

**A MEDIA ACCESS CONTROL
PROTOCOL FOR THE SPECS
PROJECT**

Candice Jang

**University of New Mexico
Computer Science**

**Summer Undergraduate Program in Engineering
Research at Berkeley (SUPERB 2003)**

Faculty Mentor: Dr. Jan Rabaey

Staff Mentor: Fred Burghardt

**College of Engineering
University of California, Berkeley**

ABSTRACT

SPECs, or Small Personal Everyday Computers, is a joint research effort between the Berkeley Wireless Research Center (BWRC) and Hewlett Packard Labs. SPECs are sensor nodes that are small in size, require very low levels of power to operate, and communicate with each other through radio links. The nodes can transmit and receive small amounts of data (such as ID, node status, and attributes) without the need for a central computer when they are in close enough proximity of one another. Nodes are inexpensive and can be attached to people and objects in an unobtrusive manner so that they can interact with each other. There are many applications for this type of exchange of information, such as inventory control. The goal of this project is to design and implement a media access control protocol for the nodes used in SPECs. After successful completion of the protocol, it will be tested under real-world conditions using a test bed.

INTRODUCTION

Small Personal Everyday Computers, or SPECs, range from cellular phones to surveillance tools that make life easier for the average consumer. SPECs can be worn or attached to a person, place, or object and networked without a centralized computer. Because SPECs can communicate with one another easily, many practical applications can be supported. For example, given the numerous encounters with objects (keys, PDAs, etc) an individual has during a typical day, it can be difficult to track them by human memory. If SPECs were attached to objects, places, and people that the individual has frequent contact with, he can keep a “log” of these encounters. For example, if a college student has several textbooks that he needs to remember to bring home at the end of the day, his SPEC will recognize the books and remind him to take them home.

The SPECs project is a joint effort of the Berkeley Wireless Research Center (BWRC) and Hewlett Packard Labs. It explores networks of very small, low-power embedded System-On-Chip (SOC) devices that use radio links to communicate with each other. Because of the large number of devices in the network, wiring them for power or changing batteries is not cost-effective. SPECs derive power from environmental sources such as light and vibration and consume very little power. Because of the low power requirement, networks must be dense so that transmission ranges are short, to keep transmission power at a minimum.

In order to get an idea of how the system will operate, first generating prototype nodes are under development. There are three major parts to these prototypes. The hardware node includes a small microcontroller, a Field Programmable Gate Array (FPGA), a radio, and power supplies. Protocols dictate how nodes communicate with one another, e.g. when is it okay to transmit information, how data is encoded in a packet, how to process received packets, etc. Lastly, the application makes use of the collected data.

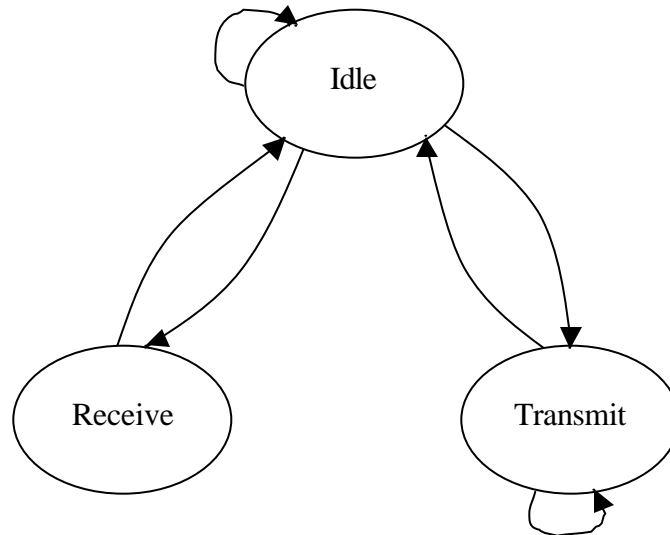
The focus of this project is the communication protocol portion. The protocol consists of several “layers” that work together to transmit and receive packets, analyze their content, compose and respond to requests, and route packets. The layers are defined in a hierarchy of abstraction, with the lowest being the physical layer, followed by the MAC/Datalink layer, and the highest being the application layer.

The Media Access Control (MAC)/Datalink layer coordinates communication between nodes within transmission range of each other. It controls access to radio channels and tells the physical layer when to transmit and which channels to use. The MAC layer handles data received from and transmitted to the physical layer as well.

The following steps were taken to implement the MAC layer: design, implementation, simulation, and testing.

