

MOTOROLA
SEMICONDUCTOR
APPLICATION NOTE

ANE405
Replaces ANE005

**Bi-directional Data Transfer between
MC68HC11 and MC6805L3 using SPI**

by
Richard Soja, Motorola, EKB

INTRODUCTION

One of the most powerful features shared by a wide range of Motorola MCUs is the Serial Peripheral Interface (SPI). It is primarily designed to operate as a synchronous, 8-bit communication system and is implemented entirely with on-chip hardware. This frees the CPU for other tasks and ensures a minimum of software overhead associated with the SPI system.

The SPI is available in two basic forms:

1. Level 1 SPI — implemented on the MC68HC11, HC05C4 MCUS,
2. Level 2 SPI — implemented on the MC6805S2/S3/L3/L8 MCUs.

Note that the HCMOS family of MCUs only support level 1, while level 2 is implemented only on HMOS MCUs.

Though both levels of SPI can communicate easily with each other, level 2 has a number of additional capabilities, including asynchronous communication. This application note is aimed at describing a method of achieving synchronous communication between a level 1 and level 2 SPI, and details the subtle relevant differences in the on-chip implementation of each.

DESCRIPTION

The two MCUs used in this application are the high performance MC68HC11 and the low cost MC6805L3.

Data is transferred between the MCUs on a single

bi-directional line, with the clock supplied on an additional line. Also, to ensure initial synchronisation between each MCU, a software handshake sequence is implemented on the same lines which provide the clock and data. This has the considerable advantage of minimising the number of lines between each MCU as additional control lines are not needed.

The handshake sequence is also necessary for two other reasons; the 6805L3 receive data register is unbuffered, and it is not possible for the 68HC11 to stop transmission of data from the 68HC11 by inhibiting the clock signal. The fact that the 6805L3 data register is unbuffered means that, if a handshake sequence was not implemented, new data could begin to get clocked in before the previous data were read. Also, the on-chip configuration of the 68HC11's SPI means that, if an attempt were made to slow down or stop its clock during transfer of a byte, there would be a resultant loss of synchronism between the transmitting and receiving MCUs.

The 68HC11 software is implemented as the clock master (i.e. it provides the clock output), while the 6805L3 is the clock slave. As there are no other clock masters or slaves in the system, software is kept to an absolute minimum. In fact the main transfer routine (XFER) for the 68HC11 is only 27 bytes long, while the 6805L3 uses only 30 bytes.

The other significant advantage of this implementation is that none of the 6805L3 timers are required for SPI operation, thus ensuring a minimal impact on any other application dependent tasks the MCU may be executing.

