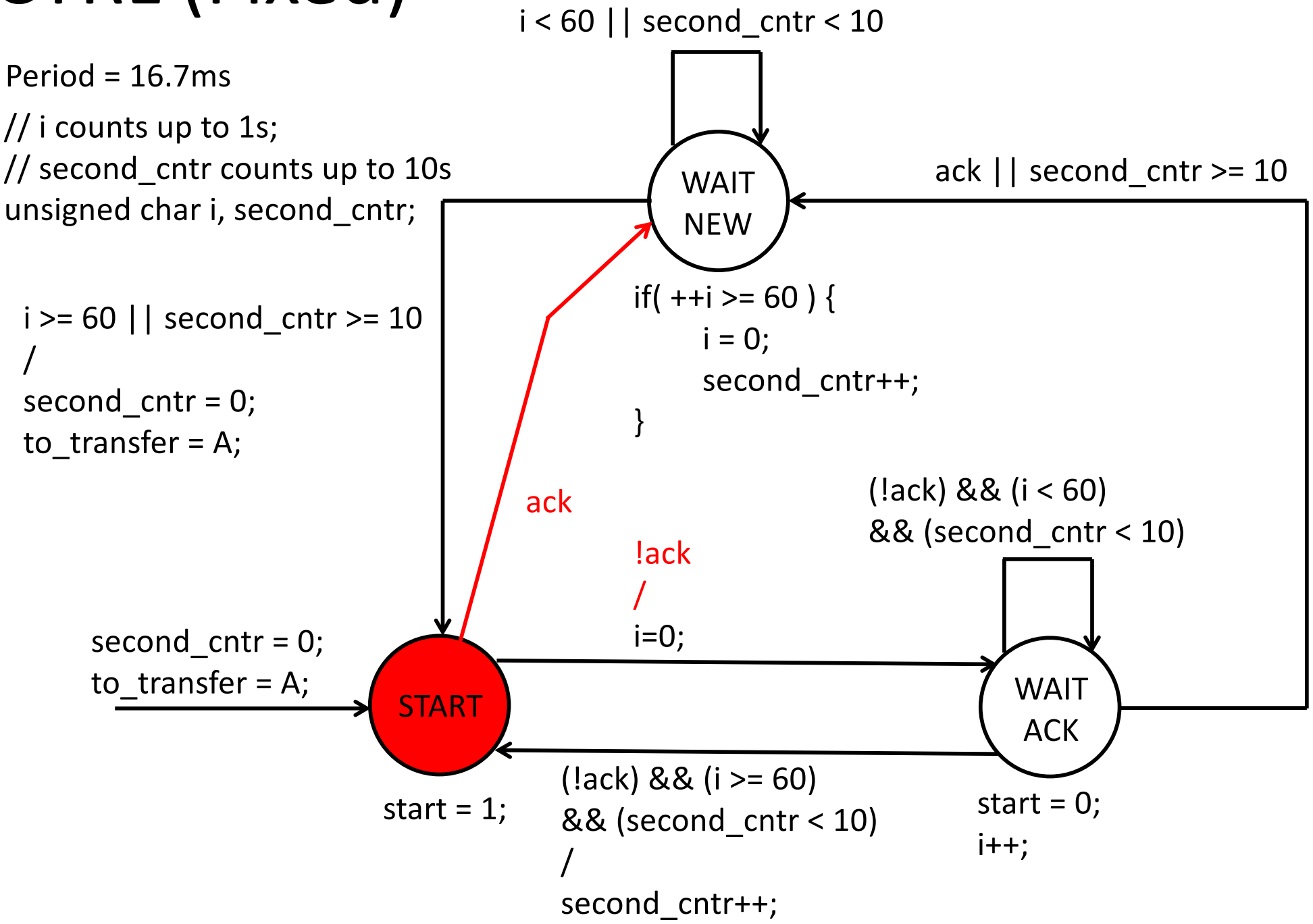
The ack signal is set for EXACTLY one tick per iteration
- This means that the CTRL task must read ack in EVERY state



