8.1 - RELATIONAL OPERATORS

(=) - Equal
(<) - Less Than
(>) - Greater Than
(<>) - Not Equal
(>=) - Greater Than or Equal
(<=) - Less Than or Equal

The relational operators are used to test a condition. The test results in a true or false. These tests are performed on the argument stack and a result of the test can actually be printed. If the result is true then the A stack = 65535 and if the result is false the A stack = 0. With the argument stack holding the result of the test, multiple relational expressions can be tested on one line. When testing with logical operators and relational operators in the same line the relational operators have a higher precedence!

EXAMPLE:

```
>PRINT 1=0
 >PRINT 1>0
 65535

 >PRINT B<>B
 0

 >PRINT B=B
 65535

 10 B=5
 20 IF 1<B.AND.B>4 THEN PRINT"RELATIONAL ARE ANDED TOGETHER
```

9.1 - SPECIAL INTERNAL CONTROL BITS

The top nine bits of the image table have been mapped to an internal counter and the status of the print buffer. This provides the programmer with useful timing bits and the print status. The timing bits toggle OFF to ON for the period listed in the following table. Along with one shots and counters accurately timed sequences can be executed. The bit numbers and their functions are listed as follows:

```
511  1 second period (timing)
510  1/2 second period (timing)
509  1/4 second period (timing)
508  1/8 second period (timing)
507  1/16 second period (timing)
506  1/32 second period (timing)
505  1/64 second period (timing)
504  1/128 second period (timing)
503  Console PRINT status bit
501  Aux RS232 DTE status bit
500  RS485 status bit
```

EXAMPLE:

```
10 XIL 509 : ROS 000
20 XIH 509 : OST 000 : PRINT"Here 4 times every second"
30 XIL 511 : ROS 001
40 XIH 511 : OST 001:CTU 000,100:CTR 000:?"after 100 secs"
.
60 XIL 503 : BAUD 1 : PRINT" send at 300 baud": OTL 300
70 XIL 503 : XIH 300 : BAUD 5 : OTU 300 : REM 9600 baud
```

9.2 - CONTROL BIT MAPPING

The following table shows the control bit mapping from 000-255. Expansion boards are mapped every 2 words because they use 16 bits. Any unused bits (039-255) or bits ranging from 256 to 502 can be used for internal relays in the user program. Remember that the bit image table is 64 words long (000-511 bits).

BIT NUMBERS								WORD	FUNCTION
007	006	005	004	003	002	001	000	00	Outputs 0-7
015	014	013	012	011	010	009	008	01	Outputs 8-15
023	022	021	020	019	018	017	016	02	Inputs 0-7
031	030	029	028	027	026	025	024	03	Inputs 8-15
039	038	037	036	035	034	033	032	04	EXIO 00
047	046	045	044	043	042	041	040	05	
055	054	053	052	051	050	049	048	06	EXIO 01
063	062	061	060	059	058	057	056	07	
071	070	069	068	067	066	065	064	08	EXIO 02
079	078	077	076	075	074	073	072	09	
087	086	085	084	083	082	081	080	10	EXIO 03
095	094	093	092	091	090	089	088	11	
103	102	101	100	099	098	097	096	12	EXIO 04
111	110	109	108	107	106	105	104	13	
119	118	117	116	115	114	113	112	14	EXIO 05
127	126	125	124	123	122	121	120	15	
135	134	133	132	131	130	129	128	16	EXIO 06
143	142	141	140	139	138	137	136	17	
151	150	149	148	147	146	145	144	18	EXIO 07
159	158	157	156	155	154	153	152	19	
167	166	165	164	163	162	161	160	20	EXIO 08
175	174	173	172	171	170	169	168	21	
183	182	181	180	179	178	177	176	22	EXIO 09
191	190	189	188	187	186	185	184	23	
199	198	197	196	195	194	193	192	24	EXIO 10
207	206	205	204	203	202	201	200	25	
215	214	213	212	211	210	209	208	26	EXIO 11
223	222	221	220	219	218	217	216	27	
231	230	229	228	227	226	225	224	28	EXIO 12
239	238	237	236	235	234	233	232	29	
247	246	245	244	243	242	241	240	30	EXIO 13
255	254	253	252	251	250	249	248	31	
499-256 internal relay bits									Relays
500 RS485 status bit								32-	
501 RS232 DTE port status bit								62	Status
502 I2C bus error bit								62	Timing
503 Console port status bit								63	
511-504 timing bits									

Image Table - Control Bit Mapping

Bit Numbers								Word	Function
511	510	509	508	507	506	505	504	63	Timing
503	502	501	500	499	498	497	496	62	Comm Status & I^2C Error
495	494	493	492	491	490	489	488	61	I^2C Status ?
487	486	465	464	483	482	481	480	60	I^2C Status ?
479	478	477	476	475	474	473	472	59	I^2C Status ?
471	470	469	468	467	466	465	464	58	
463	462	461	460	459	458	457	456	57	
455	454	453	452	451	450	449	448	56	
447	446	445	444	443	442	441	440	55	
439	438	437	436	435	434	433	432	54	
431	430	429	428	427	426	425	424	53	
423	422	421	420	419	418	417	416	52	
415	414	413	412	411	410	409	408	51	
407	406	405	404	403	402	401	400	50	
399	398	397	396	395	394	393	392	49	
391	390	389	388	387	386	385	384	48	
383	382	381	380	379	378	377	376	47	
375	374	373	372	371	370	369	368	46	
367	366	365	364	363	362	361	360	45	
359	358	357	356	355	354	353	352	44	
351	350	349	348	347	346	345	344	43	
343	342	341	340	339	338	337	336	42	
335	334	333	332	331	330	329	328	41	
327	326	325	324	323	322	321	320	40	
319	318	317	316	315	314	313	312	39	
311	310	309	308	307	306	305	304	38	
303	302	301	300	299	298	297	296	37	
295	294	293	292	291	290	289	288	36	
287	286	285	284	283	282	281	280	35	
279	278	277	276	275	274	273	272	34	
271	270	269	268	267	266	265	264	33	
263	262	261	260	259	258	257	256	32	

3.21 - IIC (cont'd) - IMAGE TABLE MAPPING

The following is a map of IIC hardware states to image table bits. The user program must test the appropriate state bits to control IIC block data transfers. It is up to the user to clear all bits! The IIC state only sets bits ON.