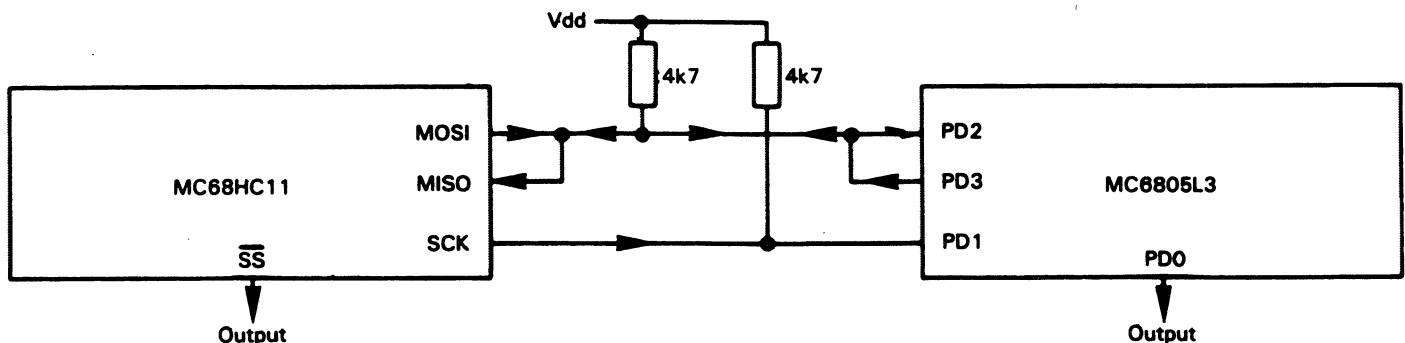
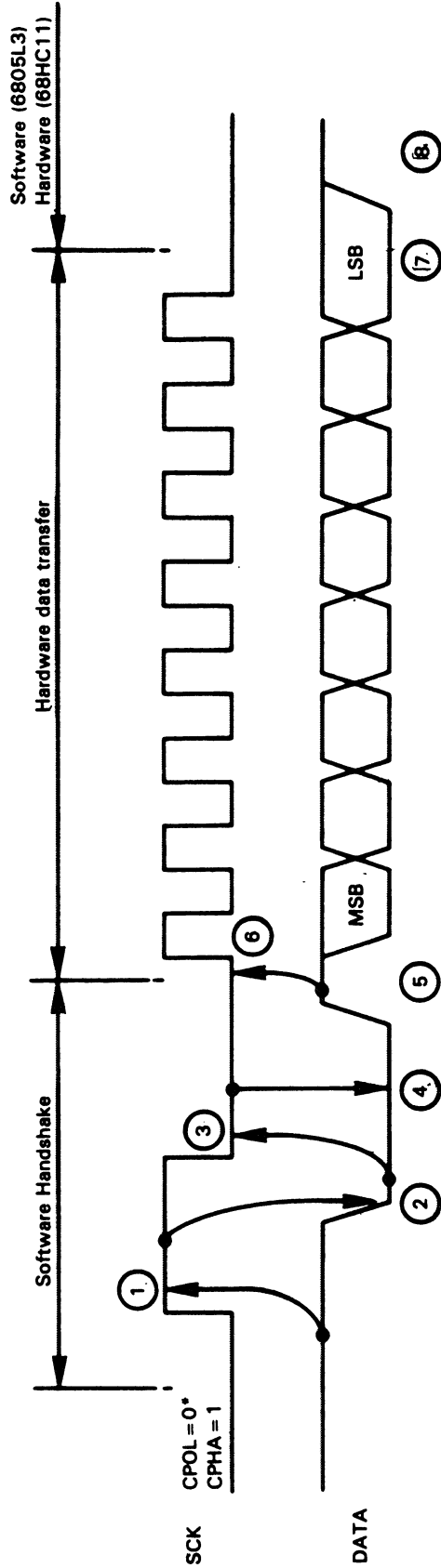


Figure 1. Hardware Implementation



Figure 2. 68HC11-6805L3 SPI Timing



*Clock control bits for 68HC11

- 1| Master releases clock line by disabling SPI.
- 2| Slave clears data line by forcing o/p clamp on PD3.
- 3| Master clears clock line by enabling SPI.
- 4| Once clock line goes low, slave stores data in SPI data register, sets its data DDR to correct state, and enables SPI.
- 5| Slave releases data clamp.
- 6| Master starts SPI clock by storing data in SPI register.
- 7| Both master and slave detect end of transmission and read data from SPI register.
- 8| Slave disables SPI to ensure data line is released.

Configuring the MCUs

The method of configuring each MCU for bi-directional data transfer is slightly different, due to the differences in their SPI silicon implementation. Figure 1 shows the hardware implementation. On the 68HC11, the input and output pins must be connected together externally. On the 6805L3, this can be done internally by software, thus requiring only 1 external data I/O pin. The other 6805L3 SPI data pin is now free to be used in any other way.

As the slave select (SS) pins are unused by either MCU, they must be configured as outputs to prevent SPI fault conditions occurring.

The spare 6805L3 data pin (PD3) is used to control the bi-directional data line, thus providing a handshake signal to the 68HC11. The 68HC11's handshake is on the clock line, and is controlled by disabling and enabling its SPI.

Data transfer and timing

Figure 2 shows details of the handshake sequence. The significant point to note from Figure 2 is that the handshake sequence is implemented purely in software, while the 8-bit data and clocks are generated by the SPI hardware.

To prevent data contention, both during data transfer and during the handshake sequence, both SPIs must operate in wired-or (open drain) mode. On the 6805L3, this is done by setting bit 3 in the miscellaneous register, while on the 68HC11, bit 5 of the SPI control register must be set. The SPI utilised on the 68HC05 family of MCUs does not support this open-drain option and cannot, therefore, be used in this application.

The clock format is: idle low, data output on positive edge and sampled on negative edge. Both SPIs must be configured to operate with the same clock format (see Figure 2). Eight data bits are transferred, at a maximum clock rate of 125 kHz,

which is limited by the 6805L3 SPI. Data transfer is preceded by the handshake sequence, which ensures that both MCUs are in the correct state, and ready to transfer a new byte of data.

