/\*This function checks whether or not the start button has been pressed in. If it has, we post a new event ("button pushed") to the state machine and note the time at which this occurs.\*/

**returns True or False: Check4ButtonPushed (takes nothing)**

{

declare a static local variable CurrentButtonState of type character

declare a static local variable LastButtonState of type character and set it equal to 0

declare a local variable TimeOfCurrentSample of type integer

declare a local variable ReturnVal of type True or False

declare a local variable new\_event of type "event"

declare a static local variable TimeOfLastSample of type integer and set it equal to 0

set ReturnVal to False

read the current state of Port AD pin 2 and set CurrentButtonState equal to this value

set TimeOfCurrentSample to the current time

if the difference between TimeOfCurrentSample and TimeOfLastSample is greater than the debounce time:

{

if the CurrentButtonState is not equal to the LastButtonState:

{

if the CurrentButtonState is high (button has been pushed in):

{

set new\_events type to "button pushed"

note that new\_event occured at TimeOfCurrentSample

post this event to the DiscoCubes state machine

set ReturnVal to True

}

set LastButtonState to CurrentButtonState

set TimeOfLastSample to TimeOfCurrentSample

}

return ReturnVal

}

}

End of Check4ButtonPushed

/\*This function checks whether the lower limit of motion has been achieved (i.e. whether or not an IR sensor has been tripped). Normally, there is nothing between the IR emitter and NPN phototransistor, so current flows in the circuit. When the limit of motion has been reached, a piece of wood blocks the IR from reaching the transistor, and the current flow is blocked.\*/

**returns True or False: Check4IRDownLimitSwitch (takes nothing)**

{

declare a local variable CurrentIRDownState of type character

declare a local variabl new\_event of type "event"

declare a local variable ReturnVal of type True or False

declare a static local variable LastIRDownState of type character

set ReturnVal to False

read the current state of port T pin 6 and set CurrentIRDownState equal to this value

if the CurrentIRDownState is not equal to the LastIRDownState:

{

if the CurrentIRDownState is high (i.e., current is not flowing, and voltage accumulates):

{

set new\_events type to "ir down limit switch tripped"

post this event to the DiscoCubes state machine

set ReturnVal to True

}

}

set LastIRDownState to CurrentIRDownState

return ReturnVal

}

End of Check4IRDownLimitSwitch

/\*\*This function checks whether the upper limit of motion has been achieved (i.e. whether or not an IR sensor has been tripped). Normally, there is nothing between the IR emitter and NPN phototransistor, so current flows in the circuit. When the limit of motion has been reached, a piece of wood blocks the IR from reaching the transistor, and the current flow is blocked.\*/

**returns True or False: Check4IRUpLimitSwitch (takes nothing)**

{

declare a local variable CurrentIRUpState of type character

declare a local variable new\_event of type "event"

declare a local variable ReturnVal of type True or False

declare a static local variable LastIRUpState of type character

set ReturnVal to False

read the current state of port T pin 7 and set CurrentIRUpState equal to this value

if the CurrentIRUpState is not equal to the LastIRUpState:

{

if the CurrentIRUpState is high (i.e., current is not flowing, and voltage accumulates):

{

set new\_events type to "ir down limit switch tripped"

post this event to the DiscoCubes state machine

set ReturnVal to True

}

}

set LastIRUpState to CurrentIRUpState

return ReturnVal

}

End of Check4IRUpLimitSwitch

/\*This function checks whether or not a cube placed on top of a pillar on the game board is correct or not. This is achieved by comparing the internal representation of the computer's answer (the random color sequence generated by the computer which the user attempts to match) with the actual answer provided by the user (the color and position of the cubes on the pillars). If a cube has been placed on a pillar correctly or incorrectly, i.e. that cube's color matches or does not match the computer's color for that position, this function notes for which pillar this event occured and posts an event to the state machine.\*/

**returns True or False: Check4CubeCorrect (takes nothing)**

{

declare a local variable new\_event of type "event"

declare a local variable answer of type character array

declare a local variable CurrentReedSwitchState of type character array

declare a local static variable LastReedSwitchState of type character array

declare a local variable j of type character

declare a local variable i of type character

declare a local variable t of type integer

declare a local variable ReturnVal of type True or False

set j equal to 0

set ReturnVal to False

set t to the current time

set answer to the current internal representation of the computers answer

set Port AD pin 0 to low. now the data on the parallel-in, serial-out registers will appear at the C32.

do nothing for 1 millisecond to ensure a clean transition.

set Port AD pin 1 to high. now the data on the parallel-in, serial-out registers can be shifted serial out onto the C32

for a loop where the variable i goes from 0 to 14 (do the following 15 times):

{

if Port T pin 4 is high (i.e., a magnetic reed switch is activated):

{

set element i/3 of CurrentReedSwitchState to i%3. since i is an iteger, this means that for i between 0 and 2, i/3 will be zero. for i between 3 and 5, i/3 will be 1 and so on. i%3 follows the pattern 0,1,2,0,1,2... for i starting at 0 and going to 14.

if element i/3 of the CurrentReedSwitchState is not equal to element i/3 of the LastReedSwitchState:

{

if element i/3 of the computers answer is the same as element i/3 of the CurrentReedSwitchState (i.e. the player is correct):

{

set new\_events type to "cube is correct"

note that this event occured for cube number i/3

post this event to the DiscoCubes state machine

}else (the player has placed a cube incorrectly):

{

set new\_events type to "cube is incorrect"

note that this event occured for cube number i/3

post this event to the DiscoCubes state machine

}

set element i/3 of LastReedSwitchState to element i/3 of CurrentReedSwitchState

}

}else (Port T pin 4 is low, so this magnetic reed switch is unactivated):

{

if i%3 is equal to 0 (the first magnetic reed switch on a given pillar is open):

{

set j equal to i

increment j by one

}else if j is equal to i:

{

if i%3 is equal to one (the second magnetic reed switch on a given pillar is also open):

{

increment j by one

}

if i%3 is equal to two (the third magnetic reed switch on a given pillar is also open):

if we got here, that means for a given pillar, no reed switches are active (all are open)

{

set element i/3 of the CurrentReedSwitchState to '#' which indicates no cube present

set element i/3 of LastReedSwitchState equal to element i/3 of CurrentReedSwitchState

}

}

}

set ReturnVal equal to True

Pulse the clock to read the next bit into the C32

}

return ReturnVal

}

End of Check4CubeCorrect

/\*This function pulses the clock of the parallel-in, serial-out registers to serially shift data into the C32.\*/

**returns nothing: PulseCLK (takes nothing)**

{

declare a local variable i of type integer

set Port M pin 5 low

declare one dummy variable to waste a little time

set Port M pin 5 high

declare another dummy variable to waste more time

print nothing to the terminal to waste even more time to ensure that all devices registered the change from low to high

}

End of PulseCLK