Bus error	state = 00	bit = 502
MT/MR Master start (receive or transmit)	state = 08	bit = 488
MT/MR Master repeated start	state = 10	bit = 489
MT Slave address + Write transmitted ACK received	state = 18	bit = 490
MT Slave address + Write transmitted NO ACK received	state = 20	bit = 491
MT Data transmitted and ACK received	state = 28	bit = 492
MT Data transmitted and NO ACK received	state = 30	bit = 493
MT/MR Arbitration lost in R/W or data	state = 38	bit = 494
MR Slave address + Read transmitted ACK received	state = 40	bit = 495
MR Slave address + Read transmitted NO ACK received	state = 48	bit = 496
MR Data received and ACK transmitted	state = 50	bit = 497
MR Data received and NO ACK transmitted	state = 58	bit = 498
SR Own address + Write received ACK transmitted	state = 60	bit = 472
SR Arbitration lost in Slave address	state = 68	bit = 473
SR General call (not applicable)	state = 70	bit = 474
SR Arbitration lost in general call	state = 78	bit = 475
SR Data received ACK transmitted	state = 80	bit = 476
SR Data receive NOACK transmitted	state = 88	bit = 477
SR General call (not applicable)	state = 90	bit = 478
SR General call (not applicable)	state = 98	bit = 479
SR Stop or Start while still addressed	state = A0	bit = 480
ST Slave address + Read received and ACK returned	state = A8	bit = 481
ST Arbitration lost in slave address	state = B0	bit = 482
ST Data transmitted and ACK received	state = B8	bit = 483
ST Data transmitted and NOACK received	state = C0	bit = 484
ST Last data byte transmitted and ACK received	state = C8	bit = 485

10.1 - EXTERNAL MEMORY MAPPING

LOCATIONS in HEX	MEMORY USAGE
0000H-00FFH	Basic Control Stack Usage
0100H-01FFH	Basic Argument Stack Usage
0200H-020FH	Reserved
0210H-021FH	ONINT line numbers
0220H-0221H	COMINT line number
0222H-0223H	COMINT # line number
0224H-0225H	COMINT % line number
0230H-024FH	JMP LBL statement bits
0250H-026FH	Oneshot statement bits
0270H-02AFH	I/O Bit image table
02B0H-02F5H	ONERR trapping line numbers
02FDH	Last error # occurred
02FEH	PLR collision value
02FFH	Warm or Cold Boot value
0300H-03FFH	PULSE statement values & bits
0400H-04FFH	PLW command buffer
0500H-05FFH	DLY time value=(S#*2)+1
0600H-06FFH	CTU count value=(S#*2)+1
0700H-07FFH	PRINT buffer
0800H-08FFH	@Integer @000=800H @255=8FFH
0900H-093FH	OTE statement flags
0940H-09FFH	Protected user storage area
0A00H-0A7FH	IIC Data Transfer buffers
0A80H-0A8FH	IIC Expansion IO data buffers
0A90H	Clock seconds
0A91H	Clock minutes
0A92H	Clock hours
0A93H	Clock day of week (Sun=1)
0A94H	Clock date
0A95H	Clock month
0A96H	Clock year
0A97H-0AFFH	RESERVED
0B00H-0BFFH	Console port receive buffer
0C00H-0CFFH	RS232 DTE port receive buffer
0D00H-0DFFH	RS485 port receive buffer
0E00H-0EFFH	RS485 port transmit buffer
0F00H-0FFFH	RS232 DTE port transmit buffer
1000H-17FFH	Line look up file area
1800H-19FFH	Free user space 512 bytes
1A00H-07FFFH	User BASIC program space (32k)
1A00H-0FFFFH	User BASIC program space (64K)

10.2 - INTERNAL MEMORY LOCATIONS

```
045H  Location of string in memory (MSB,LSB)
046H   after XSB $() is executed

048H  ONTIME Timer high byte
049H  ONTIME Timer low byte

04FH  8 bit ADC result
050H  LSB of 10 bit ADC result
051H  MSB of 10 bit ADC result

054H  Console port receive buffer pointer
055H  Character for Console port interrupt (default=13)

05BH  Console port transmit buffer in pointer
05CH  Console port transmit buffer out pointer

066H  RS485 receive buffer pointer

067H  RS232 DTE port receive buffer pointer

076H  RS485 transmit buffer in pointer
077H  RS485 transmit buffer out pointer

078H  RS232 DTE transmit buffer in pointer
079H  RS232 DTE transmit buffer out pointer

07AH  Character for RS485 port interrupt (default=13)

07BH  Character for RS232 port interrupt (default=13)
```

11.1 - CONFIGURATION JUMPERS

Figure 1 - CONSOLE BAUD, MEM PROTECT, RUN MODE, MSIZE, MC WDOG and 0-5/1-5 Jumpers
Figure 2 - Serial Port Jumpers, Remote Reset and Module Type

CONSOLE BAUD - This jumper sets the default console baud rate on the BAC552ES controller. With the jumper ON the rate is 19200 and with the jumper OFF the rate is 9600 baud.

MEM PROTECT - This jumper protects the BASIC file memory area from accidental erasure. With the jumper OFF the memory is protected and the NEW command generates an ERROR. Installing the jumper allows the NEW to clear the file area. Also if the jumper is ON and a RESET occurs the memory will also be erased.

RUN MODE - This jumper when installed locks the controller in the RUN MODE of operation. Also if the jumper is ON following a RESET condition a valid program found in FLASH memory will be transferred to the BASIC file area. The console port CTRL-C will not break program execution.

MEM SIZE - This jumper when installed sets the top of memory at 65535 bytes. With the jumper OFF the memory size is set at 32768 bytes. The memory size is only set after a RESET condition.

MC WDOG - This jumper controls the processor watchdog timer. With the jumper OFF the watchdog will never RESET the controller if it times out. In the event that the processor gets bumped out of control by a transient and the jumper is ON the watchdog will generate a system RESET just as if the user pressed the RESET button. This RESET signal is also transmitted to all expansion boards if any are connected. It is recommended that the watchdog be installed in all applications. This timer is toggled by the execution of the BASIC Interpreter.

0-5/1-5 - This jumper selects the low end reference of the analog to digital convertor. When the jumper is ON the analog to digital convertors range is 0-5Vdc and with the jumper OFF the ADC range is 1-5Vdc. The 1V reference is adjusted by adjusting VR1 which is located just above the jumper. A volt meter connected to the right most pin will show the low end voltage.

Console Tx/Rx - A 6 pin jumper selects which pins on the console port connector are connected to the controllers serial transmit and receive pins. The jumpers always install horizontally! With the two jumpers on the left side pin 3 is the received data and pin 2 is the transmitted data. This is the as shipped or normal position.

RTS/CTS EN - This jumper disables the hardware handshaking of the AUX DTE port when installed. With the jumper OFF the handshaking is enabled. Pin 8 on the 9 pin connector is the RTS sent from the DCE and pin 7 is the CTS sent to the DCE.

TERMination - This jumper terminates the RS485 port with a 120 ohm load resistor.

EX. RESET - These pins are not a jumper but a connector for an external RESET button. One pin is connected to the controller's 5 volt logic supply and the other is to the reset capacitor.

ROM CONFIGuration - These four 3 pin jumpers are used to configure the FLASH device as the controller ROM. This was done to make provision for future BASIC compilers. For normal ROM operation the jumpers are all on the left most pins (STD ROM).

AUX WDOG ENABLE - This jumper when installed enables the secondary watchdog timer on the LTC/MAX/ADM 691 device. This watchdog timer has a 1.6 second time out and must be toggled by the BASIC program. **This jumper must be removed before stopping the running program. The controller's initialization routines toggle 012 which is connected to this jumper. After start up the program must toggle 012.**

F E - This jumper which is located at the upper right corner of the FLASH device allows the FLASH chip to be replaced with a EPROM. This is for backward compatibility with older BAC552 controllers. The normal jumper position is on the left side by the 'F'.

64K 32K - This jumper selects the size of the controller's ROM memory. With the jumper on the left side a 64K device can be installed. Normally a 32K device is used, therefore the jumper is on the right side next to the '32K'.

ANALOG INPUT IMPEDANCE - These jumpers control the input impedance seen at the analog input terminals. With the two pins shorted a 250 ohm load is connected the appropriate channel. This allows 4-20mA devices to be connected to the analog channels. Note that in order to read full scale 0-1023 bits the ADC must be changed to 1-5V operation. This affects all channels. If full scale is not required and there is a mix of voltage and current sourced analogs, then the 4 Ma offset can be subtracted in the user program. See Figure 3.

