Programming the HP 48 Calculator to Control the LabWorks Interface

Matthew E. Morgan,[†*] **and John R. Amend**[‡]

Department of Chemistry, U.S. Air Force Academy, Colorado Springs, Colorado, morganme.dfc.usafa@usafa.af.mil and Department of Chemistry and Biochemistry, Montana State University-Bozeman, Bozeman, MT 59717

Abstract: While the balanced chemical equations for a multireaction system are generally not unique, the minimum number of independent equations, *R*, is a characteristic property of the system. Deleting one nonspectator species from the system leads to a system with *R* reduced by one. In this way each system can be reduced to a single-reaction system and ultimately to a no-reaction system. The least number of chemical species that can be deleted to obtain a no-reaction system equals R. Every multireaction system, therefore, can be reduced to a number of single-reaction equations which can be balanced by any one of the standard techniques. Some examples are given where balancing by inspection is employed.

The LabWorks learning system is an effective, inexpensive combination of hardware and software that enables students to obtain and analyze experimental data in general chemistry laboratories [1, 2].The LabWorks interface was designed to be controlled by a personal computer (PC), but it has been shown that a handheld graphing calculator can also control the LabWorks interface [3].This article describes the process of programming the calculator to control the LabWorks interface.

HP 48G and HP 48GX calculators have the ability to communicate with the LabWorks II interface via serial link. Like PCs, the HP 48 can send commands and receive LabWorks-acquired data. The data can also be stored, processed, and displayed in both graphical and text formats. The calculators occupy much less desk space than PCs and, at $100 per unit, are much less expensive to purchase and maintain.

Establishing Communications

Hewlett Packard manufactures a cable that enables its HP 48 calculators to connect to PCs via a 9-pin RS-232 serial port. This cable can be used to attach the calculator to the LabWorks II interface. Because the interface is also designed to communicate with a PC, a null-modem adapter is necessary to allow the calculator and interface to communicate with each other.

After connecting the calculator and LabWorks interface, it is necessary to confirm that the calculator and interface can communicate. SCI Technologies assisted with this part of the project by supplying the machine-level command codes for the LabWorks interface. Using this information, the calculator was programmed to send a series of commands to the interface, and then receive incoming data. The communications protocol for the LabWorks II interface is: 9600 baud, 0 parity bit, no stop bit.

All data communicated between the calculator and the interface are in the form of ASCII characters. ASCII stands for American Standard Code for Information Interchange, a binary number-to-letter conversion code. In this system, upper- and lower-case letters, numerals, and symbols all have assigned binary values [4]. Both commands and data are exchanged between the interface and the calculator using this encoding scheme.

The HP 48 uses two commands, XMIT and SRECV, to transmit and receive data [5]. When communication is successful, a 1 is displayed in the calculator window after both XMIT and SRECV. Another HP 48 command, DROP, removes these numbers from the HP 48 display. A very simple program in which the calculator checks for the presence of a working interface is shown in HP 48 Program 1 (Figure 1). All HP 48 programs discussed in this article are adapted from computer code that is copyrighted by SCI Technologies.

When the LabWorks interface receives an ASCII B character, it returns an ASCII U, and this character is shown in the calculator display.

Making Digital Measurements

Digital measurements are the simplest types of measurements the LabWorks interface makes because these data do not use the interface's analog-to-digital converter (ADC). Digital counter and time data are examples of digital measurements. The LabWorks counter is a 16-bit device and its data are sent using two characters. In this case, each byte of data is converted to a binary number, the two numbers are combined, and the result is converted to an integer. HP 48 Program 2 (Figure 2) shows how the HP 48 obtains a value of 15,729 from the LabWorks interface counter.

