CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

Voice-Recognition Controlled Sailboat

Speech-Recognition Control Aids Disabled Sailors

FEATURE ARTICLE

Mike Smith, Todd Turner, & Steve Alvey

For his engineering design project at the University of Calgary, Todd and his team created a prototype for a voice-recognition system that enables quadriplegic sailors to independently control a Martin 16 sailboat. Batten down the hatches for a storm of information about this practical and resourceful project.

The Canadian Engineering Accreditation Board (CEAB) now requires all graduating engineering students to have significant team design experience to prepare them for "real life" in industry. At the University of Calgary, to meet CEAB requirements, students have the opportunity of developing a semiacademic project, but many are taking an alternative approach that is more work, more fun, and far more real.

Students are finding their own projects from local industry or the community, which gives them a full year working with real projects and customers. Four students must act as project managers and engineers, work as a team, and follow a proper life-cycle approach to planning, designing, implementing, and testing. A public presentation of the work brings things together and serves as the final acceptance test.

After completing a 16-month internship at MCK Communications, Todd Turner returned for his final year and his design project. His team decided to create a prototype for a voice-recognition system for a Martin 16 sailboat. Steve Alvey, Todd's customer, and the Disabled Sailing Association of Alberta had modified the Martin 16, shown in Photo 1, so quadriplegic sailors could sail independently. The boat, funded by a grant from the Royal Bank of Canada, includes a 300-kg keel to prevent capsizing and a custom-designed seat to support the disabled sailor.

Quadriplegic sailors have already used this boat at regattas in Calgary and throughout Canada. The Autohelm, pictured in Photo 2, provides the necessary electronics to control the sails and helm using a joystick or a sip-and-puff mechanism. The controller also accepts commands

Photo 1—*The Martin 16 sailboat,* Royal Liberty, *funded by the Royal Bank of Canada, has a 300-kg keel and is designed for quadriplegic sailors. The sailor controls the sails and helm using a sip-and-puff mechanism. An able-bodied companion comes along for the ride. His hand can be seen near the back of the boat.*

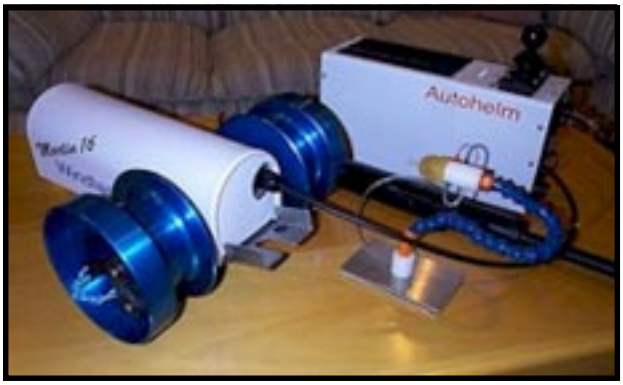

Photo 2—*A joystick or a sip-and-puff controller allows a disabled person to manage the Martin 16 Autohelm/Windlass System.*

over a serial link from the companion's remote control. A variety of devices remotely send commands or information to the controller or each other over this link. Todd's team needed to add a voice-control component to the serial link without disturbing its other functions.

The design project was divided into three technical sections:

- conditioning voice input
- developing a speech-recognition engine
- interfacing between the speech-recognition engine and the Martin 16 controller using a 9-bit UART

This article focuses on the interface. The other sections will be covered in subsequent papers.

THE HARDWARE

They chose the Analog Devices SHARC EZ-Kit Lite evaluation board for a number of reasons. This board's main processor, the ADSP-21061 SHARC DSP, operates at 40 MIPS, peaking at 120 MFLOPS. As you can see in Figure 1, the board includes an AD1847 stereo CODEC, which takes voice input via a microphone. A synchronous serial line to the SHARC processor connects to the CODEC, which is a sigma-delta oversampling converter that digitally filters the signal to avoid aliasing problems.

The development software includes both an optimizing C compiler and an assembler. Program development can be handled using the onboard kernel and a variety of basic debugging tools.

CONTROLLING THE MARTIN 16

With this device, sailors can remotely control the Martin 16, adjusting both the sail and helm, through the speech-recognition system. Because a high degree of control is needed, the voice-control system supports both discrete word commands and continuous sound commands.

For example, the command "RUDDER" places the system into rudder-control mode. The continuous sound "EEEEE" causes the Autohelm to change course by +1°. The course change continues to increase as long as the sailor holds the sound. Alternatively, if the sailor holds the sound "AAAAH," the Autohelm alters the course in –1° increments.