DC->ADC7 - This jumper connects the 12 volt power supplying the controller to analog channel #7. The 250 ohm jumper must be first removed. There is a precision divide by 4 resistor ladder which allows ADC7 to read 0-20 Vdc.

BAT ON - This jumper is the lithium battery ON/OFF switch. If the controller is keep out of service for a long period of time it is recommended that this jumper be removed. Normally the jumper is ON.

013, 014, 015 - This pins are not jumpers but are 5V logic, open collector or LED

compatible outputs. The pins to the edge of the board are connected to the collectors of the NE5090 relay driver chips. The pins toward the 470 ohm resistors are pulled to the 5V logic supply through 470 ohm resistors.

11.2 - DISCRETE I/O TERMINAL CONNECTIONS

The BAC552 controller provides 16 local discrete inputs mapped to bits (016) to (031). These inputs are 12Vdc - 24Vdc. The inputs have a series connected LED for input indication and the input current at 12Vdc is approximately 5mA. There are two separate common terminals for two groups of eight inputs. This allows separate isolation from input sources if required. The common terminal next to input 016 is for inputs 016 to 023 and the common next to input 031 is for inputs 024 to 031. Please refer to Figure 4 for layout of these terminals.

There are a total of 13 outputs available to the user on the BAC552 controller. Ten of the outputs are normally open 5A high current outputs and three are LED outputs. The relay contacts are hard contacts, that means that there is no contact suppression on them. All relay contacts have no common between them, there is a separate connection at each side of each contact. **The user must provide suppression when connecting to inductive loads.** An example of different types of suppression is shown in the following page. **The BAC552 controller has no approval for the connection of line voltages to its outputs, only low voltage (<= 24Vdc or <= 24Vac) is accepted.** The user must connect **approved devices** such as contactors, relays or solid-state relays **approved for line voltage connections.** Please refer to data sheets on the relay switching currents.

The 3 LED outputs provide a forward LED current of approx. 9mA with the anode connection toward the 470 resistors and the cathode being at the edge of the board. The connectors for the LED connections are 0.1" spaced open pin headers. Please refer to Figure 5 for output terminal connections.

11.3 - ANALOG, COMMUNICATIONS and SUPPLY CONNECTIONS

A ten pin plug-on terminal connection is provided for the analog inputs. The analog inputs are single-ended and common to the controllers ground. A single common terminal and a +12Vdc supply is provided along with the 8 analog channel connections. The +12V is actually connected to the +DC supply and is provided to power analog devices if required. The common connection is to the PCB ground. The analog inputs are protected against over-voltage and negative signals by series resistors and clamping diodes. Input filtering is also provided. Please refer to figure 6 for the terminal layout.

The analog outputs also share a common ground with each other and the controller ground. The analog outputs range is 0-5Vdc. The output impedance is approx. 50 ohms (series limiting resistor) and connection to a high input impedance device is recommended. Figure 6 shows the terminals for analog outputs.
NOTE: Leave all used analog inputs configured for low input impedance (jumper ON). This minimizes any additional noise that may be seen at the analog to digital convertor.

The power supply to the BAC552 controller must be 12-15Vdc filtered @ a minimum of 450mA. With all outputs OFF the BAC current consumption is approx. 200mA, the relay coil and LED for each output require approx. 25mA. Reverse polarity protection is provided by a reversed biased diode. Refer to figure 6 for supply terminals.

The console RS232 port connection is provided by a female D9 connector. Pin 3 is the Receive, Pin 2 is the transmit and Pin 5 is ground. Pin 4 is set for DTR. Refer to figure 6 for connector location.

The RS485 connector is a 4 terminal plug with the RS485 signal lines and a 5Vdc supply connection. This supply connection is only recommended for low current local RS485 devices such a LCD display with no backlight and a keypad. This 5V connection should not exceed 100mA. The overall current consumption of the controller increases with the additionally connected devices to these terminals. Please refer to Figure 7 for RS485 connections.

The expansion IO connects to the closed 10 pin header connector.

11.4 - LOAD SUPPRESSION NETWORKS

The following figures 8 and 9 show the recommended suppression networks for DC and AC connected loads.

For DC powered circuits a free-wheeling diode is connected across the inductive load (ie relay coil). The energy stored in the inductive load is dissipated by the resistance of the load when the contact is open. The diode rating should be a PIV (peak inverse voltage) of 6 times the load voltage and a forward current greater than the load current. A 1N4003 diode is suitable for loads less than 1A at a voltage of 24Vdc.

11.5 - BAC552ES CONTROLLER INDICATORS

The BAC controller provides LED indicators for monitoring all outputs, inputs and board functions. All output relays have a LED to indicate the operation of the relay. The LED is located just behind each relay. There are LEDs for each input, located directly inline with each input terminal.

The **RUN LED** (see figure 6) is ON when the controller is activity running a program.

The **ERROR LED** (see figure 6) indicates that the interpreter detected an error in a operation. The ERROR LED will stay ON until the CLEAR E statement is executed.

The **LED 012** Indicator (see figure 6) is a user programable LED (output 012) to provide the user with a means of toggling the secondary watchdog timer. A simple **FLFP 012** in the main program loop will toggle this LED OFF for 1 loop and then ON for the next. If the program scan time is too long, then inserting a **OTL! 012: OTU! 012** will toggle the 012 output immediately without requiring the LIO statement.

The **RS485 Tx LED** (figure 2) indicates when the RS485 driver is enabled. This only indicates transmissions and the LED will only wink for the duration of the transmission.

The **DC ON LED** indicates that power is applied to the BAC controller board.

