1.3 - GETTING STARTED

The operation of a Basic language controller is an easy concept to understand. The Basic program makes logical decisions based upon certain conditions and then sets or clears bits in a bit image table. This image table is mapped onto the inputs and outputs connected to the controller. The image table is actually only mapped to the IO when the LIO or EXIO statements are executed. The status of bits relating to outputs are then transferred to the actual output devices and the state of any inputs are transferred to the bits related to inputs. The effect of the LIO and EXIO can be seen as a snapshot of the real world connections. The Basic control program is a continually running loop. All the logical decisions are made, and then the transfer is made to the IO. After which the program loops back to the start. The Control Basic language contains special control statements which allow easy bit manipulation. There are statements that examine bit status (high or low) and statements to change bit status. A traditional Basic program would have to read a 8 bit value and then mask out the undesired bits where as the Control Basic just reads the actual bit. An example is a follows

```
10 REM read memory mapped input port and mask value
20 IF XBY(0A000H).AND.128 THEN PRINT "bit 7 ON"

10 REM Now Control Basic
20 XIH 031 : PRINT "bit 7 ON"
```

The Control Basics version of reading the input is quite simple to follow and takes a lot less space and execution time. The statement XIH examines bit 031 and if high or ON the program execution continues past the statement much like the IF THEN. The colon(:) is used to separate the statements. This type of bit testing allows multiple bits to be tested to perform AND logic on 1 line. Examples of more complex logic can be found further in the text. It can be seen that the control basic allows traditional basic to be used right along with the control statements. An example of a simple Control Basic program is as follows:

```
10 XIH 016 : OTL 000
20 XIL 016 : OTU 000
30 LIO : GOTO 10
```

In the above example the first input is examined and if it is ON then the first relay (000) is set ON. If the input goes low then the output goes low. The actual bit (000) is set ON or OFF immediately after execution but the transfer to the real world only occurs after the LIO statement is executed. **The LIO is only required once in the main program loop**. This example is very simple, more complex examples are used throughout the manual and display the powerful features of the Control Basic language. It is recommended that the reader pay close attention to how each statement operates and its requirements. A good understanding of the statements operation yields an efficient program!

The BASIC program is always maintained by the lithium battery. There are two ways to clear the memory space. The first is to type NEW at the >(prompt) with the MEM PROTECT jumper ON. The second is to RESET the controller with the MEM PROTECT jumper ON.

Any communications program set for 9600 8N1 (PROCOMM for example) or the simple BACCOM program can be used to communicate with the BAC controller. The BACCOM program waits for the prompt every time it uploads files. If any other communications program is used the user must ensure sufficient line pacing to allow the interpreter to tokenize the line.

1.5 - Differences between BAC552(X) and BAC552ES

Model We Have

The new BAC552ES appears to be physically the same as the BAC552(X) but there has been many significant advancements. The ES controller has 128K of SRAM and 128K of FLASH memory. In order to support reading and writing to the additional SRAM and FLASH memory a statement HIMEM has been added to re-direct the LD@, ST@, XBY and MOVE statements.

A secondary watchdog timer has been implemented on the LTC/MAX691 device. This watchdog has a 1.6 second time out and it is toggled by the 012 LED output. The user can disable this secondary watchdog by removing the ENABLE jumper between the 691 device and the lithium battery. The initialization routines toggle the 012 LED up to the execution of the first BASIC file line, it is up to the programmer to continue toggling this 012 LED within the 1.6 second time if this timer is enabled.

Two more serial ports have been added and changes have been made to the COMINT, INPUT, PRINT, LIST, XSB(E), BAUD, RIO and SIO statements. The auxiliary RS232 serial port is configured as a DTE with RTS/CTS flow control option. The RTS/CTS is selected by removing the RTS/CTS EN jumper before a reset condition. The last port is configured as an RS485. All three ports (console, RS232 DTE and RS485) have their own 256 byte input and output ring buffers.
The AUX. DTE port can communicate from 300-38400 baud and the RS485 port range is 300-9600 baud. All ports default to 9600 baud on RESET. A jumper called CONSOLE BAUD can force the console baud rate to 19200 baud.

A MEM SIZE jumper selects the BASIC file memory size. With the jumper ON the memory is approximately 58K and with the jumper OFF the memory size is 26K. If the ES controller is used for data logging applications the user has access to 96K of SRAM if the smaller memory size is selected.

data logging

The ES controller uses a different line look up scheme than on the old BAC552s. At RUN time a routine scans the BASIC file looking for all statements with line numbers such as GOTOs and GOSUBs. A file is created in RAM with the line number and the actual memory address for the line. When one of these statements are executed the actual address is looked up instead of scanning through every line from the start of the program as the old BAC552s did. There are now significant performance improvements to interrupt response and overall file scan times on large programs. The line file space is limited to 500 statements with destination line numbers.