The most significant bit of data appears first, on the rising edge of the first clock. Data is latched into both SPI shift registers on the falling edge of the clock. Once the last data bit is latched, the data line is released high. This is necessary to ensure correct operation of the handshake sequence. When the 68HC11 is acting as a transmitter, this state occurs automatically — as a by product of its SPI hardware implementation. However, on completion of data transmission from the 6805L3, the LSB is permanently maintained on the data line, so its driver routine has been designed to ensure that the data line is always restored to the high state.

Software routines

A glance at the software listing (see Appendix 1) reveals that the transmit and receive routines for each MCU are essentially the same! The entry and exit conditions of each are as shown in table 1.

On the 6805L3, the X register dictates the operating mode of the transfer routine. This is necessary because the same I/O pin is used for transmitting and receiving data, so its data direction register must be changed appropriately (by the contents of X).

On the 68HC11, separate pins are used for transmitting and receiving data, so their DDR pins are set up once only in the initialisation routine.

There are other important subtle differences. Prior to reception of data, the 6805L3 SPI data register content is irrelevant, while the 68HC11 SPI data register must be loaded with \$FF, to prevent data bus contention. This could occur with the 68HC11, in this application, because data is simultaneously output to and read back on the same external line

Table 1

68HC11:

	<u>Transmit</u> ACCA	<u>Receive</u> ACCA	
Entry	Data to send	\$FF	X Reg = base address of I/O Register block (normally \$1000).
Exit	Data sent	Data received	X Reg = unchanged

All other registers are unused by the 68HC11 transfer routine.

6805L3:

	Transmit		Receive	
	ACC	X Reg	ACC	X Reg
Entry	Data to send	\$5	Don't care	\$1
Exit	Data sent	\$D	Data received	\$D

during data transfer. Loading \$FF into the 68HC11 data register ensures that this data is replaced by that transmitted from the 6805L3.

Note that, as the 68HC11 is the clock master, it must provide the clock signal, not only when transmitting data, but also when receiving data from the 6805L3. It does this by writing to its SPI data register.

Completion of data transfer is indicated by a single flag bit (SPIF). On the 6805L3, this is bit 7 of the SPI control register, while on the 68HC11, it is bit 7 of the SPI status register. This flag bit is used to indicate completion of either transmission or reception of data.

Flag clearing techniques are quite different for the 68HC11 and 6805L3. On the latter, the SPIF flag is cleared simply by writing '0' to the flag bit. On the 68HC11, a two stage operation is required to clear the SPIF flag; the SPI status register must first be read, with the flag set, followed by an access of the SPI data register.

An examination of the SPI software drivers shows that on completion of a data transmission, the SPI data register is read again. This data should be the same as that transmitted, and provides information on whether data contention or corruption occurred during transfer. This facility could be incorporated in a data validation routine to improve reliability of data transfer.

The key features in this implementation are:

1. An orderly start-up sequence to ensure the correct initial synchronisation. As the 6805L3 is the slave, its initialisation routine is not exited until it detects a low on the clock line — this will occur only when the 68HC11 gains control of the SPI. Before this happens, all I/O pins are set to inputs, so the clock line will be pulled high by the external resistor.
2. A well defined transfer protocol is used. The master device (i.e. 68HC11) must always dictate the data transfer direction and the data stream size must be specified by the currently selected transmitter, so that the receiver knows when the last byte has been sent.

The transfer protocol operates such that after initialisation, the master MCU (68HC11) transmits a control byte to the slave (6805L3). This control byte selects the subsequent data transfer direction and is either a slave listen address or a slave talk address. If it is a slave listen address, then the 6805L3 stays in receive mode. The next byte indicates the total number of bytes to be received, followed by the data stream (see Figure 3).

Once the last byte is transferred, the 6805L3 can await a new control byte, or alternatively it can return to some other task, such as processing the previously received data. Similarly, at this point, the