I2C 32 BYTE BLOCK TRANSFER TESTING ROUTINE

```
REM  ************************************************************
REM  ********** I2C 32 BYTE BLOCK TRANSFER TESTING ROUTINE*********
REM  ************************************************************
REM  I2C data transfers will be tested in this program by transmitting
REM  data to a Slave Receiver and receiving data form a Slave Transmitter
REM  Two BAC552 controllers are used.
REM  One is addressed as 40H and the other is addressed as 50H.
REM  The Slave data buffers will be loaded with dummy data.
REM  The COMIT (communication int) will be used to initiate activity.
REM  'T' will transmit to a slave controller receive buffer
REM  'R' will receive data from a slave controllers transmit buffer
REM  'V' will view the buffers
REM  'C' will clear the receive buffers (master and slave)
REM  Image table bits will be tested to ensure proper transfer.
REM
REM  Setup each controllers address
10   ADR 40H
REM  Enable serial comm interrupt
20   COMINT 1000: CIC 1
REM  Load some data in Master Transmitter buffer
30   A=0: FOR X=0A00H TO 0A1FH: A=A+1: XBY(X)=A: NEXT X
REM  Load some data in Slave Transmitter buffer
40   A=32: FOR X=0A60H TO 0A7FH: A=A+1: XBY(X)=A: NEXT X
REM  Clear all bits related to I2C status (I2C uses 4 words)
50   CLW 59: CLW 60: CLW 61: CLW 62
100  REM * loop *
110  XIH 488: XIH 490: XIH 492: PRINT"Master TX status - OK": JMP 000
111  LBL 000: OTU 488: OTU 490: OTU 492
120  XIH 488: XIH 495: XIH 497: JMP 001
121  XIH 488: XIH 495: XIH 498: JMP 001
122  LBL 001: CLW 61: PRINT"Master RX status - OK"
130  XIH 472: XIH 476: PRINT"Slave TX status - OK": JMP 002
131  LBL 002: OTU 472: OTU 476
140  XIH 481: XIH 483: PRINT"Slave RX status - OK": JMP 003
141  LBL 003: OTU 481: OTU 483
999  LIO: GOTO 100
1000 K=INKEY
1001  IF K=3 THEN  STOP: REM must provide a BREAK when using COMIT
1002  IF K=84 THEN  GOSUB 1100
1003  IF K=82 THEN  GOSUB 1200
1004  IF K=86 THEN  GOSUB 1300
1005  IF K=67 THEN  GOSUB 1400
1009  RETI
1100  IIC 50H,32 :REM Transmit 32 bytes to slave
```

```
1110  PRINT" Master transmission initiated"
1199  RETURN
1200  IIC 51H,32 :REM Receive 32 bytes from slave
1210  PRINT" Master reception initiated"
1299  RETURN
1300  PRINT: PRINT"Master receiver buffer contents:"
1310  FOR X=0A20H TO 0A3FH: PRINT XBY(X);: NEXT X: PRINT
1320  PRINT: PRINT"Slave receiver buffer contents:"
1330  FOR X=0A40H TO 0A5FH: PRINT XBY(X);: NEXT X: PRINT
1399  RETURN
1400  FOR X=0A20H TO 0A3FH:XBY(X)=0: NEXT X
1410  FOR X=0A40H TO 0A5FH:XBY(X)=0: NEXT X
1420  A=0: FOR X=0A00H TO 0A1FH: A=A+1: XBY(X)=A: NEXT X
1430  A=32: FOR X=0A60H TO 0A7FH: A=A+1: XBY(X)=A: NEXT X
1440  PRINT: PRINT" Buffer contents CLEARED": PRINT
1499  RETURN
```

PROGRAM EXAMPLE #1 - ALARM SYSTEM

This example shows how the BAC controller can operate as an alarm system controller.

```
REM ************************************************************
REM * THIS IS FOR DEMONSTRATION ONLY !!! NO LIABILITY IS ASSUMED *
REM ************************************************************
REM  *** ALARM SYSTEM WITH BAC552 CONTROLLER ***
REM  *** KEY SWITCH INPUT 016 (MOMENTARY ON/OFF) ***
REM  *** STATUS LEDS - 000 LOOP OK - 001 ALARM ON ***
REM  *** SIREN OUTPUT - 002 RELAY ***
REM  *** ONLY 1 ENTRY POINT - OUTER MAN DOOR AND INNER MAN DOOR
FROM GARAGE
REM  *** ONLY 30 SECONDS TO TURN OFF ALARM AFTER ENTRY INTO HOUSE
REM  *** NIGHT ALARM PROTECTS UN-ATTACHED GARAGE AND HOUSE
REM  *** GATE ON DECK AND FRONT SCREEN DOOR 023 ***
REM  *** BASEMENT ZONE 024 ***
REM  *** UN-ATTACHED GARAGE ZONE 025 ***
REM  *** OUTER MAN DOOR 026 ***
REM  *** INNER MAN DOOR 027 ***
REM  *** FRONT DOOR, KITCHEN DOOR, & UPPER DECK DOOR  028 ***
REM  *** MOTION DETECTORS 029 ***
5   OTU 004: OTU 005: OTU 006: OTU 007: OTU 266: OTU 267
10  CLEAR O: CLEAR S: CLEAR I: CLEAR B
REM  ************ FULL SECURITY AND NIGHT ALARM FLIPFLOPS ***************
REM  ************ START OF MAIN LOOP ************************************
90  XIH 024: XIH 025: XIH 026: XIH 027: XIH 028: OTE 000
100 XIH 016: XIH 000: DLY 005,002: OST 000: FLFP 256
105 XIL 016: ROS 000: RST 005
110 XIH 019: XIH 000: DLY 009,002: OST 002: FLFP 261
115 XIL 019: ROS 002: RST 009
120 XIH 256: OTE 001
121 XIH 261: OTE 001
REM  * FLIPFLOP 256 IS THE ON/OFF CONTROL OF THE ALARM *
REM  * FLIPFLOP 261 IS THE ON/OFF CONTROL FOR NIGHT ALARM *
REM  ************************ END ON/OFF FLIPFLOPS **********************
122 XIL 261: OST 004: OTU 002: RST 003: RST 004: ?"NIGHT INIT"
123 XIH 261: ROS 004
REM  *********** INITIALIZE ROUTINE AND EXIT DELAY TIMER ****************
125 XIL 256: OST 001: RST 006: ROS 003: GOSUB 1000
126 XIH 256: ROS 001: DLY 006,060: GOTO 140
130 GOTO 150
REM  ************IF ALARM IS OFF THEN SKIP THESE ROUTINES ***************
REM  ************ TEST ALL LOOPS FOR ALARM AND ENTRY TIMERS *************
140 XIL 024: OTL 257
```

```
141  XIL 025: OTL 257
142  XIL 028: OTL 257
143  XIL 026: OTL 258
144  XIH 258: DLY 000,060: OTL 257
145  XIL 027: OTL 259
146  XIH 259: DLY 001,030: OTL 257
147  XIH 510: OST 003: OTE 002: PRINT"ARMED"
REM  **** NIGHT ALARM CONTACTS
150  XIL 261: OTU 262: GOTO 160
151  XIL 024: OTL 262
152  XIL 025: OTL 262
153  XIL 026: OTL 262
154  XIL 028: OTL 262
155  XIL 023: OTL 262
156  XIH 262: JMP 005
160  REM *END OF NIGHT ALARM*
REM  ********** END OF LOOPS ROUTINE **********
REM  **** WHEN 257 LATCHES RUN SIREN FOR DELAY 002 THEN PULSE SIREN
******
200  XIH 257: XIL 260: OTL 002: DLY 002,060: RST 002: CTU 000,001: JMP 000
210  LBL 000: CTR 000: OTL 260
220  XIH 257: XIH 260: JMP 005
REM  * JUMP TO SIREN PULSING ROUTINE
REM  ********** END CONTINUOUS SIREN RUNNING **********
REM  ********** SIREN PULSING ROUTINE **********
REM  * PULSE FOR 2 SECS EVERY 15 SEC INTERVALS FOR CTU 001 COUNTS
230  LBL 005: OTL 002: DLY 004,004: OTU 002: DLY 003,011: JMP 001
235  LBL 001: RST 003: RST 004: CTU 001,010: CTR 001: XIL 262: JMP 006

REM  THE RE-ARMING ROUTINE AND CAN ONLY BE ENABLED IF NIGHT ALARM
IS OFF
240  LBL 006: XIH 024: JMP 002
250  LBL 002: XIH 025: XIH 026: XIH 027: XIH 028: OTU 260: JMP 003
260  LBL 003: RST 000: RST 001: RST 002: CTR 000: CTR 001: JMP 004
270  LBL 004: OTU 257: OTU 258: OTU 259: ?"RE-ARMED"

999  LIO: OTC: GOTO 90
1000 REM **initialization routine**
1001 RST 000: RST 001: RST 002: RST 003: RST 004: RST 006
1002 CTR 000: CTR 001
1003 OTU 257: OTU 258: OTU 259: OTU 260
1004 OTU 002: OTU 013
1005 ?"INIT ROUTINE"
1009 RETURN
```

