

Overview:

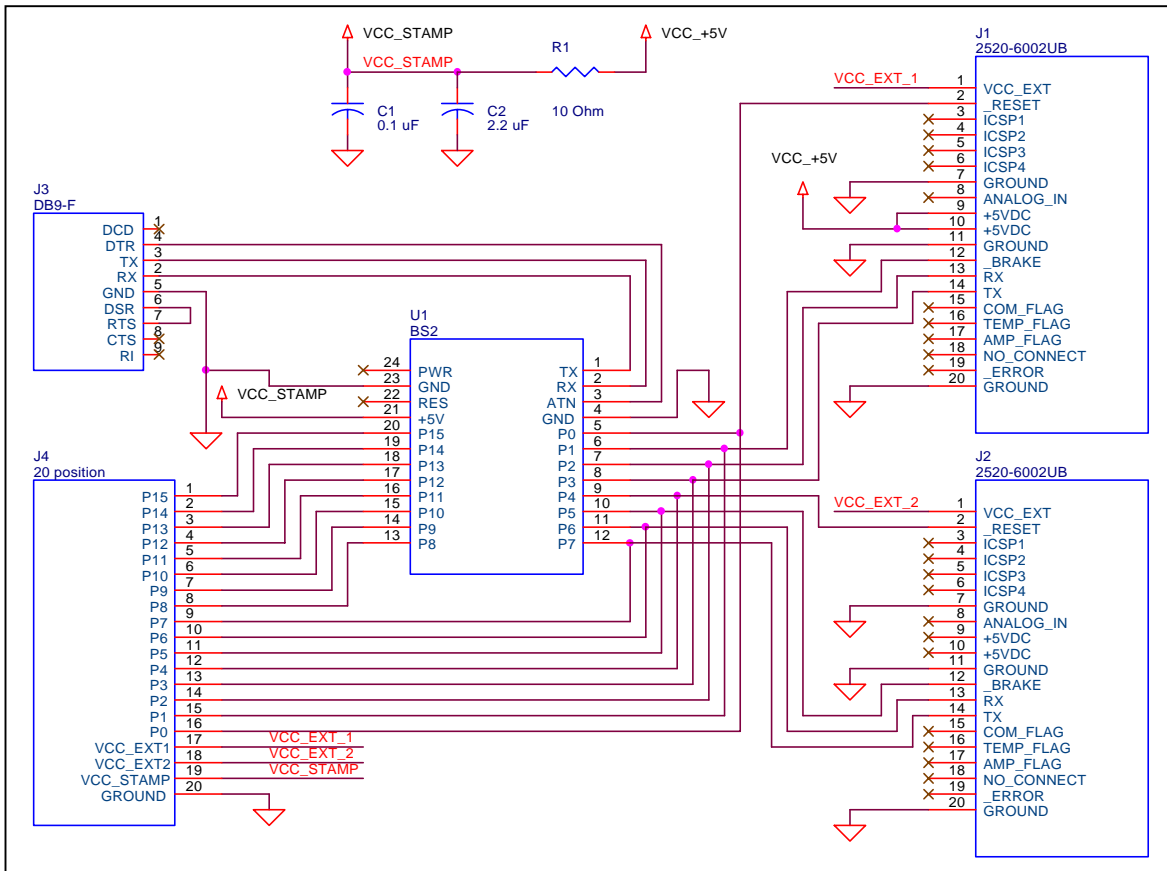
The purpose of this application note is to provide guidance and example software for BASIC Stamp users attempting to interface the BASIC Stamp to the ICON Interface Module. For this application the hardware and software is configured to control 2 ICON Interface Modules with a single BASIC Stamp 2 (BS2). This configuration could be useful in controlling the drive motors of a two-wheeled robot. The BASIC Stamp software is written in PBASIC (Parallax-BASIC), for the BS2 IC module, but could be quickly converted for use with the BS2SX by modifying the BAUD constant and adjusting some of the other timing constants.