The first line of the program sets the maximum size of the binary numbers to 16 bits, and no value is returned to acknowledge this action. In lines 2 and 3, the calculator sends a "C" character to request a counter reading, and receives the first data character. The DROP command after each XMIT and SRECV removes the successful communications acknowledgement from the HP 48's data stack (vide infra). Line 4 converts the first data character to a decimal number, and line 5 converts the decimal to binary. It would be more efficient to convert the character directly to binary, but that is

[*] Address correspondence to this author.
[†] U.S. Air Force Academy
[‡] Montana State University

Line no.*	HP 48 Commands	HP 48 Displayed Result
1	"B" XMIT DROP	
2	1 SRECV DROP	"U"

*HP 48 programs do not use numbered lines. The line numbers listed here are for reference purposes only.

Figure 1. HP 48 program 1. Check for working labworks interface.

Line no.	HP 48 Commands	HP 48 Displayed Result
1	16 STWS	
2	"C" XMIT DROP	
3	1 SRECV DROP	"q"
4	NUM	113
5	R→ B	01110001
6	1 SRECV DROP	"="
7	NUM	61
8	R→ B	00111101
9	1 8 START SL NEXT	0011110100000000
10	OR	0011110101110001
11	B → R	15,729

Figure 2. HP 48 program 2. reading the labworks counter.

not presently possible. The second data character is received in line 6, and it is converted to a decimal, and then to a binary number, in lines 7 and 8. Line 9 shifts the second binary number eight digits to the left.

Line 10 combines the binary number generated in line 5 with the left-shifted binary in line 9. The result is a 16-bit binary number, which is then converted to the decimal counter value. Finally, line 11 converts the binary number to an integer.

Reading the Analog-to-Digital Converter

Obtaining data from the LabWorks interface's ADC is more complicated than reading digital information. The ADC takes an analog signal, such as from a thermistor, and converts it to a 12-bit binary number. The HP 48, like the counter, receives this number in two bytes, but only 12 bits of the incoming data are significant. The HP 48 uses a subroutine called "ADC" that combines two data bytes that the interface has obtained and sent to the calculator. This subroutine is part of any analog data acquisition.

The ADC subroutine requires that two numbers be placed on the HP 48's number stack. The stack is memory that holds numbers waiting for mathematical operations. The size of the stack is limited only by calculator memory, but only the last four numbers on the stack are displayed. HP 48 Program 3 (Figure 3) starts with two data bytes obtained from an analog sensor, such as a pH probe, and shows how the signal from the probe is converted to a digital value. Note: The two values on the HP 48 stack were ASCII characters sent from the interface that have already been converted to integers.

The first line of the program sets the maximum size of the binary numbers to 12 bits to accommodate the 12-bit ADC. No value is returned to the calculator display to acknowledge this action. In lines 2 and 3 the number in level 1 of the stack is converted to a binary number and shifted left four bits. Line 4 switches stack locations, temporarily storing the shifted binary value, and allowing the 143 value to be manipulated. Lines 5 and 6 convert 143 to a binary value, then shift it four bits to the right. Four bits are lost during this right shift, but the lost data are meaningless because the ADC has 12-bit, not 16-bit, resolution. Finally, line 7 combines the two values on the stack using a binary OR command.

Twos-Complement Arithmetic

This conversion program returns integer values ranging from –2048 to +2047. The HP 48 command NEG in line 8 takes the twos-complement of a binary number. The values obtained by the interface's analog-to-digital converter use twos-complement binary arithmetic. This style of binary-to-decimal conversion allows both positive and negative values to be converted. The most significant binary digit is a sign bit: a 1 for a negative number, and a 0 for a positive number. All other bits in the word give positive integer values [6]. For example, a binary 1011 is equivalent to a decimal 11 (8 + 0 + 2 + 1), but using twos-complement, this binary number is –5 (–8 + 0 + 2 + 1). Finally, line 9 in HP 48 Program 3 converts this binary number to a real number.

Interface Calibration Constants

The values obtained from the ADC need to be converted again in order to be meaningful. For example, an electrical current reading obtained from the ADC is converted to a digital form, but these digital data are not accurate until operated on by a calibration routine.

