16 Bit Parallel I/O Expansion Card for Acorn RISC OS-Based Computer Systems

User Guide

Intelligent Interfaces Ltd
October 2003

All information is given by Intelligent Interfaces in good faith. However, it is acknowledged that there may be errors or omissions in this User Guide. Intelligent Interfaces welcomes comments and suggestions relating to the 16 Bit Parallel I/O Expansion Card and this User Guide.

All correspondence should be addressed to:-

Technical Enquiries
Intelligent Interfaces Ltd
P O Box 80
Eastleigh
Hampshire
SO53 2YX
Tel: 023 8026 1514
Fax: 087 0052 1281
E-mail: support@intint.demon.co.uk

This User Guide is intended only to assist the reader in the use of the 16 Bit Parallel I/O Expansion Card and, therefore, Intelligent Interfaces shall not be liable for any loss or damage whatsoever arising from the use of any information or particulars in, or any error or omission in, this User Guide, or any incorrect use of the Expansion Card.

Contents

Appendices

1 Introduction

Before using the 16 Bit Parallel I/O Expansion Card check that the following items in addition to this User Guide have been received

16 Bit Parallel I/O Expansion Card
Software Distribution Disc

If any item is missing contact the supplier.

1.1 Fitting the Expansion Card

Before fitting the Expansion Card refer to Appendix I and set the DIP switches. The default setting SW1-SW10 all off make PL2, the upper connector on the rear panel, a 16 bit input port and PL3, the lower connector on the rear panel, a 16 bit output port. Refer to Appendix II and set the Option Selection Links.

The card can be fitted in any Acorn Computer with an expansion backplane. The card can be fitted in the external expansion card socket of an A3000 computer.

To fit the card in an A300 series, A400 series, A540, A5000, Risc PC or A7000 computer:-

1 Switch off the power to the Computer.
2 Disconnect the Computer from the mains supply.
3 The card can be fitted in any unused expansion card slot.
4 Remove the blanking plate from the rear of the Computer and retain the two screws.
5 Fit the card and secure it in position using the two screws retained at stage 4 (if required, fit a joiner and blanking plate).
6 Reconnect the Computer to the mains supply.
7 Switch on the power to the Computer.

Only one 16 Bit Parallel I/O Interface Card can be fitted.

IMPORTANT NOTE
The 16 Bit Parallel I/O Expansion Card is not listed by the *PODULES command because of an error in version 1.2 of the operating system of the Archimedes Computer. This version of the operating system does not recognise simple expansion cards correctly. See also the note on Option Selection Link S9 in Appendix II.

1.2 Connecting to the 16 Bit Parallel I/O Ports of the Expansion Card

The Expansion Card has two 40-way ribbon cable headers on the rear panel. The upper connector is designated PL2 and the lower connector is designated PL3. The pin designations of both connectors are given in Appendix III.
Ribbon cables with 40-way socket connectors should be used to connect to peripherals.

1.3 The Scope of the User Guide

It is anticipated that there will be almost as many different applications for the 16 Bit Parallel I/O Expansion Card as there are cards manufactured. Therefore, this User Guide provides a description of the hardware design together with information on the 65C22 Versatile Interface Adapters (VIA's) used and details of programming in BBC BASIC, FORTRAN 77 and ARM Assembler.

2 Hardware

The 16 Bit Parallel I/O Expansion Card has been designed as a simple expansion card to be accessed through synchronous read and write cycles. HCMOS logic is used for the expansion backplane interface and series resistive termination of the data bus is used to prevent it being overdriven. A non- extended Expansion Card Identification Byte is provided and the meaning of each bit is given in Appendix IV. Option Selection Links S1 and S2 enable the Simple Expansion Card I.D. to be selected from 12 (&C) to 15 (&F).

The Expansion Card uses two 65C22 Versatile Interface Adapters (VIA's). These are clocked at 2 MHz. The 65C22 is described in detail in section 9.

One of the 65C22's is connected to the least significant (LS) 8 data bits BD[0:7] of the expansion backplane data bus and the other is connected to the most significant (MS) 8 data bits BD[8:15] of the expansion backplane data bus.

The address decoding has been arranged so that either the 65C22 connected to the LS data byte or the 65C22 connected to the MS data byte is addressed for 8 bit operation or both 65C22's are addressed together for 16 bit operation. This is shown in Appendix V.

All the peripheral data lines are buffered by SN74LS245 octal bus transceivers and all the peripheral control lines by SN74LS125 and SN74LS126 quadruple bus buffer gates. All the buffer IC's are socketed to enable easy replacement. The direction of the data and control lines, ie whether they are inputs or outputs, is determined by DIP switches. The DIP Switch settings are given in Appendix I.