This application note provides subroutines for performing various maneuvers with a two-wheeled locomotion system, but the software could easily be modified to control a single motor. Furthermore the application note provides excellent examples for implementing the ICON Interface Module communication protocol.

Hardware:

Connectivity and control of the ICON Interface Module and ICON H-Bridge can be simplified to a 38,400BPS N,8,1 serial interface. While the ICON Interface Module possesses five modes of operation, the BASIC Stamp in this application will use just the serial control mode. A DB9-female connector is used to program the BS2, while a 20-pin header provides access to each BS2 I/O pin, and various power connections. It is assumed that the BS2 is powered by the +5VDC output available on the 20-pin connector connected to ICON Interface Module #1 (connected to J1).

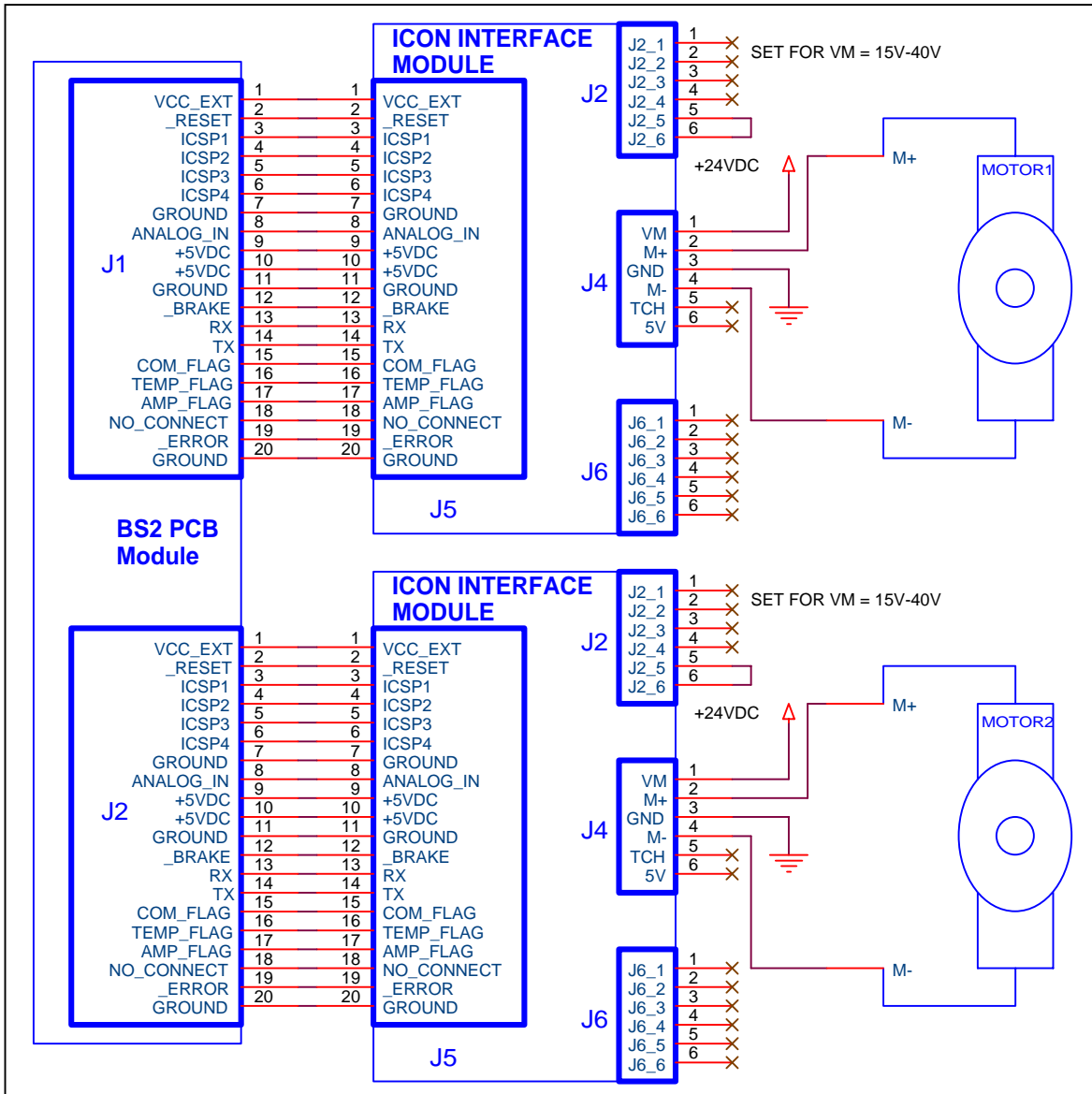
Figure 1: BS2 Connectivity to 2 ICON Interface Modules