PROGRAM EXAMPLE #2 - BACLCD ON RS485

This example below shows how to send and receive data to and from the BACLCD display unit from the BAC controller on the RS-485 serial link. The example uses timing bit 511 to GOSUB to a routine which examines the state of bit 000 (relay 0) and then sends 1 of 2 screens to the BACLCD. The state of bit 000 is determined by the character received on the RS-485 from the BACLCD inputs. The BACLCD transmits a CHR(128) for a transition on input 0 and a CHR(129) for input 1. The routine for screen 1 also toggles bit 100 and based on the state of bit 100, characters are sent to 1 of the BACLCD outputs to toggle it OFF and ON.

```
5    REM *** BACLCD on RS485 program example ***
10   COMINT #,1000: CIC#,1
30   PRINT #CHR(27),CHR(3)
40   FOR T=1 TO 100: NEXT T
50   XIL 511: ROS 000
60   XIH 511: OST 000: GOSUB 150
70   LIO: GOTO 50
100  REM two different display screens, clear display before each
150  XIL 000: ROS 001: GOSUB 200: REM screen 1
155  XIH 000: OST 001: PRINT #CHR(27),CHR(3): FOR T=1 TO 100: NEXT T
160  XIH 000: ROS 002: GOSUB 400: REM screen 2
170  XIL 000: OST 002: PRINT #CHR(27),CHR(3): FOR T=1 TO 100: NEXT T
180  RETURN
200  REM first display screen
210  PRINT #CHR(27),CHR(2),CHR(96); :TIME #;
220  PRINT #CHR(27),CHR(1),CHR(0);
225  PRINT #"THE INPUT SUPPLY VOLTAGE IS = ";
230  PRINT #F(##.##),ADC7*.01955
240  FLFP 100: REM flip 100 for toggling output at BACLCD
250  XIH 100: PRINT #CHR(27),CHR(4);
260  XIL 100: PRINT #CHR(27),CHR(5);
270  PRINT #CHR(27),CHR(2),CHR(64);: DATE #;
280  RETURN
400  REM send only time in this screen
410  PRINT #CHR(27),CHR(1),CHR(32);: TIME #;
420  RETURN
1000 REM communication interrupt routine
1010 CIC #,0: A=XBY(0D00H): REM read first byte in just like INKEY does
1020 IF A=3 THEN  CIC#,0 : STOP: REM user break for COMINT
1030 IF A=128 THEN  OTL 000: REM received 128 from BACLCD set 000
1040 IF A=129 THEN  OTU 000: REM received 129 from BACLCD clr 000
1050 CIC #,1: RETI
```