CTRL


i < 60 || second_cntr < 10
Period = 16.7ms
// i counts up to 1s;
// second_cntr counts up to 10s
unsigned char i, second_cntr;
WAIT NEW
ack || second_cntr >= 10
if(++i >= 60) {
i = 0;
second_cntr++;
}
i >= 60 || second_cntr >= 10
/
second_cntr = 0;
to_transfer = A;
(!ack) && (i < 60)
&& (second_cntr < 10)
i = 0;
second_cntr = 0;
to_transfer = A;
START
WAIT ACK
start = 1;
(!ack) && (i >= 60)
&& (second_cntr < 10)
/
second_cntr++;
start = 0;
i++;

CTRL (Fixed)


i < 60 || second_cntr < 10
Period = 16.7ms
// i counts up to 1s;
// second_cntr counts up to 10s
unsigned char i, second_cntr;
WAIT NEW
ack || second_cntr >= 10
i >= 60 || second_cntr >= 10
/
second_cntr = 0;
to_transfer = A;
if(++i >= 60) {
i = 0;
second_cntr++;
}
ack
(!ack) && (i < 60)
&& (second_cntr < 10)
!ack
/
i=0;
second_cntr = 0;
to_transfer = A;
START
WAIT ACK
start = 1;
(!ack) && (i >= 60)
&& (second_cntr < 10)
/
second_cntr++;
start = 0;
i++;

Toward Solution #2

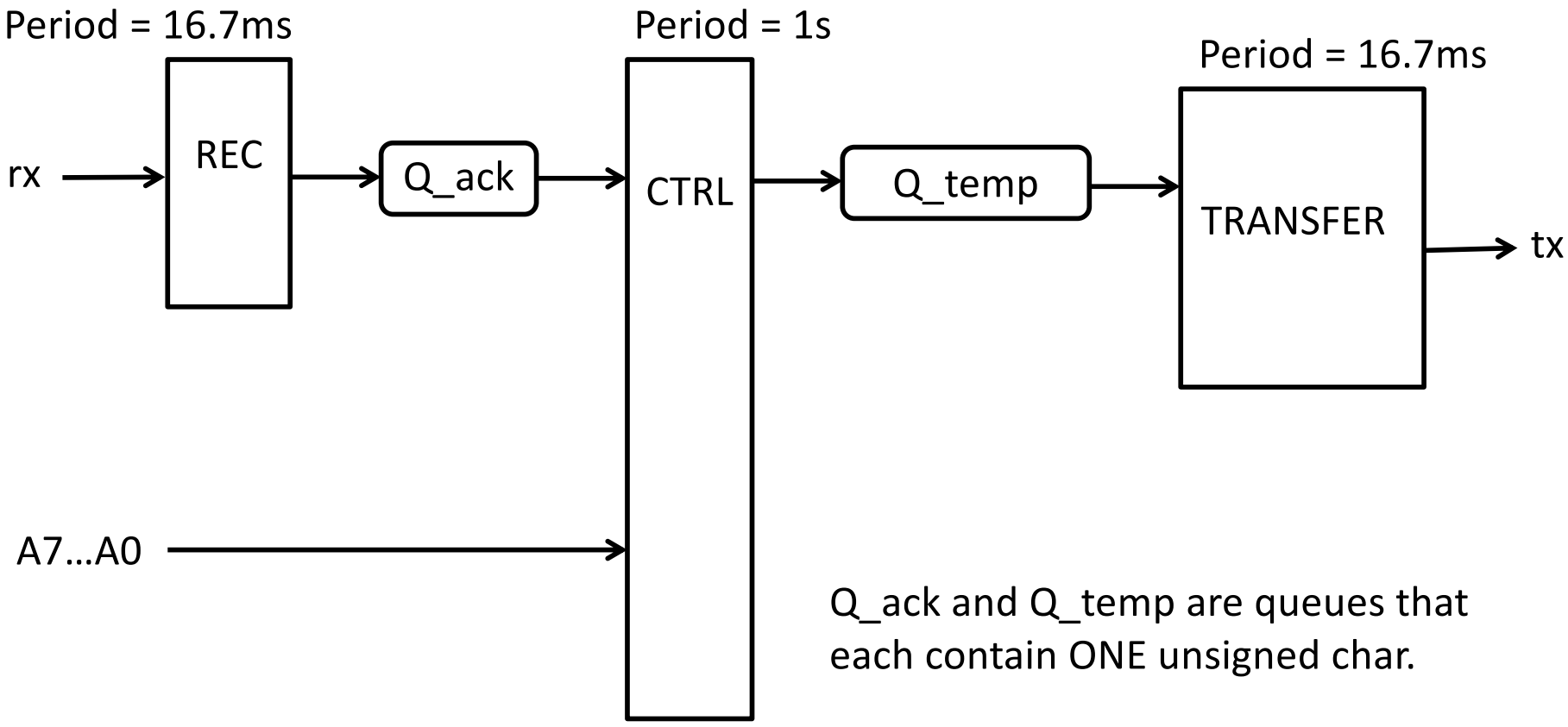
Key Points (Controller):