Nine Option Selection Links are provided and the options available are listed in Appendix II.

S1, S2 and S9 are grouped together on the Expansion Card and are associated with the Simple Expansion Card I.D.

S3, S4 and S5 are grouped together and are associated with PL2. S6, S7 and S8 are grouped together and are associated with PL3. The Expansion Card has two 40-way ribbon cable headers on the rear panel. The upper connector is designated PL2 and the lower connector is designated PL3. The pin designations of both connectors are given is Appendix III.

The 65C22 Port A lines (CA1, CA2) handshake data on both a read and a write of the port. The 65C22 Port B lines (CB1, CB2) handshake data only on a write of the port. Therefore, the default setting of the DIP switches (all off) has been arranged to make PL2, to which the Port A lines are connected, a 16 bit input port and PL3, to which the Port B lines are connected, a 16 bit output port.

If Option Selection Links S7 and S8 are both in position A (PL3 pins 1 and 3 to +5V and PL3 pins 21 and 23 to +5V) then both the lower set of 20 pins (1 to 20) and the upper set of 20 pins (21 to 40) have the same pin designations as the User Port of the BBC Microcomputer. If Option Selection Links S4 and S5 are both in position A (PL2 pins 1 and 3 to +5V and PL2 pins 21 and 23 to +5V) then both the lower set of 20 pins (1 to 20) and the upper set of 20 pins (21 to 40) have the same pin designations as the User Port of the BBC Microcomputer except that the data lines are from the A Port of the 65C22 rather than from the B Port. Note that as the 65C22's of the Expansion Card are clocked at 2 MHz rather than the 1 MHz of the 6522 of the BBC Microcomputer the internal timers run at twice the speed.

3 The 16 Bit Parallel I/O Software

A relocatable module which extends the operating system to enable reading and writing of the 65C22 registers through the II16BitPIO SWI' s (Software Interrupts) allocated by Acorn Computers is provided on the distribution disc. These SWI' s are listed in Appendix VI.

The module is loaded by typing

```
 *RMLOAD $.Modules.16BitPIO
```

To check that the modules has been loaded type

```
*MODULES
```

This lists all the modules present. Once initialised the software remains active until the module is either deleted or re-initialised. To delete the module type

```
*RMKILL 16BitPIO
```

The module is also deleted by a reset of the computer.

II16BitPIO SWI' s are passed by the operating system to the SWI handler code of the above module via the offset in the module header as described in Volume 2 of the Archimedes Programmers Reference Manual. Unfortunately, passing through the operating system takes a long time which results in slow reading and writing of the 65C22 registers.

Therefore, an alternative relocatable module is provided on the distribution disc which ' wedges' in between the SWI Exception Vector at address hexadecimal 00000008 and the operating system SWI entry. This module intercepts the II16BitPIO SWI' s and executes them. No time is taken to pass through the operating system and this results in fast reading and writing of the 65C22 registers. The module is loaded by typing

```
*RMLOAD $.Modules.16BitPIOft
```

As it is to some extent outside the operating system it should be used with care and only when fast operation is required.

4 Using the 16 Bit Parallel I/O Expansion Card from BASIC

To simplify the use of the Expansion Card from BASIC, a library of BASIC procedures and functions is supplied on the software distribution disc.

The procedures and functions can be INSTALLed in a BASIC Library by typing

```
>INSTALL "$.BASICProcs.16BitPIO"
or XXXX $.BASIC.Library.16BitPIO XXXXX
```