The system also greatly increases the confidence of the disabled sailors by adding the equivalent of a push-to-talk command for the safety.

PLUG AND SAY

The SeaTalk bus from Raytheon forms the serial connection between the speech-recognition system and the associated controller. Because the controller already supports the SeaTalk commands sent by the companion's safety remote, we could easily add the voice-control system to the existing system just by plugging the voice-control box into the SeaTalk port.

The serial commands for the SeaTalk bus are sent in an asynchronous format at 4800 bps with 8 data bits, 1 address/data flag, 1 start, and 1 stop bit. The first value in a command sets the address flag, so a device recognizes when the address refers to it. The remaining values define the command to be acted on.

This approach is referred to as a multidrop mode because the master controller can direct commands to multiple devices connected to the same serial link. Each slave device is programmed to recognize its own address and a global-broadcast address, enabling the master to efficiently command slaves simultaneously or individually. The transmitting device first sends a 9-bit value containing the receiver's address. When this address is recognized, the receiver acts on the data that follows.

CREATING A SOFTWARE UART

For this project, we need a universal asynchronous receiver transmitter (UART) to transmit data asynchronously. To avoid adding external devices to the design, many microcontrollers have on-chip UARTs. In fact, the SHARC processor has two on-chip time-division-mulitpleXing serial ports (known as SPORTs), which it uses to transfer data at up to 40 Mbps in numerically intensive real-time applications. However, these ports are synchronous, not asynchronous.

As you can see in Figure 1, the EZ-Kit Lite evaluation board has a standard 16550 PC UART. However, this UART is not easily accessible as it is

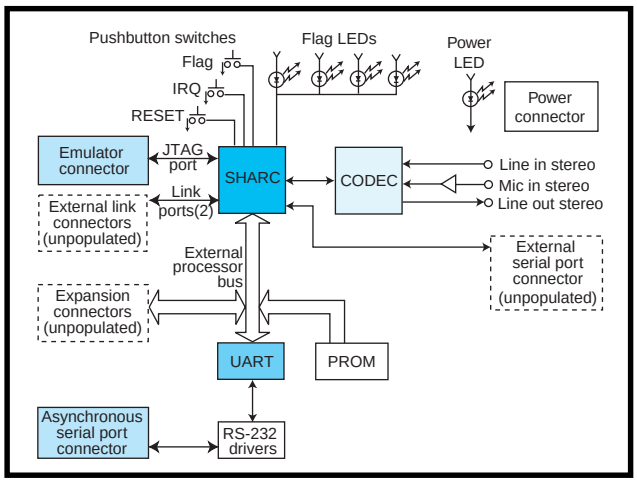

Figure 1—*The SHARC EZ-Kit Lite evaluation board uses an ADSP-21061 SHARC DSP as its main processor. The chip operates at 40 MIPS and peaks at 120 MFLOPS.*

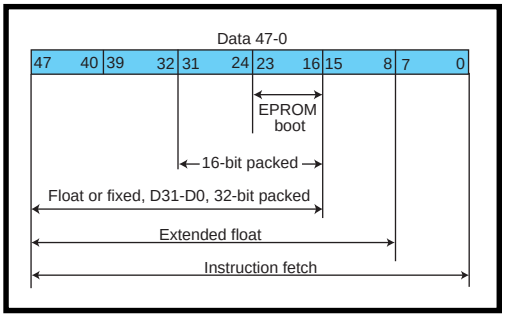

Figure 2—*The 48-bit SHARC external port handles a variety of data and instruction formats. Transmission of 8-bit data requires special positioning of the device on the SHARC data bus.*

intended for communications between the board kernel and diagnostic tools running on the host PC.

The SHARC user's manual suggests a way around the problem by using the pins on the evaluation board's expansion connector to clock out and receive bits from the multidrop serial line. The software UART uses the SHARC general-purpose I/O flag pins as well as the onboard programmable timer interrupt. The UART is emulated by detecting the start bit through programming and oversampling the received signal. The nine bits needed for communications are clocked by reading one of the flag pins. Similarly, transmission occurs by setting and clearing a second flag pin to clock out characters.

Although a software UART is feasible, the implementation would take a lot of time. More important, it was not clear that there would be enough room in the SHARC's internal memory for this and the speech-recognition software. Clearly, we had to develop a standard interface to an external UART chip.