The LED outputs 013,014,015 are now 5 volt logic compatible or open collector. The limiting resistors on these outputs are 470 ohms tied to 5 volts.

A 256 byte serial EEPROM has been added. Users can install up to a 2K byte device in the same socket.

ROM CONFIGuration jumpers have been added to allow the FLASH device to replace the BAC552ES ROM. This is to support a future BASIC compiler or users who desire to use the boards I/O, but write the application in another language such as C. SYLVA will provide on request the ASM drivers for the controller's I/O.

The ES controller will now only print a CR LF when it auto runs from a RESET condition whereas the old BAC552 would print to the console port the program transfer, checksum verification and program running.

The analog channels are now only read when the ADC(0-7) is executed. The old BAC552s updated all channels in the background at a very slow rate. The users can now determine their own analog update rates and a new 8 bit result is available.

An additional MOVE statement has been added for moving data blocks from the BASIC file memory space. This was added for users who want to transfer data from serial buffers or manipulate stored strings. The MOVE is move time efficient than the traditional FOR NEXT loop to perform a memory move function and HIMEM also redirects the **destination** of the MOVE.

Commands

- RUN
- CONT
- LIST
- NEW
- HEXDUMP
- BST
- SAVE
- LOAD
- ERASEL
- ERASEH
- RENUM
- EHMEM

DESCRIPTION OF COMMANDS

RUN(CR)

After RUN(CR) is typed at the command prompt the program execution begins at the first line number. All variables are cleared and a **CTRL-C** is required to break program execution.
If the user has invoked the **COMINT** and turned the communication interrupt ON (CIC 1) then the user must supply a custom break routine. An example of a custom break routine can be found in the description of the **COMINT** statement.

CONT(CR)

The **CONT** command allow the user to continue execution of the program if it was stopped by a **CTRL-C** or a **STOP** command. If the program was modified and then the **CONT** was typed then a **CANNOT CONTINUE** error would be generated.

LIST(CR), LIST%(CR)

The **LIST** command types the program to the console device. Typing a **CTRL-C** will stop the listing at any time. There are some variations to the **LIST** command and they are as follows:

LIST (CR) - lists the whole program start to finish
LIST [ln number] (CR) - lists that line only
LIST [ln num]-[ln num] (CR) - lists that range of lines
LIST % - re-directs the list output to the RS232 DTE port
Note: The % switch works with all variations of list

NEW (CR)

The NEW command erases the program memory if the memory protect jumper is ON. If NEW is typed and the jumper is OFF then a MEMORY PROTECTION error will occur. New also clears all system variables.

HEXDUMP (CR)

The HEXDUMP command will generate a INTEL HEX FILE of the current program in user memory. This file is used to program an EPROM for permanent program storage. After a RESET condition BASIC will examine the EPROM socket for a valid program and if one is found it will transfer that program to the memory, verify the program checksum and then run that program. If the checksum failed then the BASIC will display that error and stop at the command prompt. The EPROM program is always loaded to allow for application updates without connecting a console device.

BST [integer](CR)

The command BST [integer] where integer equals 00 to 63 returns the status of the bits in the image table for the word represented by the integer value. This is useful for checking the status of any bits in the image table.

SAVE (CR)

The command SAVE copies the BASIC program file from static memory (SRAM) to the FLASH memory device. The FLASH memory is first erased if it contains an existing program, if the memory is blank then the file is copied immediately. The status of these operations are echoed to the console device.
IMPORTANT: The user must remove the MC WDOG jumper and the AUX WDOG jumper. The SAVE, ERASE, and RENUM routines do not service the watchdogs timers. Leaving the jumpers ON will crash the

operation of these functions!

LOAD (CR)

The command LOAD reads the program from FLASH memory and copies it to SRAM. The transfer from FLASH to SRAM is always done after a reset if the RUN MODE jumper is ON and there is a valid program in FLASH memory.

ERASEL (CR)

The command ERASEL, erases the lower 64K block of FLASH memory. This is automatically done if you use the SAVE command.

ERASEH0, ERASEH1, ERASEH2, or ERASEH3 (CR)

The command ERASEH,[0-3], selectively erases the four 16K byte sectors of upper FLASH memory. ERASEH,0 erases the first sector and ERASEH,3 erases the last sector.