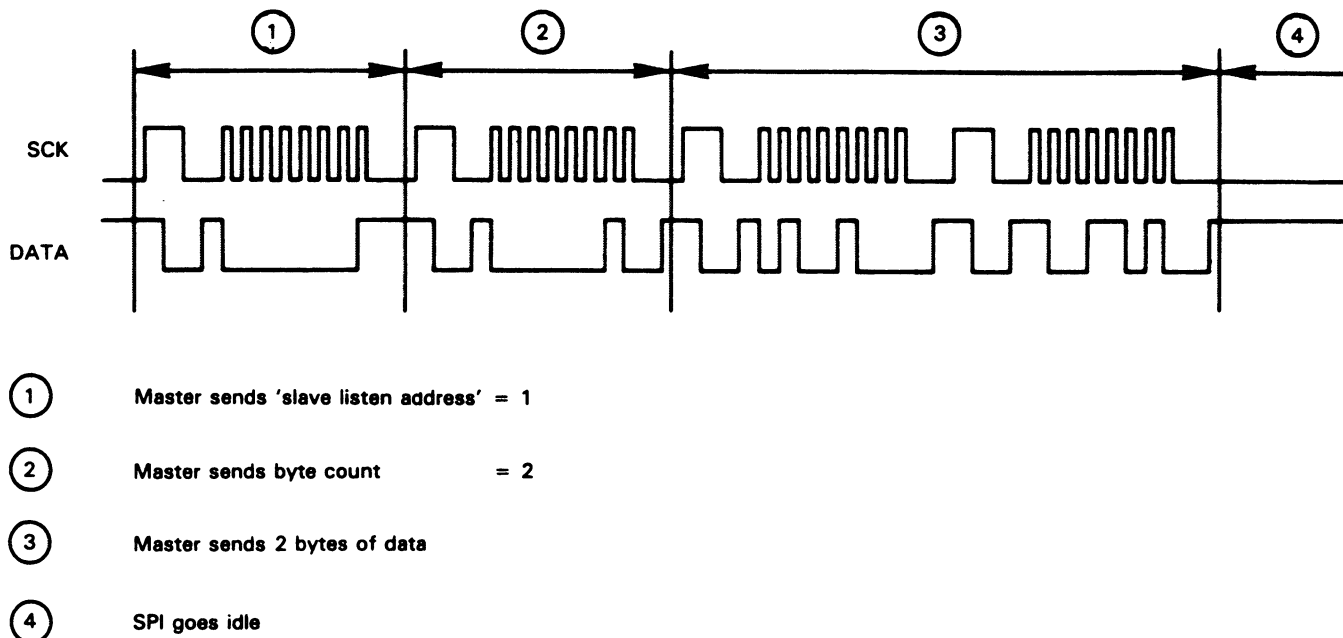


Figure 3. Master Transmitter — Slave Receiver

master transmitter (68HC11) can return to another task, or send a new control byte.

If the control byte now sent is a slave talk address, then the 6805L3 will switch to transmit mode, and the master will switch to receive mode. The 6805L3 will send a byte count, followed by the data stream to the master (see Figure 4).

Note that, once the last byte is transferred in either direction, both processors are free to continue other tasks or attempt a new data transfer. The handshake sequence always ensures synchronisation of data transfer, independent of the response time of either MCU.

CONCLUSION

Potential uses for this type of data transfer are in applications which require remote interrogation of an

MCU based system via a minimal number of lines, such as:

- Development and diagnosis of engine management systems
- Smart card and key card security applications
- Instrumentation and data logging equipment.

APPENDIX

The demonstration programs listed in the following pages simply transfer a string of characters from the 68HC11 to the 6805L3, which converts them to upper case and sends them back again.

The HC11SPI program is listed on pages 6 to 7; the L3SPI program on pages 8 to 10.