HARDWARE INTERFACE TO THE SHARC

Interfacing between a UART and a DSP bus involves interesting contradictions. Although the UART sends and receives 9-bit data, the device has an external 8-bit wide data port. The ADSP-21061 48-bit external port handles a wide range of data formats, including an interface to an 8-bit device, such as a boot EPROM or UART chip (see Figure 2). The timing control signals are given in Figure 3.

The SHARC 21061(don't forget that SHARC stands for Super Harvard ARChitecture) has 1 MB of internal RAM that can be configured as independent program and data memory. To read and write to an off-chip external memory location, you activate the appropriate memory select (MS) line. When the MS signal is activated, an address is placed on the memory bus and a read (RD) or write (WR) signal is asserted.

This approach differs from other processors, which multiplex READ/WRITE* control signals. Some processors even require additional decoding logic, as there are no external memory-select or chip-enable signals generated directly by the processor.

After the MS and RD or WR signal asserts, the DSP places data on the bus in a write operation (the peripheral does it with a read). To match the slow speed of the external device, wait states can be programmed for the DSP to access a specific memory location. On the 21061, you can control the number of wait states through an external acknowledge signal (ACK), an internally programmed wait-state register, or a combination of the two.

CONNECTING TO THE DUART

The Exar XR-88C681A UART(actually a dual UART or DUART since it has two separate transceiver channels) handles the 9-bit serial transmission to the SeaTalk bus. Figure 4 shows the connections necessary to interface the 88C681 DUART to the 21061 external data port, address, and control signals. Although the *INTR interrupt signal output on the DUART can be programmed to assert a number of events, in this application, it is asserting *INTR on the reception of a character on channel A. This signal activates the DSP's IRQ0 interrupt service routine to process the character from the DUART.

CONFIGURING THE DSP FOR UART ACCESS

The SHARC processor has four external memory selects and three IRQ pins. On the EZ-Kit Lite, memory-select line MS3 and interrupt line IRQ0 are available to manipulate the DUART.

On microcontrollers, like the 68332, manipulating the chip-select lines can get complicated. Each chip-select line is mapped to a block of addresses anywhere in memory space and configured for 8- or 16-bit operations.

Things are a little more wasteful on the 21061. Here you choose between 48-bit access or no access. As a result, you manipulate bits to ensure that the 8-bit value is correctly placed for transmission and to clear unnecessary bits from received values.

There are four external memory banks, which correspond to the memory-select signals, on the SHARC starting with memory bank 0 at address 0x00400000. Memory bank 1 follows memory bank 0, memory bank 2 follows memory bank 1, and so on. Each bank is the same size and can be configured from 8-KB of 32-bit words to 256-MB of words. Although the base address of memory bank 0 is the same, the base address of the following memory banks varies based on the size of the preceding banks.

Only 16 registers are needed to control the UART. We innocently decided to code the setting for the external memory banks to their minimum 8-KB size, which makes the base address of the DUART on memory select line MS3 at 0x00406000. Only address lines A0 to A3 are explicit in the schematic as the line MS3 provides all the additional decoding necessary to specify

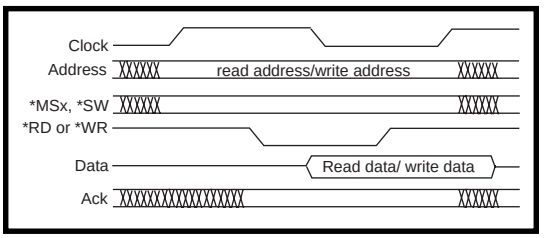

Figure 3—*Although the data bus of the SHARC external port is unusual, the timing of the control signals is conventional.*

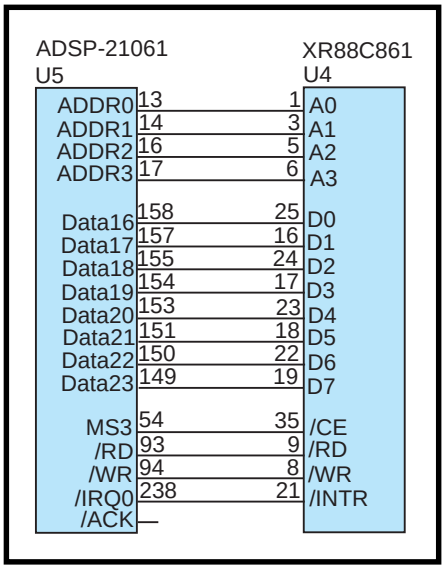

Figure 4—*Here you can see the necessary connections of the XR-88C681 DUART to the SHARC external data port.*

the memory locations.