DESIGN

The sensor nodes will be able to transmit and receive data from each other, so each node has dual capability. There are three states that a node can be in: idle, transmit, and receive, as illustrated in the following diagram.



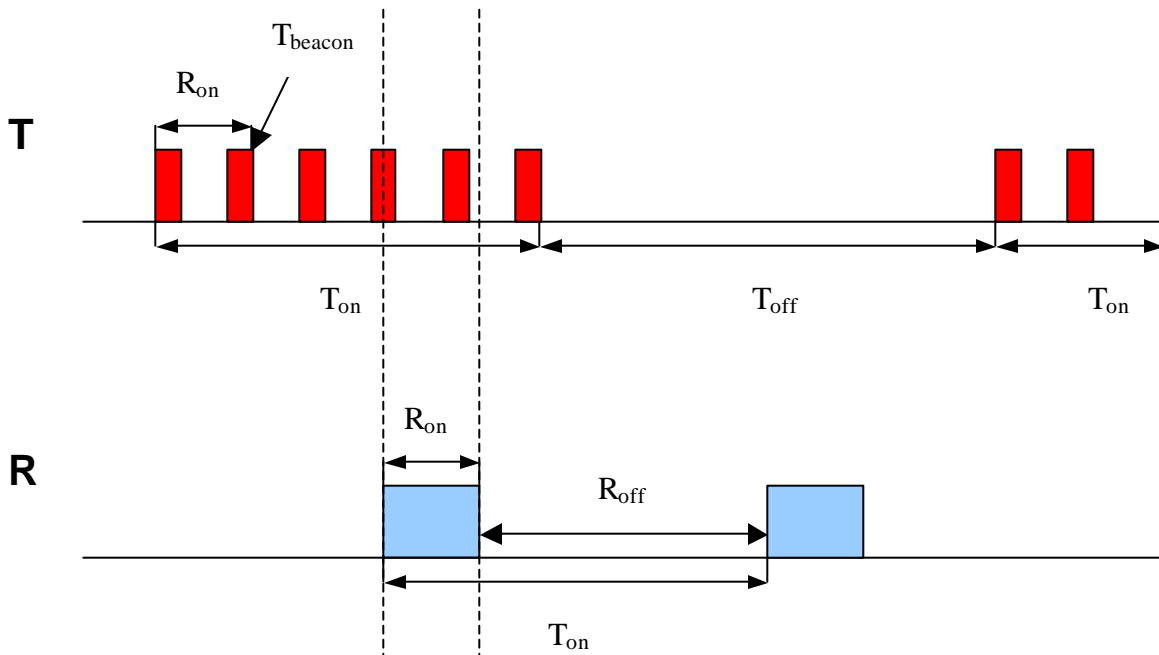
The idle state is the default state. When a node is in idle, it can do one of three things: enter the transmit state, enter the receive state, or stay in the idle state. In idle, the node does nothing, consuming very little power; it waits for an event to occur to trigger a state change.

In the receive state, a node can switch to idle when a timer expires, or if it wants to switch to the transmit state, it must first go to idle. This is mainly because receive and transmit are completely independent of each other (except that they cannot occur simultaneously) in this protocol. The receive state should not be concerned with the transmit state and vice versa. In other protocols the diagram would be different. In receive, the node actively “listens” for incoming data for a period of time before shutting off, and later it can listen again. For purposes of power conservation, which is a primary

goal of the project, this is a more efficacious design than if the receiver stayed on constantly.