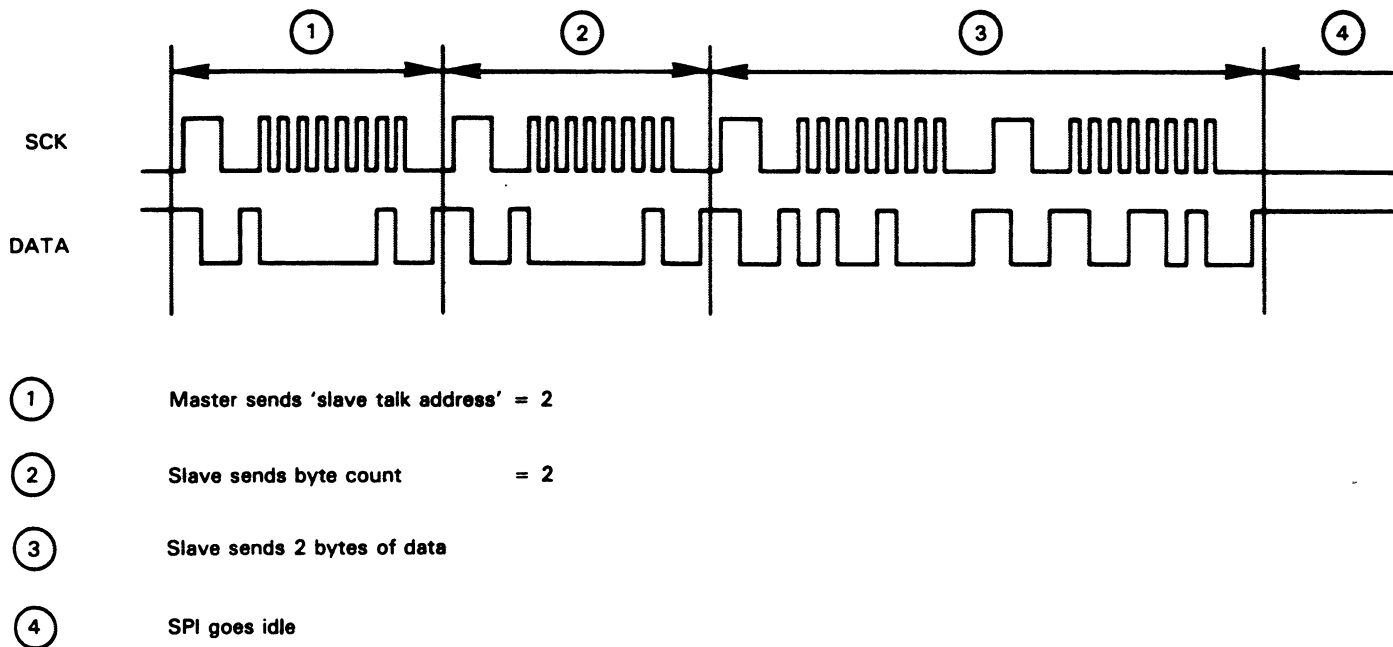


Figure 4. Master Receiver — Slave Transmitter

```

1 P *****
2 P *
3 P *           HC11SPI 4/9/86           *
4 P *
5 P * SPI TRANSFER WITH HANDSHAKE ON SPI DATA AND *
6 P * CLOCK LINES. THIS IS NECESSARY AS HC11'S CLOCK *
7 P * CANNOT BE SLOWED BY SLAVE DEVICE. *
8 A      0008 PORTD EQU 8
9 A      0009 DORD EQU 9
10 A     0028 SPCR EQU $28
11 A     0029 SPSR EQU $29
12 A     002A SPDR EQU $2A
13 A     0024 TMSK2 EQU $24
14 A     0025 TFLG2 EQU $25
15 P *
16 A     0004 MISO EQU 4
17 A     0008 MOSI EQU 8
18 A     0010 SCK EQU $10
19 A     0080 SPIF EQU $80
20 A     0040 SPE EQU $40
21 P *
22 A     0000 TIMCOUNT ORG $0
23 A 0000 0001 TIMCOUNT RMB 1
24 A 0001 1C MSG1 FCB 28
25 A 0002 62692D6469 MSG1 FCB /BI-DIRECTIONAL DATA TRANSFER/
26 A *
27 A     C000 START ORG $C000
28 A C000 8E0035 START LDS #$35
29 A C003 8D3C BSR INIT
30 A C005 18CE0001 START1 LDY #MSG1
31 A C009 8D08 BSR
32 A C00B 18CE0001 LDY #MSG1
33 A C00F 8D17 BSR READ
34 A C011 20F2 BRA START1
35 A *
36 A     C013 SEND EQU *
37 A C013 8601 LDAA #1
38 A C015 8D3A BSR XFER
39 A C017 18A600 LDAA .Y
40 A C01A 16 TAB
41 A C01B 8D34 BSR XFER
42 A C01D 1808 SEND1 INY
43 A C01F 18A600 LDAA .Y
44 A C022 8D2D BSR XFER
45 A C024 5A DECB
46 A C025 26F6 BNE SEND1
47 A C027 39 RTS
48 A *
49 A     C028 READ EQU *
50 A C028 8602 LDAA #2
51 A C02A 8D25 BSR XFER
52 A C02C 86FF LDAA #$FF
53 A C02E 8D21 BSR XFER
54 A C030 16 TAB
55 A C031 18A700 STAA .Y
56 A C034 1808 READ1 INY
57 A C036 86FF LDAA #$FF
58 A C038 8D17 BSR XFER

```

NUMBER OF BYTES IN DATA BLOCK
/BI-DIRECTIONAL DATA TRANSFER/

LOAD INTO EVB RAM
BUFFALO USES RAM ABOVE THIS

Y POINTS TO BEGINNING OF DATA BLOCK TO TRANSFER
TRANSMIT BLOCK TO SLAVE, STARTING WITH BYTE COUNT
RE-INITIALISE POINTER TO DATA BLOCK
AND READ THE BYTES BACK.