PROGRAM EXAMPLE #3 - REMOTE MODEM

This application program was written to demonstrate remote control with a BAC552 controller via a serial modem.
This program only applies to BAC ROM versions 2.1 and higher. The version 2.1 has enhanced serial interrupt routines which make remote control very easy. A new statement XSB (examine serial buffer) enables the programmer to test received strings in the COM buffer (16 bytes max).
The following program was written to demonstrate the techniques required to successfully communicate via a modem with a BAC552 controller. The ONTIME statement along with TIC (timer interrupt control) are used for time-out routines. The COMINT, CIC and XSB demonstrate how serial buffer testing is done. A subroutine which functions as a PRINT queue driver is also included. The queue driver is necessary to prevent queue overflow. Other subroutines demonstrate how to capture RTC times and save them for time stamping of events.

```
10   REM This program demonstrates the ability of the BAC552 controller
11   REM to handle remote data logging and communications with a modem.
12   REM The BAC552 BASIC V2.1 has enhanced serial port interrupt
13   REM routines which make communication with a modem very easy.
14   REM Version 2.0 has only a single byte buffer (INKEY) where as V2.1
15   REM has a 16 byte ring buffer. The BASIC will now vector to the
16   REM line number in the COMINT statement only after a CR is received.
17   REM Any received Lfs (line feed) are stripped out.
18   REM
19   REM The buffer is in external memory addressed at 0B00H to 0B0FH.
20   REM A new statement called XSB (examine serial buffer) is very
21   REM powerful for testing the buffer contents. The XSB will be true
22   REM if the string in the quotes (XSB "NO CARRIER") matches the string
23   REM in the buffer. The string in the XSB does not have to be
24   REM complete. XSB "NO CAR" will read true as XSB "NO CARRIER". The
25   REM XSB only matches the bytes up to the last (").
26   REM 15 bytes or less can be tested. (MAX buffer size-1)
27   REM This program will setup up the modem for auto answer after 5
28   REM rings. The modem power is controlled by relay 000. This gives
29   REM the controller the ability to RESET the modem by powering it
30   REM down if the modem crashes!
40   REM
41   REM NOTE: Once the program is executed the CTRL-C break no longer
42   REM works! The user must press the RESET button with the AUTO RUN
43   REM jumper OFF. The controller will then goto the default baud rate
44   REM of 9600 baud. For testing purposes line 100 can be changed to
45   REM 100 REM. The user can then simulate the modem with the console.
50   REM ************************************************************
```

```
51   REM  * THIS IS FOR DEMONSTRATION ONLY - NO LIABILITY IS ASSUMED *
52   REM  * FOR ITS USE ! *
53   REM  ************************************************************
55   REM  It has been discovered that upon connection some garbage from
56   REM  the host ends up in the serial buffer, therefore before entering
57   REM  the password, enter a CR which will in effect purge the buffer.
60   REM  To operate this package follow these instructions:
61   REM  1)-connect 1200 baud stand alone modem (get TxD and RxD correct)
62   REM  2)-install AUTO RUN jumper and press RESET button
63   REM  (the controller will auto run and setup the modem for 5 rings)
64   REM  3)-dial up remote from host unit
65   REM  4)-after connection enter a CR to purge buffer (as noted above)
66   REM  5)-enter password PW7231786TEST +CR NOTE: you see no echo
67   REM  (the controller will respond if password is OK)
68   REM  6)-enter SEND DATA +CR (as many times as you want)
69   REM  (the controller will send data to you)
70   REM  7)-enter END +CR to terminate or just drop carrier at host
71   REM
100   BAUD 2: REM set baud for 1200 or REM for testing
110   CLEAR B: CLEAR R: CLEAR O: CLEAR I: CLEAR D: CLEAR C: LIO
120   COMINT 1000: CIC 0: ONTIME 90,2000: TIC 0
130   DLY 000,005: RST 000: GOTO 150
135   DLY 001,002: OTL 000: LIO: RST 001
140   GOTO 130: REM turn modem ON after 2 secs while waiting for 5 seconds
150   PRINT "AT S0=5": CIC 1: TIC 1: REM enable comm & timeout interrupts
200   REM main loop
210   XIH 016: OST 010: GOSUB 3000
220   XIH 017: OST 011: GOSUB 3100
230   XIH 018: OST 012: GOSUB 3200
240   XIH 019: OST 013: GOSUB 3300
250   XIH 020: OST 014: GOSUB 3400
300   REM the subroutine at 4000 will handle flow control for the queue
310   XIH 100: XIH 101: GOSUB 4000
400   CTU 001,050: FLFP 012: CTR 001
500   XIH 151: OTU 150: OTU 151: OTU 100: OTU 110: OTU 101: GOTO 100
505   REM  line 500 resets everything
600   XIH 152: DLY 002,002: OTL 153: DLY 003,002: OTL 154: JMP 005
605   LBL 005: RST 002: RST 003: OTU 152: OTU 153: OTU 154: ROS 030: ROS
031
610   REM line 600 sends +++ (pause) ATH to hang-up modem
620   XIH 153: OST 030: PRINT "+++";
630   XIH 154: OST 031: PRINT "ATH"
640   REM bits 153 and 154 are used to time the escape hang-up strings
999   LIO: GOTO 200
```

```
1000  CIC 0
1010  XSB "OK": TIC 0: OTL 150: ROS 000
REM if modem responds turn OFF timer and set bit 150
1020  XSB "RING": OST 000: TIC 1
REM timer back ON
1030  XSB "CONNECT": OTL 110: PRINT "CONNECT": ROS 000: TIC 1
REM timer ON
1040  XIH 110: XSB "PW7231786TEST": OTL 100: TIC 0: PRINT "Password
accepted"
REM set a bit if password matches
1050  XSB "NO CARRIER": ROS 000: OTU 100: OTU 110: OTU 152
REM clear the password bit
1060  XIH 100: XSB "SEND DATA": ROS 001: OTL 101: PRINT "Sending data now"
REM controller will respond to a command SEND DATA
1070  XSB "END": OTU 100: OTU 101: OTU 150: OTU 151: JMP 001
1075  LBL 001: OTU 110: PRINT "Bye !": OTL 152
1080  XIH 100: XSB "ON1": OTL 001: PRINT "Relay 1 ON"
1085  XIH 100: XSB "OFF1": OTU 001: PRINT "Relay 1 OFF"
1099  CIC 1: RETI
2000  TIC 0: REM timer interrupt
2010  XIL 150: OTL 151: RETI
2020  REM if 150 is clr then set 151 so main loop can reset modem
2030  CIC 0: OTU 150: OTU 151: OTU 100: OTU 110: OTU 101: CIC 1
2040  RETI
3000   OTL 200
3005   REM set bit and save HMS in protected mem for time stamp
3010   XBY(0B20H)=XBY(8212H): XBY(0B21H)=XBY(8211H):
XBY(0B22H)=XBY(8210H)
3020   RETURN
3100   OTL 201
3105   REM set bit and save HMS in protected mem for time stamp
3110   XBY(0B23H)=XBY(8212H): XBY(0B24H)=XBY(8211H):
XBY(0B25H)=XBY(8210H)
3120   RETURN
3200   OTL 202
3205   REM set bit and save HMS in protected mem for time stamp
3210   XBY(0B26H)=XBY(8212H): XBY(0B27H)=XBY(8211H):
XBY(0B28H)=XBY(8210H)
3220   RETURN
3300   OTL 203
3305   REM set bit and save HMS in protected mem for time stamp
3310   XBY(0B29H)=XBY(8212H): XBY(0B2AH)=XBY(8211H):
XBY(0B2BH)=XBY(8210H)
3320   RETURN
```

```
3400   OTL 204
3405   REM set bit and save HMS in protected mem for time stamp
3410   XBY(0B2CH)=XBY(8212H): XBY(0B2DH)=XBY(8211H):
XBY(0B2EH)=XBY(8210H)
3420   RETURN



4000   REM This subroutine works as a driver to handle the print queue
4005   XIL 503: OST 001: CTU 000,010: GOTO 4020
4010   XIH 503: RETURN
4020   XCT 000,001: GOSUB 4500: ROS 001: RETURN
4030   XCT 000,002: GOSUB 4600: ROS 001: RETURN
4040   XCT 000,003: GOSUB 4650: ROS 001: RETURN
4050   XCT 000,004: GOSUB 4700: ROS 001: RETURN
4060   XCT 000,005: GOSUB 4750: ROS 001: RETURN
4070   XCT 000,006: GOSUB 4800: ROS 001: RETURN
4080   XCT 000,007: OTU 101: CTR 000: REM stop data request and clr counter
4099   RETURN

4500   PRINT CHR(12): PRINT "* BAC552 remote data report *": PRINT
4502   PRINT "Main AC supply voltage is ",(ADC0*.293)
4505   PRINT "Battery voltage is ",(ADC1*.059)
4510   PRINT "Building temperature is ",(ADC2*.195)
4520   PRINT "Controller supply voltage is ",(ADC7*.0195)
4590   REM ** The user is limited to print up to 255 bytes each time **
4599   RETURN

4600   XIL 200: PRINT "ALARM #1 STATUS - OK": RETURN
4610   PRINT "ALARM #1 at ",XBY(0B20H),":",XBY(0B21H),":",XBY(0B22H)
4620   ROS 010: OTU 200: RETURN
4650   XIL 201: PRINT "ALARM #2 STATUS - OK": RETURN
4660   PRINT "ALARM #2 at ",XBY(0B23H),":",XBY(0B24H),":",XBY(0B25H)
4670   ROS 011: OTU 201: RETURN
4700   XIL 202: PRINT "ALARM #3 STATUS - OK": RETURN
4710   PRINT "ALARM #3 at ",XBY(0B26H),":",XBY(0B27H),":",XBY(0B28H)
4720   ROS 012: OTU 202: RETURN
4750   XIL 203: PRINT "ALARM #4 STATUS - OK": RETURN
4760   PRINT "ALARM #4 at ",XBY(0B29H),":",XBY(0B2AH),":",XBY(0B2BH)
4770   ROS 013: OTU 203: RETURN
4800   XIL 204: PRINT "ALARM #5 STATUS - OK": RETURN
4810   PRINT "ALARM #5 at ",XBY(0B2CH),":",XBY(0B2DH),":",XBY(0B2EH)
4820   ROS 014: OTU 204: RETURN
```

EEPROM.BAS

```
10   REM  This sample program will show how to write and read back from
11   REM  a serial EEPROM. The program first loads 256 memory locations
12   REM  with a number pattern.
13   REM  Then each EEPROM location is written to using the EEPROM
14   REM  statement. The syntax is - EEPROM [address],[location],[data].
15   REM  When writing to EEPROM the bit 0 of the address is zero.
16   REM  The location can be a number or variable and the same is for the
17   REM  data.
18   REM  When reading a EEPROM location the bit zero of the address is 1
19   REM  The location can be a variable or a value.
20   REM  The data variable is where the data is to be returned in.
21   REM  The FOR T loops are used to delay the program for the EEPROM
22   REM  reads and writes. This is only required if the data is written
23   REM  continuously as in this program. If your application is only
24   REM  writing 1 location initiated by a slow event then its not
25   REM  required.
26   REM
30   FOR X=0 TO 255: XBY(2048+X)=255-X: NEXT X: REM write data
35   PRINT: PRINT" NOW SAVE DATA TO EEPROM"
40   FOR X=0 TO 255
50   EEPROM 0A0H,X,XBY(2048+X): REM Bit zero of address = 0 for write
60   PRINT "SAVING DATA IN LOCATION -",X
70   FOR T=1 TO 50: NEXT T
80   NEXT X
90   REM Now read our data back
100   FOR T=1 TO 200: NEXT T
110   FOR X=0 TO 255
115   REM ** Our data is loaded into variable DAT
120   EEPROM 0A1H,X,DAT
125   REM Bit zero of address is = 1 for read operation
130   PRINT "READING DATA FROM LOCATION ",X,"   DATA =",DAT
140   FOR T=1 TO 50: NEXT T
150   NEXT X
```

IIC [INTEGER],[INTEGER]

The statement IIC [address],[number bytes] initiates a master mode transmit or receive. The transmit or receive is determined by the last bit in the address. Therefore, to initiate a master transmission the address would be even (42H) and for receive the address would be odd (43H). The maximum number of bytes is 32. A 128 byte block of SRAM is reserved for IIC data buffers. The first block is for master transmit and is addressed at 0A00H to 0A1FH. The second is for master receive and is addressed at 0A20H to 0A3FH. The third is for slave receive and is addressed at 0A40H to 0A5FH. The last is for slave transmit and is addressed at 0A60H to 0A7FH.