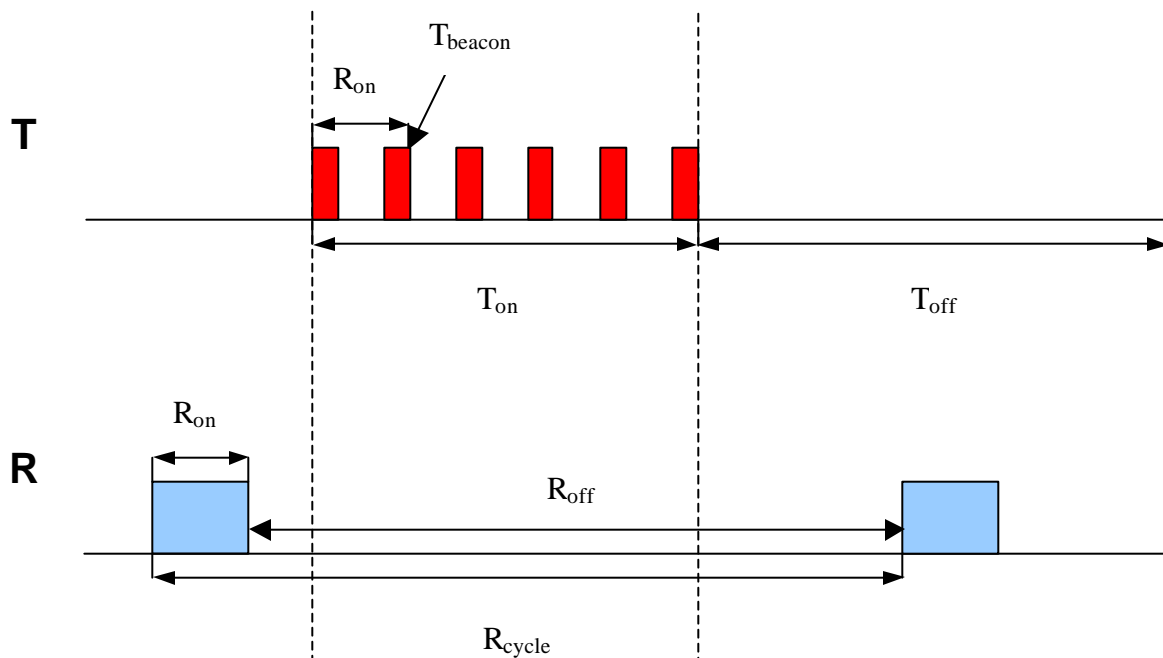
Similarly, when a node is in the transmit state, it can only switch to idle, whether that is its final destination or receive mode is its final destination. In transmit, the node sends out beacons that a neighboring node can receive if it is within “hearing” range. The beacons contain information, such as node identification, node status, and attributes like available power. Beacons are identical to one another and are sent out in groups with idle intervals in between, which conserves more power than if the node was transmitting constantly for a period of time. Transmit turns off and the node returns to idle after a group completes sending its beacons.

Illustrated in the figure below is a transmit train, denoted T, from node X and its reception, denoted R, by node Y. T_{beacon} represents transmit beacons, R_{on} and T_{on} represents the time that the node is in receive state and transmit state, respectively, R_{off} and T_{off} represents the time that the node is in idle, and T_{off} represents the time that the node is not in transmit mode.



The time that a node is listening for data, R_{on} , must be at least as long as two full transmit beacons in order to ensure that no data is lost if the node begins listening in the middle of a transmit beacon (see diagram on previous page). In this case, the first beacon will be incompletely received but the second beacon will be fully received. If R_{on} is shorter than two full beacons, then there is a possibility that two incomplete beacons will be received and data will be lost. R_{on} could be longer, so that if a complete received beacon has corrupt data, there will be other beacons to use as backup, but this results in higher power consumption.

In addition, T_{on} must be equal to or longer than the length of a receive cycle (R_{on} plus R_{off}), denoted R_{cycle} below. This is to guarantee that at least one full transmit beacon will fall within R_{on} , since transmit and receive occur asynchronously, as mentioned before. Otherwise, either the receiving node will completely miss the beacons or only an incomplete beacon will be received. The figure below shows what happens if T_{on} was shorter than R_{cycle} - data is lost because the transmit pulse train occurred when the receiver was off.



Lastly, R_{on} is shorter than T_{on} because only one beacon needs to be successfully received by the recipient node and receiving extraneous beacons would unnecessarily waste power.

IMPLEMENTATION

HandelC, a derivative of the software programming language C designed specifically to describe hardware, is used to implement the communication protocol. HandelC is a “super-subset” of ANSI C in that it includes constructs that are not part of ANSI C (e.g. `par`, which indicates that a group of statements should be executed in the same clock cycle) and does not include features that are difficult or inefficient to implement in hardware (e.g. dynamic memory). In addition, HandelC is executed in step with a clock such that each statement takes precisely one clock cycle: this helps organize the code in sync with hardware requirements.

The following snippet of code illustrates the use of HandelC construct `par`.

```
macro proc EndRxPkt(val)
{
    par
    {
        RxBusyFlag = ((val == Idle) ? FALSE:TRUE);
        State = val;
        Ecc_fail_cnt_ce = TRUE;
        WrCtrl = TRUE; // Turn off linebalance
    }
}
```

The assignment statements inside the `par` block are independent of each other, so that in hardware, it is more efficient to assign them on the same clock cycle. If the `State` variable could not be assigned until after the `RxBusyFlag` conditional statement was executed, then `State` should be assigned outside the `par` block, immediately after it. `Par` is one way HandelC allows developers to optimize hardware performance, making it a desirable tool to use in development.