Connectivity to each ICON Interface Module is accomplished with a ribbon cable (CW Industries PN: C3AAT-2006G) available through Digi-Key. The shrouded 20-pin headers on the BS2 module match those on the ICON Interface Modules (Tyco-Amp PN: 103308-5) which is also available from Digi-Key.

It is assumed that power is supplied to the BS2 module from the ICON Interface Module #1, that both ICON Interface Modules are being powered by a 24VDC supply, and that J2 on each ICON Interface Module has its jumper selected appropriately.

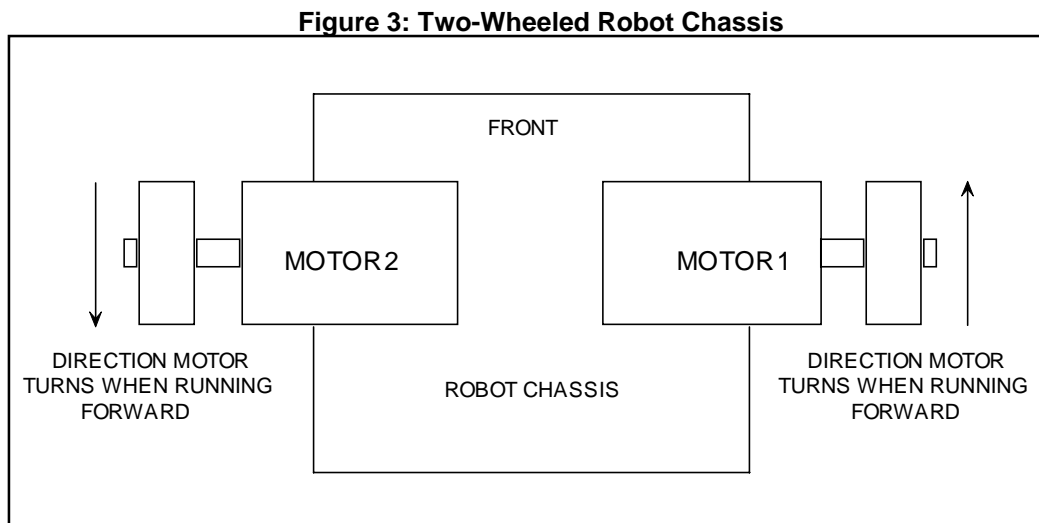
Figure 2: ICON Interface Module and BS2 PCB Module Connectivity Diagram



For motor voltages between 10VDC and 15VDC the voltage selection header (J2) on each ICON Interface Module should be jumped at J2_3 and J2_4. For motor voltages less than 10VDC the user will have to supply an external 12VDC signal at the BS2 module and place a jumper across J2_1 a J2_2 of J2 on each ICON Interface Module. The 12VDC signal would then be connected to the VCC_EXT_1 and VCC_EXT_2 connection on the BS2.

Software:

AN600 was designed to display a simple method for controlling a two-wheeled robot. It is assumed that the software would have to be modified to account for the particular idiosyncrasies of any mechanical systems. The software describes a method that may be used to ramp motor speed up or down to prevent over current conditions that can occur by rapidly changing speeds in large motors. It also describes methods for implementing slow and fast turns, and reversing a two-wheeled robot. The software is relatively simple and well commented. But for clarification a diagram of the envisioned mechanical system is included here.