The necessary calibration constants are stored in the LabWorks interface in a three-dimensional array. The calibration values reside in a 51 × 8 array; four redundant copies of this array exist. The array is stored in the interface's erasable, programmable, read-only memory (EPROM). When the computer version of LabWorks begins, this array is copied into computer memory. This process was extremely difficult to duplicate using the HP 48. Calibration constants were therefore empirically derived and stored as variables in calculator memory.

Using the HP 48 to Read Electric Current

When the LabWorks interface reads electric current, its input differs from other devices using the ADC. This device sends a different set of values to the HP 48 than do the other ADC devices in the interface. Because the electrical current amplifier is autoranging, three data bytes are sent to the HP 48, rather than the normal two. The third data byte is the current range code that tells the calculator the power of ten that needs to be applied to the data.

The command to read current from the interface is a single byte with bit 8 set to a value of 1. Bit 0 selects whether to read I1 or I2, and bit 1 selects fast or average ADC reading. The values of the five intermediate bits are not read by the interface. Table 1 summarizes the different commands used to read current from the LabWorks interface.

The subroutine shown in HP 48 Program 4 (Figure 4), called I1SUB, is used with all HP 48 programs that need data from the LabWorks interface's I1 input. The subroutine calls ADC to convert the interface's analog-to-digital converter output to a real number. The SND command is a subroutine that combines CHR, XMIT, and DROP commands into a single

Line no.	HP 48 Commands	HP 48 Stack 1	HP 48 Stack 2
1	12 STWS	234	143
2	R→ B	# 11101010b	143
3	1 4 START SL NEXT	# 111010100000b	143
4	SWAP	143	# 111010100000b
5	R→ B	# 10001111b	# 111010100000b
6	1 4 START SR NEXT	# 1000b	# 111010100000b
7	OR	# 111010101000b	
8	NEG	# 101011000	
9	B → R	344	

Figure 3. HP 48 program 3. adc conversion subroutine.

Line no.	HP 48 Commands	HP 48 Stack 1	HP 48 Stack 2	HP 48 Stack 3
1	129 SND			
2	1 3 START RCV NEXT	24	234	143
3	3 ROLLD	234	143	24
4	ADC	344	24	
5	→ x y			
6	CASE			
7	‘x = = 0’ THEN y ‘I’ STO END			
8	‘x = = 8’ THEN y 10 / ‘I’ STO END			
9	‘x = = 16’ THEN y 100 / ‘I’ STO END			
10	‘x = = 24’ THEN y 1000 / ‘I’ STO END	0.344		
11	END I 1.221 * ‘I’ STO I	0.420		

Figure 4. HP 48 program 4. Subroutine to read current from i1.

Table 1. LabWorks Electric Current Reading Commands

Decimal Value	Binary Value	Input	ADC mode
128	10000000	I 1	fast
129	10000001	I 1	averaged
130	10000010	I 2	fast
131	10000011	I 2	averaged

function. Likewise, RCV combines 1, SRECV, DROP, and NUM.

Line 1 sends the command to read from one of the current inputs. This command reads the I1 input in the averaged mode. Line 2 receives the two data bytes and the current-range value from the interface. The possible current-range values sent by the interface are 0, 8, 16, and 24 for I1, and 0, 32, 64, and 96 for I2.

Line 3 moves the current-range byte from stack level 1 to stack level 3. The ADC subroutine is invoked in line 4 to convert the two data bytes, which are now on stack levels 1 and 2, into a real number. Line 5 saves the converted ADC number and the current range value into local variables *x* and *y*.

Lines 6 through 10 change the ADC number depending on the value of the current-range value. The CASE command executes only the necessary line and ignores the rest. The multiplicative factor in line 11 is an empirical constant to give the current a maximum value of 2048 microamperes.