ENTER WITH Y POINTING AT BYTE COUNT OF DATA BLOCK
COMMAND SLAVE TO RECEIVE

NOW SEND SLAVE THE BYTE COUNT,
BUT 1ST STORE IT IN BYTE COUNTER.

POINT AT NEXT BYTE

AND SEND IT
UNTIL ALL DONE

ENTER WITH Y POINTING AT BYTE COUNT OF DATA BLOCK.
COMMAND SLAVE TO TRANSMIT

NOW READ BYTES BACK FROM SLAVE.
1ST BYTE IS BYTE COUNT SO
STORE IT IN BYTE COUNTER
THEN STORE IT AT BEGINNING OF DATA BLOCK.
BUMP POINTER

THEN CONTINUE TO READ

```

59 A C03A 18A700      STAA    .Y          AND STORE BYTES
60 A C03D 5A         DECB
61 A C03E 26F4       BNE     READ1      UNTIL ALL DONE
62 A C040 39         RTS
63 A
64 A      C041      *
65 A C041 CE1000     EQU     INIT      EQU     *
66 A C044 8638       LDAA   LDX        #$1000
67 A C046 A709       LDAA   LDAA       #$38
68 A C048 8676       STAA   STAA       DDDR,X   ENABLE OUTPUT MODE ON SS,SCK,MOSI, INPUT ON MISO
69 A C04A A728       LDAA   LDAA       #$76   (WITH SS AS OUTPUT, MODF IS DISABLED.)
70 A C04C 8618       STAA   STAA       SPCR,X   ENABLE SPI AS 125KHZ CLOCK MASTER, WIRED-OR MODE
71 A C04E A708       LDAA   LDAA       #$18
72 A C050 39         STAA   STAA       PORTD,X  SET DATA & CLOCK OUTPUT BUFFERS TO LOGIC '1'
73 A                RTS        TO GENERATE ACKNOWLEDGE CLOCK.
74 A                *
75 A                *****
76 A                *      BIDIRECTIONAL DATA TRANSFER, WITH      *
77 A                *      HANDSHAKE ON CLOCK & DATA LINES.      *
78 A                *      ENTERED WITH DATA IN ACCA, SPI ENABLED,  *
79 A                *      CLOCK LOW, DATA LINE DETERMINED BY SLAVE DATA PIN.*
80 A                *
81 A C051 1F0804FC   EQU     XFER      EQU     *
82 A C055 1D2840     BRCLR  PORTD,X,#MISO,* 1ST WAIT FOR SLAVE TO RELEASE DATA LINE.
83 A C058 1E0804FC   BCLR   SPCR,X,#SPE     SEND ACK ON CLOCK LINE, BY DISABLING SPI
84 A C05C 1C2840     BRSET  PORTD,X,#MISO,*  WAIT FOR SLAVE TO ACKNOWLEDGE.
85 A C05F 1F0804FC   BSET   SPCR,X,#SPE     ENABLE SPI, CLEARING CLOCK LINE.
86 A C063 A72A       BRCLR  PORTD,X,#MISO,*  WAIT FOR SLAVE TO RELEASE DATA LINE,THEN
87 A C065 1F2980FC   STAA   SPDR,X         START SPI TRANSACTION BY WRITING DATA.
88 A C069 A62A       BRCLR  SPSR,X,#SPIF,*  NOW WAIT FOR TRANSMISSION COMPLETE FLAG.
89 A C06B 39         LDAA   SPDR,X         CLEAR SPIF, AND READ DATA
90 A                RTS        BEFORE RETURNING.
91 A                *
92 A                *      ORG $FFFE
93 A                *      FDB START
94 A                *
                    END
***** TOTAL ERRORS      0--  0
***** TOTAL WARNINGS   0--  0