- Transmit new Temp every 10s
- Retransmit current Temp every 1s until we receive an ACK

Key Points (General):

- Everything complex in Solution #1 can be simplified with a queue

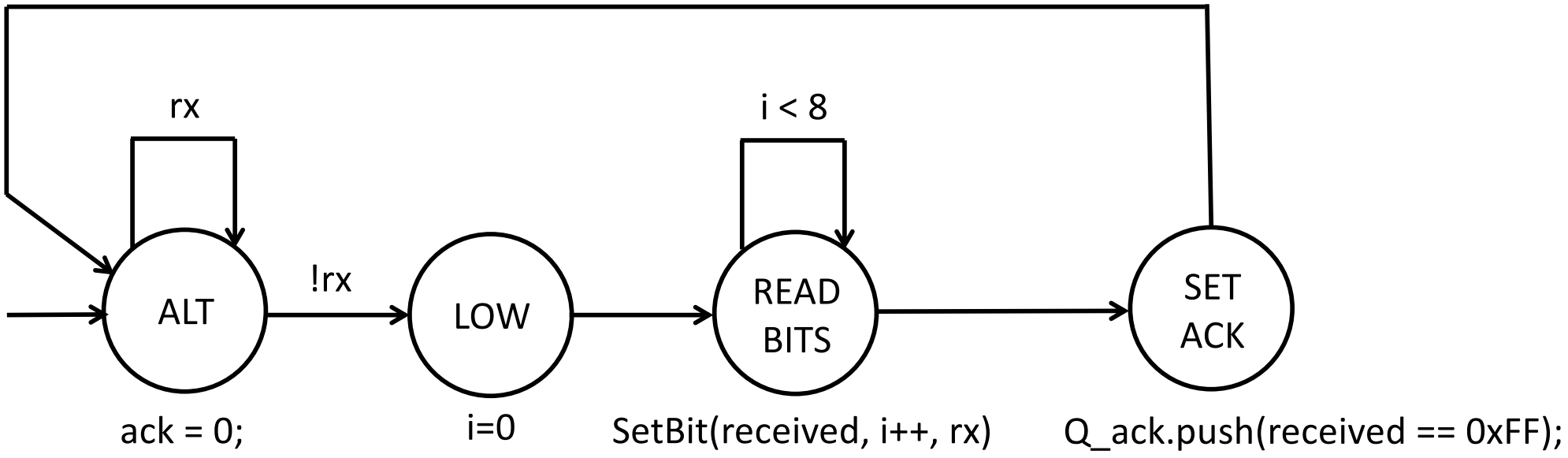
Solution #2


Period = 16.7ms
Period = 1s
Period = 16.7ms
rx
REC
Q_ack
CTRL
Q_temp
TRANSFER
tx
A7…A0
Q_ack and Q_temp are queues that each contain ONE unsigned char.