This simple IIC operating system was devised to transfer blocks of data between multiple BAC552 controllers. To transfer data to another controller the master transmit buffer is loaded with data, then the IIC statement is executed. (address and number of bytes must be specified)
Assuming that another controller has been setup with the matching address, the transmitted data would appear in it's slave receive buffer. If the address was odd (master receiver) then data can be requested from the other controllers slave transmitter buffer. Four of the upper image table words are used to test the IIC hardware states for IIC data transfers.

It is recommended that the user become familiar with the Philips IIC bus and how it operates.
IIC bus operational description is available in the Philips Microcontroller Handbook.
A sample demonstration program shows how to block transfer data between two controllers using the IIC bus.

NOTE: Not all user configurations of 552 controllers and expansion boards will allow easy implementation of this IIC block data transfer. For example, only one X10 controller can exist in any system because of address conflicts (X10 uses IIC bus communication). Please discuss your application with SYLVA if you have any concerns.

ADR [INTEGER]

The ADR statement sets the I2C address for the BAC552 controller. If the ADR is not executed then the controller will have a default address of 30H. The I2C address uses the upper 7 bits and a address of 31H is the same as 30H. The ADR need only be executed once at the program start.

IIC - IMAGE TABLE MAPPING

The following is a map of IIC hardware states to image table bits. The user program must test the appropriate state bits to control IIC block data transfers. It is up to the user to clear all bits! The IIC state only sets bits ON.

Bus error	state=00	bit=502
MT/MR Master start (receive or transmit)	state=08	bit=488
MT/MR Master repeated start	state=10	bit=489
MT Slave address + Write transmitted ACK received	state=18	bit=490
MT Slave address + Write transmitted NO ACK received	state=20	bit=491
MT Data transmitted and ACK received	state=28	bit=492
MT Data transmitted and NO ACK received	state=30	bit=493
MT/MR Arbitration lost in R/W or data	state=38	bit=494
MR Slave address + Read transmitted ACK received	state=40	bit=495
MR Slave address + Read transmitted NO ACK received	state=48	bit=496
MR Data received and ACK transmitted	state=50	bit=497
MR Data received and NO ACK transmitted	state=58	bit=498
SR Own address + Write received ACK transmitted	state=60	bit=472

SR	Arbitration lost in Slave address	state=68	bit=473	
SR	General call (not applicable)	state=70	bit=474	
SR	Arbitration lost in general call	state=78	bit=475	
SR	Data received ACK transmitted	state=80	bit=476	
SR	Data receive NOACK transmitted	state=88	bit=477	
SR	General call (not applicable)	state=90	bit=478	
SR	General call (not applicable)	state=98	bit=479	
SR	Stop or Start while still addressed	state=A0		bit=480
ST	Slave address + Read received and ACK returned		state=A8	bit=481
ST	Arbitration lost in slave address	state=B0	bit=482	
ST	Data transmitted and ACK received	state=B8	bit=483	
ST	Data transmitted and NOACK received	state=C0	bit=484	
ST	Last data byte transmitted and ACK received	state=C8		bit=485

3.21 - IIC (cont'd) - IMAGE TABLE MAPPING

The following is a map of IIC hardware states to image table bits. The user program must test the appropriate state bits to control IIC block data transfers. It is up to the user to clear all bits! The IIC state only sets bits ON.

Bus error	state = 00	bit = 502
MT/MR Master start (receive or transmit)	state = 08	bit = 488
MT/MR Master repeated start	state = 10	bit = 489
MT Slave address + Write transmitted ACK received	state = 18	bit = 490
MT Slave address + Write transmitted NO ACK received	state = 20	bit = 491
MT Data transmitted and ACK received	state = 28	bit = 492
MT Data transmitted and NO ACK received	state = 30	bit = 493
MT/MR Arbitration lost in R/W or data	state = 38	bit = 494
MR Slave address + Read transmitted ACK received	state = 40	bit = 495
MR Slave address + Read transmitted NO ACK received	state = 48	bit = 496
MR Data received and ACK transmitted	state = 50	bit = 497
MR Data received and NO ACK transmitted	state = 58	bit = 498
SR Own address + Write received ACK transmitted	state = 60	bit = 472
SR Arbitration lost in Slave address	state = 68	bit = 473
SR General call (not applicable)	state = 70	bit = 474
SR Arbitration lost in general call	state = 78	bit = 475
SR Data received ACK transmitted	state = 80	bit = 476
SR Data receive NOACK transmitted	state = 88	bit = 477
SR General call (not applicable)	state = 90	bit = 478
SR General call (not applicable)	state = 98	bit = 479
SR Stop or Start while still addressed	state = A0	bit = 480
ST Slave address + Read received and ACK returned	state = A8	bit = 481
ST Arbitration lost in slave address	state = B0	bit = 482
ST Data transmitted and ACK received	state = B8	bit = 483
ST Data transmitted and NOACK received	state = C0	bit = 484
ST Last data byte transmitted and ACK received	state = C8	bit = 485

I2C 40H . BAS

```
REM    ************************************************************************
REM    ********** I2C 32 BYTE BLOCK TRANSFER TESTING ROUTINE **************
REM    ************************************************************************
REM    I2C data transfers will be tested in this program by transmitting
REM    data to a Slave Receiver and receiving data form a Slave Transmitter
REM    Two BAC552 controllers are used. One is addressed as 40H and the other
REM    is addressed as 50H.
REM    The Slave data buffers will be loaded with dummy data.
REM    The COMIT (communication interrupt) will be used to initiate activity.
REM    'T' will transmit to a slave controller receive buffer
REM    'R' will receive data from a slave controllers transmit buffer
REM    'V' will view the buffers
REM    'C' will clear the receive buffers (master and slave)
REM    Image table bits will be tested to ensure proper transfer.
REM
REM    Setup each controllers address
10     ADR 40H
REM    Enable serial comm interrupt
20     COMINT 1000: CIC 1
REM    Load some data in Master Transmitter buffer
30     A=0: FOR X=0A00H TO 0A1FH: A=A+1: XBY(X)=A: NEXT X
REM    Load some data in Slave Transmitter buffer
40     A=32: FOR X=0A60H TO 0A7FH: A=A+1: XBY(X)=A: NEXT X
REM    Clear all bits related to I2C status (I2C uses 4 words)
50     CLW 59: CLW 60: CLW 61: CLW 62
100    REM * loop *
110    XIH 488: XIH 490: XIH 492: PRINT"Master TX status - OK": JMP 000
111    LBL 000: OTU 488: OTU 490: OTU 492
120    XIH 488: XIH 495: XIH 497: JMP 001
121    XIH 488: XIH 495: XIH 498: JMP 001
122    LBL 001: CLW 61: PRINT"Master RX status - OK"
130    XIH 472: XIH 476: PRINT"Slave TX status - OK": JMP 002
131    LBL 002: OTU 472: OTU 476
140    XIH 481: XIH 483: PRINT"Slave RX status - OK": JMP 003
141    LBL 003: OTU 481: OTU 483
999    LIO: GOTO 100
1000   K=INKEY
1001    IF K=3 THEN  STOP: REM must provide a BREAK when using COMIT
1002    IF K=84 THEN  GOSUB 1100
1003    IF K=82 THEN  GOSUB 1200
1004    IF K=86 THEN  GOSUB 1300
1005    IF K=67 THEN  GOSUB 1400
1009    RETI
1100    IIC 50H,32 :REM Transmit 32 bytes to slave
1110    PRINT" Master transmission initiated"
1199    RETURN
1200    IIC 51H,32 :REM Receive 32 bytes from slave  →
1210    PRINT" Master reception initiated"
1299    RETURN
1300    PRINT: PRINT"Master receiver buffer contents:"
1310    FOR X=0A20H TO 0A3FH: PRINT XBY(X);: NEXT X: PRINT
1320    PRINT: PRINT"Slave receiver buffer contents:"
1330    FOR X=0A40H TO 0A5FH: PRINT XBY(X);: NEXT X: PRINT
1399    RETURN
1400    FOR X=0A20H TO 0A3FH:XBY(X)=0: NEXT X
1410    FOR X=0A40H TO 0A5FH:XBY(X)=0: NEXT X
1420    A=0: FOR X=0A00H TO 0A1FH: A=A+1: XBY(X)=A: NEXT X
1430    A=32: FOR X=0A60H TO 0A7FH: A=A+1: XBY(X)=A: NEXT X
1440    PRINT: PRINT" Buffer contents CLEARED": PRINT
1499    RETURN
```