Advanced Experiment Control: Setting and Incrementing the DAC

Along with acquiring data from the LabWorks interface, the HP 48 can also set and increment voltage values to the interface’s digital-to-analog converter (DAC). The DAC functions as an adjustable voltage source. The DAC can be set or changed in 1-mV increments from −2048 to +2047 mV.

The process of sending values to the interface’s 12-bit DAC is fairly complicated. A command byte is sent to the interface, along with three data bytes. Four bits (one nibble) from each data byte are incorporated into a single 12-bit value. The HP 48 accomplishes this process by taking an integer, converting it to a binary number, dividing it into three parts, and sending these parts to the interface. The program DAC1 (HP 48 Program 5) (Figure 5) shows how the HP 48 sends a value of 1000 mV to DAC1.

Line 1 of this program takes an integer and sends it to a local variable, *m*. The value of the integer determines the value at which the DAC will be set. Incrementing or decrementing this integer will then increment or decrement the DAC. The relationship between *m* and the value of the DAC has been empirically determined to follow the equation

$$\text{DAC value (in mV)} = 1.2242\,m - 2506.5$$

Line no.	HP 48 Commands	HP 48 Stack 1	HP 48 Stack 2
1	2864 → m		
2	49 SND		
3	m R→ B	#101100101101b	
4	15 R→ B	#1111b	#101100101101b
5	AND	#1101b	
6	B→ R SND		
7	m R→ B	#101100101101b	
8	1 4 START SR NEXT	#10110010b	
9	15 R→ B	#1111b	#10110010b
10	AND	#0010b	
11	B→ R SND		
12	m R→ B	#101100101101b	
13	1 8 START SR NEXT	#1011b	
14	B→ R SND		

Figure 5. HP 48 program 5. Sending 1000 mV to dac1.

Using this equation, the DAC could be set to 0 mV by using an m value equal to 2047, or set to −1000 mV by making m equal to 1230.6.

Line 2 sends a decimal 49 to the interface, which is the equivalent to an ASCII character 1, which is the interface command to set DAC1. The next three values sent to the interface will determine the value of the DAC.

Lines 3 through 5 recall the variable *m*, convert it to a binary number, and isolate the last four bits using a bitwise AND function. This binary number is converted to a real number and sent to the interface in line 6. Lines 7 through 11 perform the same function, with the addition of shifting the binary number four places to the right and then performing the bitwise AND. This bit-shift isolates the center four bits so they may be converted to a real number and sent to the interface. Finally, lines 12 through 14 send the last four bits by shifting eight bits right, converting to a real number, and then sending this value to the interface. When the interface receives three data values after the 49, it sets the DAC to the appropriate value.

Conclusions and Future Work

Programming the HP 48 is not a simple task. Students can use a preprogrammed calculator to store and manipulate chemistry data, but incorporating student experiment design is presently beyond this system. Future effort can be devoted toward allowing students to choose experiment input, by way of a simple menu-driven interface. This system will become much more popular if the more widely used Texas Instruments calculators could be adapted to control the LabWorks interface. The main obstacle for TI-based control is this calculator's proprietary communications protocol. Other devices such as handheld computers or palm-size PCs might also be programmed to control the LabWorks interface.

References and Notes

1. Amend, J. R.; Furstenau, R. P. *Acad. Comp*. **1989**, *4*(3), 20.
2. Amend, J. R.; Furstenau, R. P.; Tuicker, K. *J. Chem. Educ.* **1990**, *67*(7), 857.
3. Morgan, M. E.; Amend, J. R. *Chem Educator* **1998**, *3*(*5*) S1430-4171 (98) 05253-6. Avail. URL: http://journals.springer-ny.com/chedr.
4. Foster, L. S. *C by Discovery,* 2nd ed.; Scott/Jones: El Granada, CA, 1994; p 6.
5. *HP 48G Series Advanced User's Reference Manual*; Hewlett-Packard Corp.: Corvallis, OR, 1994.
6. Bronson, G.; Menconi, S. *A First Book of C;* West: Saint Paul, 1988.