Some aspects of the software require explanation. When implementing a command and waiting for a response from the ICON Interface Module the SERIN command is loaded with a 150ms timeout. In most cases the ICON Interface Module will respond within 5-10ms to a command. But the ICON Interface Module is regularly polling the ICON H-Bridge (about 5 times per second) to check the ICON H-Bridge's status. The communication between the ICON H-Bridge and the ICON Interface Module occurs at 2400BPS. If the ICON Interface Module is in the process of communicating with the ICON H-Bridge it may not be able to reply to a command for up to 150ms. You may reduce this timing or eliminate waiting for an ACK to speed up your program flow. If this is the route you take you should periodically check the status of each ICON Interface Module to verify that it is operating. This can be done with the READ_STATUS subroutine that is included in the software listing but is not called during program run time as AN600 is currently written.

If modifying AN600 for use with a BS2SX you should modify the BAUD constant to decimal 45, and change the timeout used with the SERIN commands to 500 (from 150). The software used for this application note (AN600.BS2) may be downloaded at www.solutions-cubed.com.

Finally, when interfacing to a large motor the resulting voltage and current spikes can destroy electronic circuits or reset processors. It is usually a good idea to avoid dramatic changes in motor speed or direction. If you find you have spurious failures check your power supply for glitches and try ramping motor speeds up and down to avoid dramatic speed changes.

AN600.BS2 Software Listing

'AN600 Interfacing a BASIC Stamp 2 to 2 ICON Interface Modules

'Communication string variables

CMMD	VAR	BYTE	'Command byte storage
ADDR	VAR	BYTE	'Address byte storage
LENG	VAR	BYTE	'Length byte storage
CKSUM	VAR	BYTE	'Checksum byte storage
DAT1	VAR	BYTE	'Data byte registers
DAT2	VAR	BYTE	
DAT3	VAR	BYTE	
DAT4	VAR	BYTE	
DAT5	VAR	BYTE	
DAT6	VAR	BYTE	
DAT7	VAR	BYTE	
DAT8	VAR	BYTE	
ERROR1	VAR	BYTE	'Communication error counters
ERROR2	VAR	BYTE	
LOOP	VAR	WORD	'For next loop word
SPEED	VAR	WORD	'Speed value storage
STRT_SPD	VAR	WORD	'Start speed used in ramp up and ramp down
END_SPD	VAR	WORD	'End speed used in ramp up and ramp down

'PWM storage registers

PWM_REG1	VAR	WORD	'PWM storage register for ICON Module 1
P1HI	VAR	PWM_REG1.HIGHBYTE	
P1LO	VAR	PWM_REG1.LOWBYTE	
PWM_REG2	VAR	WORD	'PWM storage register for ICON Module 2
P2HI	VAR	PWM_REG2.HIGHBYTE	
P2LO	VAR	PWM_REG2.LOWBYTE	

'Program constants

BAUD	CON	45	'Use BAUD = 45 for BS2SX
BAUD	CON	6	'Use BAUD = 6 for BS2

'ICON Interface Module #1 control lines

RESET1	CON	0	'Causes hardware reset when pulled low
BRAKE1	CON	1	'Implements braking function when pulled low
DOUT1	CON	2	'TTL serial data to ICON Interface Module
DIN1	CON	3	'TTL serial data from ICON Interface Module

'ICON Interface Module #2 control lines

RESET2	CON	4	'Causes hardware reset when pulled low
BRAKE2	CON	5	'Implements braking function when pulled low
DOUT2	CON	6	'TTL serial data to ICON Interface Module
DIN2	CON	7	'TTL serial data from ICON Interface Module

'Set BS2SX i/o direction and level

DIRS	=%0000000001110111	'Set P0,P1,P2,P4,P5,P6 as outputs
OUTS	=%1111111111111111	'Set all outputs high
LOW	RESET1	'Reset ICON Interface Modules
LOW	RESET2	
PAUSE	5	
HIGH	RESET1	
HIGH	RESET2	
PAUSE	750	'Wait 750ms for Interface Modules to power up
DEBUG	CLS	'Clear debug screen
GOSUB	INIT_IM	'Initialize the ICON Interface Modules

START:

'The "ramp-up forward" portion of this code ramps up the speed of both motors. This technique
'may be used to increase motor speed in large motors without abruptly changing motor speed
'and therefore tripping the over current fuse in either of the attached ICON H-Bridges.