What we didn't realize was that the DSP kernel's 16550 UART is mapped to memory-select line MS2. Changing the size of the memory banks should have changed the addresses for the 16550 registers, cutting the communications necessary for debugging.

However, somebody at Analog Devices was way ahead of us, and we got lucky! The kernel assumes that the average user code is written by the dim-witted. The kernel accounts for this by recalculating the effective address of the onboard 16550 before every access just in case the user code changes the bank size from the default value.

Because the DUART doesn't produce an ACK signal suitable for controlling the speed of memory accesses, you use the internal WAIT register to generate the six memory wait states to synchronize the 21061 and DUART

TALKING TO THE DUART

The C program *DuartAlive()* (see Listing 1) tests the bus connection. It first sets the 21k processor interface for 8-KB memory blocks and six wait states. If the values stored or received from the DUART Interrupt Vector Register (IVR) are the same, the DUART and SHARC are communicating.

Any register in the DUART can be accessed in C by casting a pointer to the address of the register. Because the EZ-SHARC g21k C compiler from Analog Devices treats characters as a 32-bit number, care must be taken to mask out the upper 24 bits when reading the DUART registers.

The registers must be treated as volatile char, which lets the compiler know that the stored char values may be changed by a process outside of the C program. By assigning this, we avoid the problem of the compiler incorrectly replacing multiple reads with a single read when the same memory location is read repeatedly without writes.

Because both mode registers have the same address, the protocol for accessing the DUART's mode registers makes Listings 2 and 3 a little confusing. After a read or write of a value to the one-mode register, the DUART automatically switches to allow access to the second-mode register. The function *ResetToMODE1( )* ensures that the correct register is accessed.

With the interface tested, the next step is to set up routines to initialize and control the DUART. Listing 2 shows you how to initialize both the 21k interface and the DUART.

Listing 3's routine verifies that it's possible to send commands from the SHARC to the multidrop bus for receipt by the Autohelm. The full protocol for handling bus collisions is not given.

TEST RUN

When the voice-recognition system was put to the test, the interface component did well. Todd was able to demonstrate that the voice-control system was communicating via the

Listing 1—*We used a dead-or-alive function test to check communications between the XR-88C681 UART and the ADSP-21061 DSP.*

```
#define SYSCON21K ((unsigned int *) 0x00)
                                  // 21K processor definitions
#define BANKS8K 0xFFFF0FFF
                                  // Set External Banks to 8K size
#define WAIT21K   ((unsigned int *) 0x02)
                                 // Wait state register
#define BANK3MASK  0xFFF07FFF
          // Need 6 internally generated wait states for bank 3
#define WAIT6BANK3  0x000B8000

#define DUARTBASE 0x00406000            // DUART definitions
#define IVR ((volatile char *) 0x0C+DUARTBASE)
         //  Memory mapped System Configuration Register
        // IVR register has 0x0C offset in the DUART

/***********************************
  Function: char DuartAlive(void)
  Descriptions:
       Returns ALIVE if the DSP can talk to DUART.
       Returns DEAD if no communication
***********************************/
#define ALIVE 1
#define DEAD 0
#define TESTVALUE 0xA5

char DuartAlive(void){
     *SYSCON21K &= BANKS8K;             // Set 21k bank size
     *WAIT21K &= BANK3MASK;             // Clear Bank 3 bits
     *WAIT21K |= WAIT6BANK3;            //  6 Waits to set

                                  // Try to say "hello" to DUART
*IVR = TESTVALUE;                       // Write a value to IVR
      if ((*IVR) & 0xFF) != TESTVALUE)  // and then bring back
      return DEAD;                      // Oops!
      else return ALIVE;                // Success!
}
```

Listing 2—*The XR-88C681A DUART goes into multidrop 9-bit operation and interrupt generation when a character is received.*

```
// Full DUART Register Definitions
#define MODE1  ((volatile char *) 0x00 + DUARTBASE)
         // Mode 1 Register – Channel A
#define MODE2  ((volatile char *) 0x00 + DUARTBASE)
         // Mode 2 Register – Channel A
#define TXDONE ((volatile char *) 0x01+ DUARTBASE)
         // Status Register – Channel A
#define CLOCK   ((volatile char *) 0x01+ DUARTBASE)
         // Clock Select Register - Channel A
#define CMD        ((volatile char *) 0x02+ DUARTBASE)
         // Command Register - Channel A
#define INTMASK  ((volatile char *) 0x05+ DUARTBASE)
         // Interrupt Mask Register
#define TXHOLD ((volatile char *)0x03+ DUARTBASE)
         // Transmit Hold Register
#define ResetToMODE1() *CMD = 0x10 | (*CMD & 0x0F)
         // Macro to force access to Mode Register 1

void Initialize21KplusDUART(void) {
         // 21k processor configuration
     *SYSCON21K &= BANKS8K;    // Set bank size
     *WAIT21K &= BANK3MASK;    // Clear Bank 3 bits
     *WAIT21K |= WAIT6BANK3;   //  6 Waits to set
         //  XR-88C681A Channel A Configuration
      *CLOCK = 0x99;           // 4800 Baud
      *CMD = 0xD5;             // TX and RX enabled
       ResetToMODE1();         // Access first-mode register
       *MODE1  = 0x1F;         // 9-bit char, multidrop mode
         // Interrupt asserted on RXRDY
       *MODE2 = 0x07;
       *INTMASK = 0x02;        // Activate interrupt
}
```