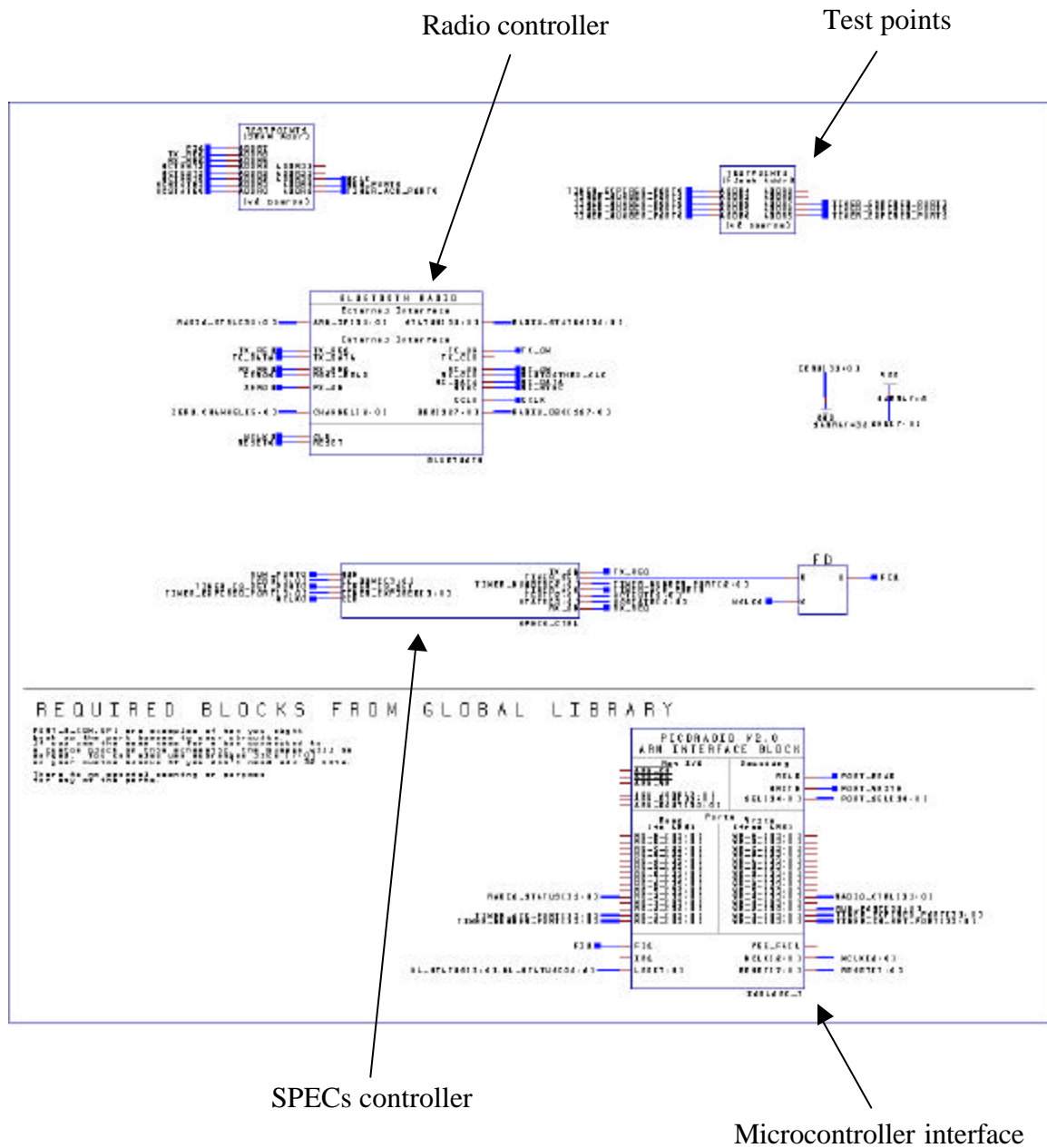
Code developed in HandelC is compiled into VHDL, which can then be run by ModelSim, a simulation tool.

For the SPECs project, timers will be used to keep track of the length of time a node should spend in transmit mode, receive mode, and idle mode. Timers are implemented in

the microcontroller. Timers are set using interrupts and timer expirations provide the “events” that cause state changes in HandelC. Each node has four separate timers: RxOnTimer, RxOffTimer, TxWaitTimer, and TxOffTimer. RxOnTimer indicates how long a node stays in active receive mode. RxOffTimer indicates how long a node stays in inactive receive mode – in other words, in a state in which the receiver power is off. TxWaitTimer keeps track of the time between beacons when a node is in the transmit mode. Finally, TxOffTimer keeps track of how long the transmitter power is turned off between transmit pulse sequences.