RAMP_UP_FORWARD:

```

DEBUG          CR,"FORWARD RAMP UP",CR
STRT_SPD       =$0
END_SPD        = $3E8
FOR            SPEED = STRT_SPD TO END_SPD STEP ((END_SPD - STRT_SPD)/10)
              GOSUB FORWARD
              PAUSE 20
NEXT
PAUSE          2500
    
```

'The "ramp-down forward" portion of this code ramps down the speed of both motors. This technique
'may be used to decrease motor speed in large motors without abruptly changing motor speed
'and therefore tripping the over current fuse in either of the attached ICON H-Bridges.

RAMP_DN_FORWARD:

```

DEBUG          CR,"FORWARD RAMP DOWN",CR
STRT_SPD       =$3E8
END_SPD        = $0
FOR            SPEED = STRT_SPD TO END_SPD STEP ((STRT_SPD - END_SPD)/10)
              GOSUB FORWARD
              PAUSE 20
NEXT
PAUSE          500
SPEED          = $200                                'Set motor speed to 50%

DEBUG          CR,"BACKWARD",CR
GOSUB          BRAKE                                  'Brake motors before changing direction
GOSUB          BACKWARD                              'Run robot in reverse
PAUSE          2000

DEBUG          CR,"SLOW LEFT",CR
GOSUB          BRAKE                                  'Brake motors before changing direction
GOSUB          SLOW_LEFT                             'Perform a slow left turn
PAUSE          2000

DEBUG          CR,"SLOW RIGHT",CR
GOSUB          SLOW_RIGHT                             'Perform a slow right turn
PAUSE          2000

DEBUG          CR,"FAST LEFT",CR
GOSUB          FAST_LEFT                             'Make a fast left turn
PAUSE          2000                                'Stop left motor (2) and run right motor (1)

DEBUG          CR,"FAST RIGHT",CR
GOSUB          FAST_RIGHT                             'Make a fast right turn
PAUSE          2000                                'Stop right motor (1) and run left motor (2)

LOW            BRAKE1
LOW            BRAKE2
PAUSE          2000                                'Assert brake pins to stop the motors from turning
HIGH          BRAKE1
HIGH          BRAKE2

GOTO          START                                'Return to start of program
    
```

```

***** Subroutines *****
*****
'INIT_IM:      This subroutine initializes the ICON Interface Modules by ensuring that the
'              ADDRESS registers are programmed to "1" and programming the IM_FUNCTION
'              registers to binary %10100000. The IM_FUNCTION value enables dynamic
'              braking and amps retry settings (see the communication protocol for
'              more information on these functions). The ADDRESS and IM_FUNCTION registers
'              are written to using the universal address of "0", therefore no ACK will be
'              be received from the ICON Interface Modules. The BS2 then sends the STORE
'              command to each ICON Interface Module and waits for the ACK. This process
'              occurs every time the circuit is powered up, but in reality it only needs to
'              occur once since the settings are stored in EEPROM with the STORE command.
*****
INIT_IM:
    CMMD      = $D2                'WRITE command
    ADDR      = $00                'Universal address
    LENG      = $04                'Message length
    DAT1      = $0E                'ADDRESS register index value
    DAT2      = $01                'Write "1" to ADDRESS register
    DAT3      = $0F                'IM_FUNCTION register index value
    DAT4      = $A0                'Write %1010000 to IM_FUNCTION register
    CKSUM     = CMMD+ADDR+LENG+DAT1+DAT2+DAT3+DAT4
    SEROUT    DOUT1,BAUD,[CMMD,ADDR,LENG,DAT1,DAT2,DAT3,DAT4,CKSUM]
    SEROUT    DOUT2,BAUD,[CMMD,ADDR,LENG,DAT1,DAT2,DAT3,DAT4,CKSUM]

    PAUSE     20

    CMMD      = $D3                'STORE command
    ADDR      = $01                'ICON Interface Module address of "1"
    LENG      = $00                'Length of 0, no data in command
    CKSUM     = CMMD+ADDR+LENG
    SEROUT    DOUT1,BAUD,[CMMD,ADDR,LENG,CKSUM]
    SERIN     DIN1,BAUD,150,NA_INIT1,[DAT1]
    IF        DAT1 = $6 THEN A_INIT1      'Wait 150ms for an ACK from module 1
NA_INIT1:
    DEBUG     "NO ACK INIT1",CR
A_INIT1:
    SEROUT    DOUT2,BAUD,[CMMD,ADDR,LENG,CKSUM]
    SERIN     DIN2,BAUD,150,NA_INIT2,[DAT1]
    IF        DAT1 = $6 THEN A_INIT2      'Wait 150ms for an ACK from module 2
NA_INIT2:
    DEBUG     "NO ACK INIT2",CR
A_INIT2:
    RETURN

*****
'SETDC_IM1:   This routine sends speed and direction data to the ICON Interface Module
'             number one located on the right hand side of the robot. The speed and
'             direction data are stored in the PWM_REG1 register. If an ACK is not
'             received within 150ms then the BS2 will attempt to send the command again.
'             This retry will be attempted up to 5 times.
*****
SETDC_IM1:
    DEBUG     "PWM1 = ",ISHEX4 PWM_REG1,TAB
    CMMD      = $D0                'SETDC command
    ADDR      = $01                'ICON Interface Module address of "1"
    LENG      = $02                'Length of SETDC is 2
    CKSUM     = CMMD+ADDR+LENG+P1LO+P1HI
    SEROUT    DOUT1,BAUD,[CMMD,ADDR,LENG,P1HI,P1LO,CKSUM]
    SERIN     DIN1,BAUD,150,NA_SDC1,[DAT1]
    IF        DAT1 <> $6 THEN NA_SDC1
    ERROR1    = 0
    RETURN
NA_SDC1:
    ERROR1    = ERROR1+1
    IF        ERROR1 < 5 THEN SETDC_IM1
    ERROR1    = 0
    RETURN