Listing 3—*Transmitting a value using multidrop protocol is a two-stage operation. First, the drop address must be sent, then the character. The protocol for handling bus contentions is not given.*

```
#define SENDMultiDropADDRESS 0x80
                // Transmit using multidrop protocol
#define WRITEREADY 0x04
#define CLOCKED 0x08

void MultiDropSend(char value, char dropaddress) {
     char temp;
     ResetToMODE1( ); // Configure DUART to send 9-bit address
     temp = *MODE1;
     ResetToMODE1( );
     *MODE1 = temp | SENDMultiDropADDRESS;
     while((TXDONE & WRITEREADY) != WRITEREADY)
                // Wait till DUART can send

      *TXHOLD = dropaddress;  /* Wait */ ;
     while((*TXDONE & CLOCKED) != CLOCKED)
                // Wait till value has been clocked out
                /* Wait */ ;

     ResetToMODE1( ); // Configure DUART to send 9-bit value
     temp = *MODE1;
     ResetToMODE1( );
     *MODE1 = temp &  ~ SENDMultiDropADDRESS;

     while((*TXDONE & WRITEREADY) != WRITEREADY)
                // Wait till DUART can send
                /* Wait */ ;
     * TXHOLD = value;
     while((*TXDONE & CLOCKED) != CLOCKED)
                // Wait till value has been clocked out
                /* Wait */ ;
}
```

serial link correctly with the signals coming in at the right time.

However, the voice-recognition system was only responding appropriately to approximately 20% of the commands it was given. Both the voice conditioning and speech-recognition engine components are back on the drawing board, waiting for another team of students to pick the project up. Certainly, the next crew will have an easier time as Analog Devices has donated its Visual DSP development simulator as well as an in-circuit emulator. Many of the problems of this first team were exacerbated by a lack of debugging tools.

In the meantime, Todd's interface is being put to work in other projects. It's currently playing a significant role in an MP3 compression/decompression.

ACKNOWLEDGEMENTS

We appreciate the help of Con Korirus of Analog Devices University Support and Jim Forsythe and Stan Parker of BBD Electronics, Calgary, the local distributor for Analog Devices, who donated a class set of SHARC EZ-Kit Lite demo boards, documentation, and in-circuit emulators. Rob Thompson at Raytheon Marine, U.K. was only an e-mail away with info on SeaTalk protocols. Thanks to the project design team—Stuart Bergen, Chris Leskiw, and Sunny Sandu—for their expertise and moral support. Thanks also to Ron Johnston, head of Electrical and Computer Engineering, for making the laboratory facilities available after hours. And, last but not least, without the help and patience of Alberta Disabled Sailing Association, this project would have sunk.

Mike Smith is a professor of Engineering at the University of Calgary. He specializes in microprocessor applications with a biomedical slant or any project he thinks might be interesting. You may reach Mike at mailto:smith@enel.ucalgary.ca.

Todd Turner graduated in May 1999 with a B.Sc. in Electrical Engi-

neering. He has returned to MCK Communications to continue his work with embedded systems. You may reach Todd at todd_turner@canada.com.

Steve Alvey continues to improve the Martin 16's control system. He'd be pleased to hear from others who would like to help get disabled sailors on the water. He can be contacted at salvey@arclite.com.

SOURCES

Analog Devices DSP applications
(800) ANALOGD
(617) 329-4700
Fax: (617) 329-1241
www.analogdevices.com

Disabled Sailing Association of Alberta
www.inventmgmnt.ab.ca/sipandpuff/

Exar XR-88C681A DUART reference manual
www.exar.com/products/xr88c681.html

Raytheon Marine
www.autohelm.com