The three states that a node can be in: idle, receive, and transmit, are implemented in HandelC in a switch statement. Timers are set according to the state a node is in, and changes from one mode to another are made accordingly.

Xilinx Project Manager and ViewDraw are tools used to create an image of the resulting hardware that can be tested to ensure that the system performs satisfactorily under real-world conditions. The screenshot below shows the hardware components of a SPECs node.



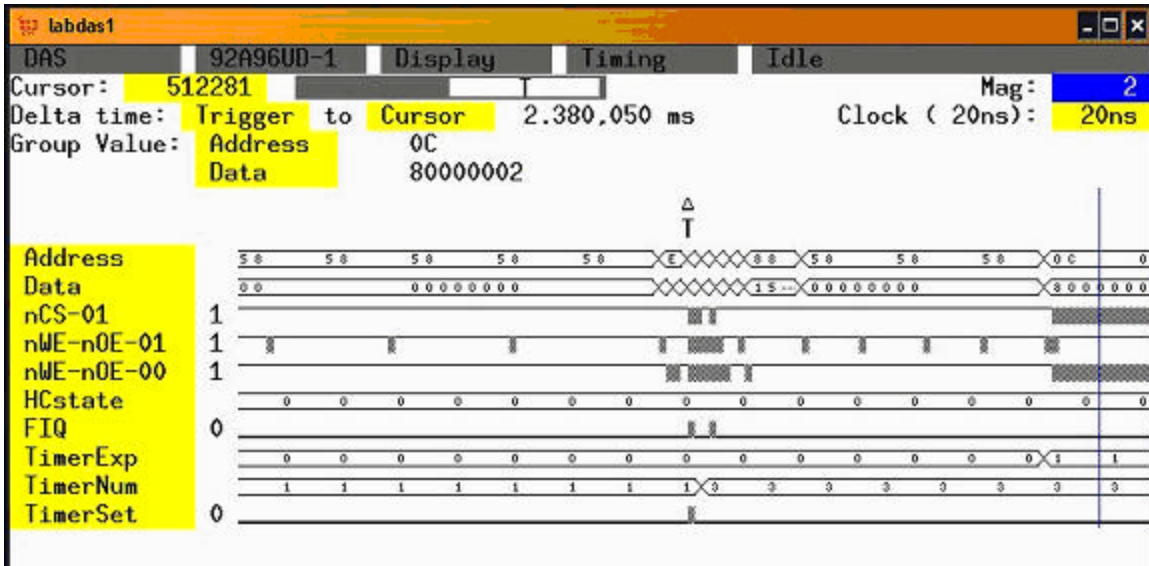
TESTING

Following simulation, an image is created and tested on the PicoRadio TestBed. The TestBed is a general-purpose embedded systems test platform designed for limited deployment in support of activities related to low-power wireless systems. The TestBed node consists of a digital board, a power board, a radio board, a sensor board, and a case. Below is a picture of a TestBed node.



RESULTS/CONCLUSION

The SPECS project is still in its infancy, however, strides are being made. The following screenshot from the PicoRadio TestBed shows results from setting the transmit timer and the receive timer and the timer expiration.



Future work may include adding functionality to avoid data corruption (i.e. error correction), handling multiple packets of information simultaneously, and optimizing transmit and receive times for power minimization.

ACKNOWLEDGEMENTS

Several wonderful people made my summer research experience possible. I would like to thank my faculty mentor, professor Jan Rabaey, for giving me the opportunity to work on the SPECs project. I would also like to thank my mentor, Fred Burghardt, for being most helpful and patient with me throughout the project; without him, I would not have learned as much as I did this summer. I am also thankful to DeLynn Bettencourt, who provided me with guidance and who is truly an inspiration. I am very grateful to Sheila Humphreys, Erika Tate, Marie Mayne, and everyone at SUPERB for organizing a great program and for caring so much about its participants. Last but not least, thank you to my fellow SUPERB students for making this summer memorable.

REFERENCES

“SPECs: Personal Pervasive Systems,” Lamming, Bohm, June 2003, IEEE Computer Society.

“The PicoRadio Test Bed,” Burghardt, Mellers, Rabaey, December 2002, Berkeley Wireless Research Center.

“Analytical Power Model for a Wireless Sensor Node,” Lin, June 2003, Berkeley Wireless Research Center.