```

```

*****
'SETDC_IM2: This routine sends speed and direction data to the ICON Interface Module
'           number two located on the left hand side of the robot. The speed and
'           direction data are stored in the PWM_REG2 register. If an ACK is not
'           received within 150ms then the BS2 will attempt to send the command again.
'           This retry will be attempted up to 5 times.
*****

SETDC_IM2:
  DEBUG          "PWM2 = ",ISHEX4 PWM_REG2,CR
  CMMD           = $D0                      'SETDC command
  ADDR           = $01                      'ICON Interface Module address of "1"
  LENG          = $02                      'Length of SETDC is 2
  CKSUM         = CMMD+ADDR+LENG+P2LO+P2HI
  SEROUT        DOUT2,BAUD,[CMMD,ADDR,LENG,P2HI,P2LO,CKSUM]
  SERIN         DIN2,BAUD,150,NA_SDC2,[DAT1]
  IF            DAT1 <> $6 THEN NA_SDC2
  ERROR2        = 0
  RETURN

NA_SDC2:
  ERROR2        = ERROR2+1
  IF            ERROR2 < 5 THEN SETDC_IM2
  ERROR2        = 0
  RETURN

*****
'FORWARD: This subroutine runs motor one in the forward direction and motor two at
'          the same speed but in the reversed direction. This should move the robot
'          forward.
*****

FORWARD:
  PWM_REG1      = SPEED
  PWM_REG2      = -SPEED
  GOSUB         SETDC_IM1
  GOSUB         SETDC_IM2
  RETURN

*****
'BACKWARD: This subroutine runs motor two in the forward direction and motor one at
'          the same speed but in the reversed direction. This should move the robot
'          backward.
*****

BACKWARD:
  PWM_REG1      = -SPEED
  PWM_REG2      = SPEED
  GOSUB         SETDC_IM1
  GOSUB         SETDC_IM2
  RETURN

*****
'SLOW_LEFT: This subroutine runs motor one forward and motor two at half the speed of
'           motor one in reverse. This will cause the robot to veer to the left.
*****

SLOW_LEFT:
  PWM_REG1      = SPEED
  PWM_REG2      = -(SPEED/2)
  GOSUB         SETDC_IM1
  GOSUB         SETDC_IM2
  RETURN