I2C 50H.BAS

```
REM    ************************************************************************
REM    ********** I2C 32 BYTE BLOCK TRANSFER TESTING ROUTINE **************
REM    ************************************************************************
REM    I2C data transfers will be tested in this program by transmitting
REM    data to a Slave Receiver and receiving data form a Slave Transmitter
REM    Two BAC552 controllers are used. One is addressed as 40H and the other
REM    is addressed as 50H.
REM    The Slave data buffers will be loaded with dummy data.
REM    The COMIT (communication interrupt) will be used to initiate activity.
REM    'T' will transmit to a slave controller receive buffer
REM    'R' will receive data from a slave controllers transmit buffer
REM    'V' will view the buffers
REM    'C' will clear the receive buffers (master and slave)
REM    Image table bits will be tested to ensure proper transfer.
REM
REM    Setup each controllers address
10     ADR 50H
REM    Enable serial comm interrupt
20     COMINT 1000: CIC 1
REM    Load some data in Master Transmitter buffer
30     A=64: FOR X=0A00H TO 0A1FH: A=A+1: XBY(X)=A: NEXT X
REM    Load some data in Slave Transmitter buffer
40     A=96: FOR X=0A60H TO 0A7FH: A=A+1: XBY(X)=A: NEXT X
REM    Clear all bits related to I2C status (I2C uses 4 words)
50     CLW 59: CLW 60: CLW 61: CLW 62
100    REM * loop *
110    XIH 488: XIH 490: XIH 492: PRINT"Master TX status - OK": JMP 000
111    LBL 000: OTU 488: OTU 490: OTU 492
120    XIH 488: XIH 495: XIH 497: JMP 001
121    XIH 488: XIH 495: XIH 498: JMP 001
122    LBL 001: CLW 61: PRINT"Master RX status - OK"
130    XIH 472: XIH 476: PRINT"Slave TX status - OK": JMP 002
131    LBL 002: OTU 472: OTU 476
140    XIH 481: XIH 483: PRINT"Slave RX status - OK": JMP 003
141    LBL 003: OTU 481: OTU 483
999    LIO: GOTO 100
1000   K=INKEY
1001    IF K=3 THEN  STOP: REM must provide a BREAK when using COMIT
1002    IF K=84 THEN  GOSUB 1100
1003    IF K=82 THEN  GOSUB 1200
1004    IF K=86 THEN  GOSUB 1300
1005    IF K=67 THEN  GOSUB 1400
1009    RETI
1100    IIC 40H,32 :REM Transmit 32 bytes to slave
1110    PRINT" Master transmission initiated"
1199    RETURN
1200    IIC 41H,32 :REM Receive 32 bytes from slave
1210    PRINT" Master reception initiated"
1299    RETURN
1300    PRINT: PRINT"Master receiver buffer contents:"
1310    FOR X=0A20H TO 0A3FH: PRINT XBY(X);: NEXT X: PRINT
1320    PRINT: PRINT"Slave receiver buffer contents:"
1330    FOR X=0A40H TO 0A5FH: PRINT XBY(X);: NEXT X: PRINT
1399    RETURN
1400    FOR X=0A20H TO 0A3FH:XBY(X)=0: NEXT X
1410    FOR X=0A40H TO 0A5FH:XBY(X)=0: NEXT X
1420    A=64: FOR X=0A00H TO 0A1FH: A=A+1: XBY(X)=A: NEXT X
1430    A=96: FOR X=0A60H TO 0A7FH: A=A+1: XBY(X)=A: NEXT X
1440    PRINT: PRINT" Buffer contents CLEARED": PRINT
1499    RETURN
```

I2CEMUL.BAS

```
1       REM The following program demonstrates how to use the ASM CALLS
2       REM to emulate an older I2C connected keypad device.
3       REM These calls are 361AH for the console port, 361DH for the AUX
4       REM port and 3620H for the RS485 port.
5       REM
9       REM The ASM calls greatly reduce the BASIC code that would be required
10      REM to perform the same task.
11      REM The user must add the CLW 07 after each bit test to clear the I/O
12      REM word as the old EXIO did.
13      REM
20      REM I2CEMUL.BAS Program DEMO
21      REM
25      CLEAR C: CLEAR O: CLW 00: CLW 01: CLEAR J
26      COMINT #,2000: CIC #,1: REM use RS485 port
30      REM keys 0-3
40      XIH 056: XIH 060: FLFP 000: CLW 07
45      XIH 057: XIH 060: FLFP 001: CLW 07
50      XIH 058: XIH 060: FLFP 002: CLW 07
55      XIH 059: XIH 060: FLFP 003: CLW 07
60      REM keys 4-7
65      XIH 056: XIH 061: FLFP 004: CLW 07
70      XIH 057: XIH 061: FLFP 005: CLW 07
75      XIH 058: XIH 061: FLFP 006: CLW 07
80      XIH 059: XIH 061: FLFP 007: CLW 07
85      REM keys 8-B
90      XIH 056: XIH 062: FLFP 008: CLW 07
95      XIH 057: XIH 062: FLFP 009: CLW 07
100     XIH 058: XIH 062: PRINT "KEY A": CLW 07
105     XIH 059: XIH 062: PRINT "KEY B": CLW 07
106     REM keys C-F
110     XIH 056: XIH 063: PRINT "KEY C": CLW 07
115     XIH 057: XIH 063: PRINT "KEY D": CLW 07
120     XIH 058: XIH 063: PRINT "KEY E": CLW 07
125     XIH 059: XIH 063: PRINT "KEY F": CLW 07
180     FLFP 012
190     LIO : GOTO 30

2000    CIC #,0
2010    CALL 3620H
2020    HEXOUT"The I/O image table data is ",XBY(277H)
2030    CIC #,1
2099    RETI
```