```

SYMBOL TABLE LISTING

SYMBOL NAME	SECT	VALUE	SYMBOL NAME	SECT	VALUE
DDRD	A	0009	SPCR	A	0028
INIT	A	C041	SPDR	A	002A
MISO	A	0004	SPE	A	0040
MOSI	A	0008	SPIF	A	0080
MSG1	A	0001	SPSR	A	0029
PORTD	A	0008	START	A	C000
READ	A	C028	START1	A	C005
READ1	A	C034	TFLG2	A	0025
SCK	A	0010	TIMCOUNT	A	0000
SEND	A	C013	TMSK2	A	0024
SEND1	A	C01D	XFER	A	C051

```

1 P
2 P
3 P
4 P
5 P
6 P
7 P
8 P
9 A      0000   PORTA   EQU    0
10 A     0003   PORTD   EQU    3
11 A     0004   DDRA    EQU    4
12 A     0007   DDRD   EQU    7
13 A     0008   TADAT  EQU    8
14 A     0009   TACR   EQU    9
15 A     000A   MISC   EQU   $A
16 A     000C   TBDAT  EQU   $C
17 A     000D   TBCR   EQU   $D
18 A     000E   SPIDAT EQU   $E
19 A     000F   SPICR  EQU   $F
20 A     0010   PRESCL EQU  $10
21 P
22 A     0000   SS     EQU    0
23 A     0001   SCK    EQU    1
24 A     0002   SDA    EQU    2
25 A     0003   CLAMP  EQU    3           DATA LINE CLAMP ON D3
26 A     0004   SPE    EQU    4
27 A     0007   SPIF   EQU    7
28 P
29 A     0020   BYTCNT ORG   $20
30 A 0020 0001   RMB    1
31 A 0021 0001   RMB    1
32 A 0022 0028   DATA  RMB   40           RESERVE ENOUGH SPACE FOR RECEIVED DATA.
33 A
34 A     0080   START  ORG   $80
35 A     0080   EQU   *
36 A 0080 AD3D   BSR    INIT           INITIALISE SPI, AND WAIT FOR HC11 TO BECOME READY
37 A 0082 4F    CLRA
38 A 0083 AE01   LDX    #1           1ST RECEIVE COMMAND BYTE
39 A 0085 AD4C   BSR    XFER
40 A 0087 A101   CMP    #1           IF SLAVE REQUESTED TO LISTEN, THEN
41 A 0089 2706   BEQ    READ         READ MORE DATA.
42 A 008B A102   CMP    #2           IF SLAVE REQUESTED TO TALK, THEN
43 A 008D 271A   BEQ    SEND         SEND DATA TO MASTER.
44 A 008F 20F2   BRA    START1
45 A
46 A     0091   READ   EQU   *
47 A 0091 AE01   LDX    #1           GET NEXT BYTE, WHICH IS BYTE COUNTER.
48 A 0093 AD3E   BSR    XFER
49 A 0095 B720   STA   BYTCNT
50 A 0097 B721   STA   DATLEN        ALSO STORE VALUE AS DATA LENGTH.
51 A 0099 AE01   LDX    #1           NOW READ ALL REMAINING BYTES
52 A 009B AD36   BSR    XFER
53 A 009D BE20   LDX    BYTCNT
54 A 009F A020   SLB    #$20         (CONVERT LOWER CASE TO UPPER CASE)
55 A 00A1 E722   STA   DATA,X       AND STORE IN BUFFER
56 A 00A3 3A20   DEC   BYTCNT
57 A 00A5 26F2   BNE   READ1
58 A 00A7 20DA   BRA   START1        UNTIL ALL DONE, AND RETURN.