```

```

*****
'SLOW_RIGHT: This subroutine runs motor two reversed and motor one at half the speed of
'            motor two in the forward direction. This will cause the robot to veer to
'            the right.
*****

SLOW_RIGHT:
    PWM_REG1    = SPEED/2
    PWM_REG2    = -SPEED
    GOSUB       SETDC_IM1
    GOSUB       SETDC_IM2
    RETURN

*****
'FAST_LEFT: This subroutine runs motor one in the forward direction and stops motor two
'            by sending it a speed of $0000. This causes the robot to pivot to the left.
*****

FAST_LEFT:
    PWM_REG1    = SPEED
    PWM_REG2    = $0
    GOSUB       SETDC_IM1
    GOSUB       SETDC_IM2
    RETURN

*****
'FAST_RIGHT: This subroutine runs motor two in the reversed direction and stops motor
'            one by sending it a speed of $0000. This causes the robot to pivot to the
'            right.
*****

FAST_RIGHT:
    PWM_REG1    = $0
    PWM_REG2    = -SPEED
    GOSUB       SETDC_IM1
    GOSUB       SETDC_IM2
    RETURN

*****
'READ_STATUS: This subroutine reads the IM_STATUS, and IH_STATUS registers from both
'            ICON Interface Modules and displays the information in a binary format.
*****

READ_STATUS:
    CMMD        = $D1
    ADDR        = $01
    LENG        = $02
    DAT1        = $00
    DAT2        = $01
    CKSUM       = CMMD+ADDR+LENG+DAT1+DAT2
    SEROUT      DOUT1,BAUD,[CMMD,ADDR,LENG,DAT1,DAT2,CKSUM]
    SERIN       DIN1,BAUD,150,NA_RDST1,[DAT1,DAT2,DAT3,DAT4]
    DEBUG       "IM1 STATUS =",BIN8 DAT3," IH1 STATUS =",BIN8 DAT4,CR
    GOTO        RDST2

NA_RDST1:
    DEBUG       "NO ACK READ STATUS 1",CR

RDST2:
    DAT1        = $00
    DAT2        = $01
    CKSUM       = CMMD+ADDR+LENG+DAT1+DAT2
    SEROUT      DOUT2,BAUD,[CMMD,ADDR,LENG,DAT1,DAT2,CKSUM]
    SERIN       DIN2,BAUD,150,NA_RDST2,[DAT1,DAT2,DAT3,DAT4]
    DEBUG       "IM2 STATUS =",BIN8 DAT3," IH2 STATUS =",BIN8 DAT4,CR
    GOTO        DONE_RDST2

NA_RDST2:
    DEBUG       "NO ACK READ STATUS 2",CR

DONE_RDST2:
    RETURN

```

```
*****  
'BRAKE:      This subroutine asserts the /BRAKE pins for both ICON Interface Modules  
'           waits 250ms and then releases the /BRAKE pins.  
*****  
  
BRAKE:      LOW           BRAKE1  
            LOW           BRAKE2  
            PAUSE        250  
            HIGH         BRAKE1  
            HIGH         BRAKE2  
            RETURN  
  
END:
```

Summary:

As can be seen from AN600 the ICON Interface Module and ICON H-Bridge can simplify connectivity and control of brushed DC motors with the BASIC Stamp. This application note used only the simplest of the ICON Interface Module's capabilities. Additionally, with the 12A continuous current handling capability of the ICON DC Motor Control system larger robotic systems can be envisioned with little development time and at a reduced cost.