RENUM [starting line #],[line spacing] (CR)

The command RENUM, re-numbers the user BASIC file in SRAM. The user must specify the starting line number and the value of the line spacing. The RENUM, re-numbers the entire program, you cannot re-number a range of lines! The staring line number is a valid integer (1-65535) and the line spacing is a valid 8 bit integer (1-255).

You must ensure that you insert 2 REMark lines, 1 at the very beginning of the program and one at the end. The RENUM requires the very start and very end of the program to have lines which do not involve the calculation of re-numbered lines. REMs are recommended.

Example:

```
1          REM start
.
User program
.
65535      REM end
```

EHMEM(CR)

The command EHMEM erases the upper 64K block of static RAM to all zeros.

2.1 - RUN(CR)

After RUN(CR) is typed at the command prompt the program execution begins at the first line number. All variables are cleared and a **CTRL-C** is required to break program execution.
If the user has invoked the **COMINT** and turned the communication interrupt **ON** (CIC 1) then the user must supply a custom break routine. An example of a custom break routine can be found in the description of the **COMINT** statement.

2.2 - CONT(CR)

The **CONT** command allow the user to continue execution of the program if it was stopped by a **CTRL-C** or a **STOP** command. If the program was modified and then the **CONT** was typed then a **CANNOT CONTINUE** error would be generated.

2.3 - LIST(CR), LIST%(CR)

The **LIST** command types the program to the console device. Typing a **CTRL-C** will stop the listing at any time. There are some variations to the **LIST** command and they are as follows:

LIST (CR) - lists the whole program start to finish
LIST [ln number] (CR) - lists that line only
LIST [ln num]-[ln num] (CR) - lists that range of lines
LIST % - re-directs the list output to the RS232 DTE port
Note: The % switch works with all variations of list

2.4 - NEW (CR)

The **NEW** command erases the program memory if the memory protect jumper is **ON**. If **NEW** is typed and the jumper is **OFF** then a **MEMORY PROTECTION** error will occur. New also clears all system variables.

2.5 - HEXDUMP (CR)

The HEXDUMP command will generate a INTEL HEX FILE of the current program in user memory. This file is used to program an EPROM for permanent program storage. After a RESET condition BASIC will examine the EPROM socket for a valid program and if one is found it will transfer that program to the memory, verify the program checksum and then run that program. If the checksum failed then the BASIC will display that error and stop at the command prompt. The EPROM program is always loaded to allow for application updates without connecting a console device.

2.6 - BST [integer](CR)

The command BST [integer] where integer equals 00 to 63 returns the status of the bits in the image table for the word represented by the integer value. This is useful for checking the status of any bits in the image table.

2.7 - SAVE (CR)

The command SAVE copies the BASIC program file from static memory (SRAM) to the FLASH memory device. The FLASH memory is first erased if it contains an existing program, if the memory is blank then the file is copied immediately. The status of these operations are echoed to the console device.
IMPORTANT: The user must remove the MC WDOG jumper and the AUX WDOG jumper. The SAVE, ERASE, and RENUM routines do not service the watchdogs timers. Leaving the jumpers ON will crash the operation of these functions!

2.8 - LOAD (CR)

The command LOAD reads the program from FLASH memory and copies it to SRAM. The transfer from FLASH to SRAM is always done after a reset if the RUN MODE jumper is ON and there is a valid program in FLASH memory.

2.9 - ERASEL (CR)

The command ERASEL, erases the lower 64K block of FLASH memory. This is automatically done if you use the SAVE command.

2.10 - ERASEH0, ERASEH1, ERASEH2, or ERASEH3 (CR)

The command ERASEH,[0-3], selectively erases the four 16K byte sectors of upper FLASH memory. ERASEH,0 erases the first sector and ERASEH,3 erases the last sector.

2.11 - RENUM [starting line #],[line spacing] (CR)

The command RENUM, re-numbers the user BASIC file in SRAM. The user must specify the starting line number and the value of the line spacing. The RENUM, re-numbers the entire program, you cannot re-number a range of lines! The staring line number is a valid integer (1-65535) and the line spacing is a valid 8 bit integer (1-255).

You must ensure that you insert 2 REMark lines, 1 at the very beginning of the program and one at the end. The RENUM requires the very start and very end of the program to have lines which do not involve the calculation of re-numbered lines. REMs are recommended.

Example:

```
1 REM start
User program
65535     REM end
```

2.12 - EHMEM(CR)

The command EHMEM erases the upper 64K block of static RAM to all zeros.