```



```

59 A
60 A      00A9      * SEND EQU *
61 A      00A9 B621 LDA DATLEN
62 A      00AB B720 STA BYCNT GET LENGTH OF DATA
63 A      00AD AE05 LDX #5 AND STORE IT IN BYTE COUNTER.
64 A      00AF AD22 BSR XFER
65 A      00B1 BE20 SEND1 LDX BYCNT SEND BYTE COUNT TO MASTER.
66 A      00B3 E622 LDA DATA,X NOW GET NEXT BYTE TO SEND IN ACC
67 A      00B5 AE05 LDX #5
68 A      00B7 AD1A BSR XFER SEND IT TO MASTER
69 A      00B9 3A20 DEC BYCNT
70 A      00BB 26F4 BNE SEND1
71 A      00BD 20C4 BRA START1 UNTIL ALL DONE,
72 A * AND RETURN.
73 A *
74 A      00BF      * INIT EQU *
75 A      00BF A648 LDA #$48 INHIBIT INT2, ENABLE PORTD OPEN DRAIN.
76 A      00C1 B70A STA MISC
77 A      00C3 A60D LDA #$D SET DATA,SS & CLAMP O/P BUFFERS. CLEAR REST.
78 A      00C5 B703 STA PORTD NOTE: CLAMP ON D3
79 A      00C7 A609 LDA #9 SELECT SPI CLOCK SLAVE, D2 AS I/P, CLAMP & SS
80 A      00C9 B707 STA DDRD AS O/P. (SS O/P STOPS SPI RESETTING)
81 A      00CB A644 LDA #$44 DISABLE START BIT DETECTION, DATA I/O ON D2,
82 A      00CD B70F STA SPICR DATA SAMPLED ON -IVE CLOCK, SPI DISABLED.
83 A      00CF 0203FD BRSET SCK,PORTD,* MUST WAIT FOR MASTER TO GAIN CONTROL OF SPI.
84 A      00D2 81 RTS
85 A *
86 A * *****
87 A * BIDIRECTIONAL DATA TRANSFER ON SPI *
88 A * ENTERED WITH SPI DISABLED, DATA PIN HIGH* *
89 A * D2 PIN I/P, CLAMP PIN O/P (HIGH) *
90 A * ENTRY: TX MODE: ACCA=DATA, X=5 *
91 A * RX MODE: ACCA=X, X=1 *
92 A * EXIT: TX MODE: ACCA=TX DATA, X=$D *
93 A * RX MODE: ACCA=RX DATA, X=$D *
94 A *
95 A      00D3      * XFER EQU *
96 A      00D3 0303FD BRCLR SCK,PORTD,* WAIT FOR MASTER TO ACKNOWLEDGE ON CLOCK LINE.
97 A      00D6 1703 BCLR CLAMP,PORTD SEND ACKNOWLEDGE TO MASTER.
98 A      00D8 0203FD BRSET SCK,PORTD,* WAIT FOR MASTER TO ENABLE ITS SPI
99 A      00DB B70E STA SPIDAT NOW WRITE DATA TO SPI REGISTER,
100 A     00DD 180F BSET SPE,SPICR BEFORE ENABLING SPI, AND RELEASING CLAMP
101 A     00DF BF07 STX DDRD CLAMP PIN I/P: DATA EITHER I/P OR O/P.
102 A * TRANSFER STARTS NOW!
103 A     00E1 A644 LDA #$44 PREPARE TO CLEAR SPI FLAG, DISABLE SPI,
104 A     00E3 AE0D LDX #$D AND SEND ACKNOWLEDGE.
105 A     00E5 0F0FFD BRCLR SPIF,SPICR,* WAIT FOR DATA TO ARRIVE.
106 A     00E8 B70F STA SPICR DISABLE SPI TO ALLOW DATA PIN TO BE FORCED HIGH
107 A     00EA B60E LDA SPIDAT READ DATA.
108 A     00EC BF03 STX PORTD FORCE DATA PIN HIGH & RELEASE CLAMP
109 A     00F0 81 STX DDRD BY MAKING BOTH OUTPUTS.
110 A *
111 A * END

```

```


***** TOTAL ERRORS 0-- 0
***** TOTAL WARNINGS 0-- 0

```

SYMBOL TABLE LISTING

SYMBOL NAME	SECT	VALUE	SYMBOL NAME	SECT	VALUE
BYTCNT	A	0020	SEND	A	00A9
CLAMP	A	0003	SEND1	A	00B1
DATA	A	0022	SPE	A	0004
DATLEN	A	0021	SPICR	A	000F
DDRA	A	0004	SPIDAT	A	000E
DDRD	A	0007	SPIF	A	0007
INIT	A	00BF	SS	A	0000
MISC	A	000A	START	A	0080
PORTA	A	0000	START1	A	0083
PORTD	A	0003	TACR	A	0009
PRESCL	A	0010	TADAT	A	0008
READ	A	0091	TBCR	A	000D
READ1	A	0099	TBDAT	A	000C
SCK	A	0001	XFER	A	00D3
SDA	A	0002			

This page intentionally left blank.

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

Mfax is a trademark of Motorola, Inc.

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217. 1-303-675-2140 or 1-800-441-2447

JAPAN: Nippon Motorola Ltd.: SPD, Strategic Planning Office, 4-32-1,
Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan. 81-3-5487-8488

Customer Focus Center: 1-800-521-6274

Mfax™: RMFAX0@email.sps.mot.com – TOUCHSTONE 1-602-244-6609
Motorola Fax Back System – US & Canada ONLY 1-800-774-1848
– <http://sps.motorola.com/mfax/>

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

HOME PAGE: <http://motorola.com/sps/>



MOTOROLA

31850-1 PRINTED IN USA 9/97 IMPERIAL LITHO 100

ANE405/D