REC

Period = 16.7ms

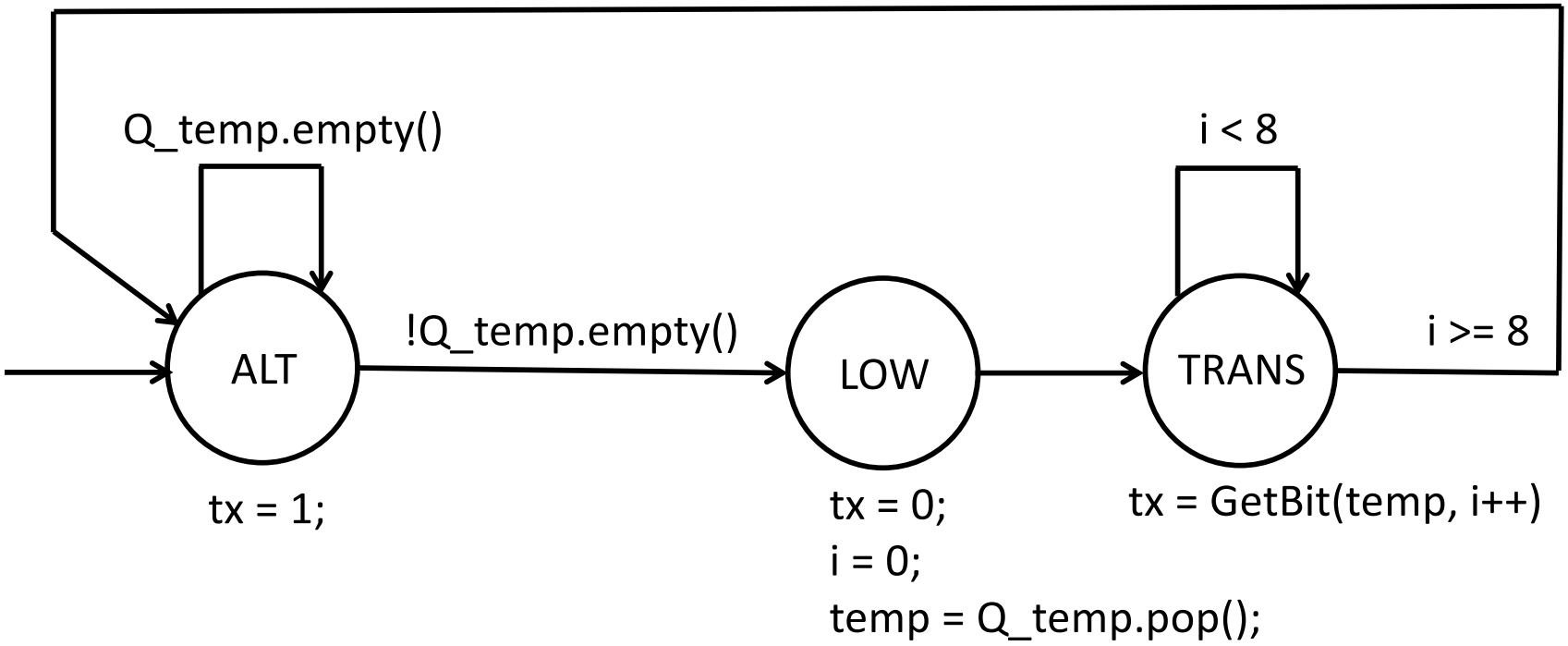
unsigned char received, i;



TRANSFER

Period = 16.7ms

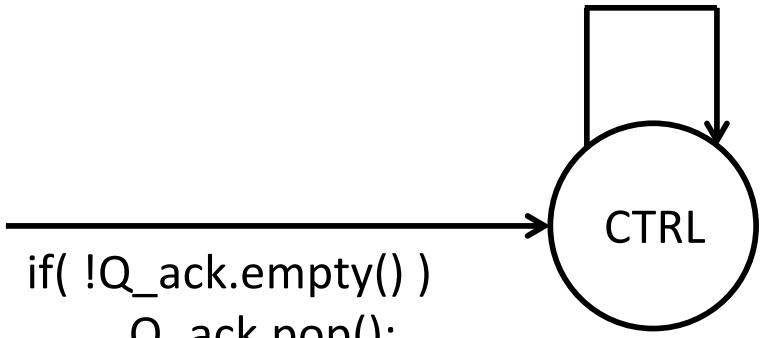
unsigned char i, temp;



CTRL

Period = 1s

```
unsigned char temp;
unsigned char sec_cnt;
unsigned char ack_rcvd;
```




```
if( !Q_ack.empty() )
        Q_ack.pop();
sec_cnt = 0;
ack_rcvd = 0;
temp = A;
```

```
// Check for an ACK
if( !Q_ack.empty() ) {
        Q_ack.pop();
        ack_rcvd = 1;
}

// If we hit 10 seconds, transmit
// a new temperature and reset
if( ++sec_cnt >= 10 ) {
        temp = A;
        Q_temp.push(temp);
        sec_cnt = 0;
        ack_rcvd = 0;
}

// Else, retransmit the current
// temperature if we have not yet
// received an ACK
else if( !ack_rcvd )
        Q_temp.push(temp);
}
```

And the Moral of the Story Is…

- Queues make ~~everything~~ many things better
- Avoid “flashing” the value of a shared variable for a small number of ticks, as this requires complex cross-task synchronization
- CS/EE 120B can help you design networking protocols too!