Example Program Using the BASIC Library

The following test program illustrates the use of the Expansion Card from BASIC. Note that a special cable connecting PL2 and PL3 is required to use it.

```
 10 REM 11th August 1988
 20 REM automatic test for 16 bit Parallel I/O Expansion Card
 30 REM using BASIC Procedures and Functions Library
 40
 50 CLS
 60 PRINT"DIP Switches all OFF?"
 70 PRINT
 80 PRINT"S3 - S8 position B?"
 90 PRINT
100 INPUT"Y/N "a$:     a$ = CHR$(ASC(a$)AND&DF)
110 IF a$<>"Y" THEN PRINT"Switch off computer and set up
    correctly": END
120
130 PRINT
140 INPUT"Expansion Card in slot number "slot%
150 PRINT
160 REM test for 16 bits
170
180 fail=FALSE
190 PROCwr16(slot%,2,&FFFF) :    REM DDRB outputs
200 PROCwr16(slot%,3,&0000) :    REM DDRA inputs
210 PROCwr16(slot%,12,&8888) :   REM PCR CA2 and CB2 handshake output
     modes
220 REM CA1 and CB1 control negative transitions
230 PROCwr16(slot%,11,&0101) :    REM ACR PA latch enable
240
250 FOR I% = 0 TO &FFFF STEP &0101
260  PROCwr16(slot%,0,I%) :     REM write ORB
270
280  REPEAT
300  UNTIL (FNrd16(slot%,13) AND &0202) = &0202 :
     REM CA1 interrupt flag set in IFR
320
330  IF FNrd16(slot%,1) <> I% THEN PRINTTAB(0,8);~I%,"16
     Fail":      fail=TRUE ELSE PRINTTAB(0,8);~I%,"16 Pass"
340
350  REPEAT
370  UNTIL (FNrd16(slot%,13) AND &1010) = &1010 :
     REM CB1 interrupt flag set in IFR
380
390 NEXT I%
400
410 REM test for ls 8 bits
```

```
420
430 faills8=FALSE
440 FOR I% = 0 TO &FF STEP &01
450  PROCwrls(slot%,0,I%) :     REM write ORB
460
470  REPEAT
490  UNTIL (FNrdls(slot%,13) AND &02) = &02 :
     REM CA1 interrupt flag set in IFR
500
520  IF FNrdls(slot%,1) <> I% THEN PRINTTAB(0,10);~I%,"ls8     Fail":
 faills8=TRUE ELSE PRINTTAB(0,10);~I%,"ls8 Pass"
530
540  REPEAT
560  UNTIL (FNrdls(slot%,13) AND &10) = &10 :
     REM CB1 interrupt flag set in IFR
570
580 NEXT I%
590
600 REM test for ms 8 bits
610
620 failms8=FALSE
630 FOR I% = 0 TO &FF STEP &01
640  PROCwrms(slot%,0,I%) :     REM write ORB
650
660  REPEAT
680  UNTIL (FNrdms(slot%,13) AND &02) = &02 :
     REM CA1 interrupt flag set in IFR
690
710  IF FNrdms(slot%,1) <> I% THEN PRINTTAB(0,12);~I%,"ms8
     Fail":     failms8=TRUE ELSE PRINTTAB(0,12);~I%,"ms8 Pass"
720
730  REPEAT
750  UNTIL (FNrdms(slot%,13) AND &10) = &10 :
    REM CB1 interrupt flag set in IFR
760
770 NEXT I%
780
790 PRINT
800 IF fail THEN PRINT"16 bit failed" ELSE PRINT"16 bit passed"
810 IF faills8 THEN PRINT"ls8 bit failed" ELSE PRINT"ls8 bit
    passed"
820 IF failms8 THEN PRINT"ms8 bit failed" ELSE PRINT"ms8 bit
    passed"
830 END
```

Example Program Using the SYS Statement

The II16BitPIO SWI' s can be called directly from BASIC using the SYS statement. The following test program illustrates the use of the Expansion Card from BASIC and runs faster than the previous program which uses the BASIC Library. Note that a special cable connecting PL2 and PL3 is required to use it.

```
 10 REM 13th July 1988
 20 REM automatic test for 16 bit Parallel I/O Expansion Card
 30 REM 2nd August 1988 adds variable slot position
 40
 50 CLS
 60 PRINT"DIP Switches all OFF?"
 70 PRINT
 80 PRINT"S3 - S8 position B?"
 90 PRINT
```

```
100 INPUT"Y/N "a$:     a$ = CHR$(ASC(a$)AND&DF)
110 IF a$<>"Y" THEN PRINT"Switch off computer and set up
    correctly":END
120
130 PRINT
140 INPUT"Expansion Card in slot number "slot%
150 PRINT
160 REM test for 16 bits
170
180 fail=FALSE
190 SYS &48605,slot%,2,&FFFF :    REM DDRB outputs
200 SYS &48605,slot%,3,&0000 :    REM DDRA inputs
210 SYS &48605,slot%,12,&8888 :    REM PCR CA2 and CB2 handshake
    output modes
220 :    REM CA1 and CB1 control negative transitions
230 SYS &48605,slot%,11,&0101 :    REM ACR PA latch enable
240
250 FOR I% = 0 TO &FFFF STEP &0101
260  SYS &48605,slot%,0,I% :     REM write ORB
270
280  REPEAT
290   SYS &48604,slot%,13 TO ,,data%
300  UNTIL (data% AND &0202) = &0202 : REM CA1 interrupt flag set in
     IFR
310
320  SYS &48604,slot%,1 TO ,,data% :     REM read ORA
330  IF data% <> I% THEN PRINTTAB(0,8);~I%,"16 Fail": fail=TRUE ELSE
     PRINTTAB(0,8);~I%,"16 Pass"
340
350  REPEAT
360   SYS &48604,slot%,13 TO ,,data%
370  UNTIL (data% AND &1010) = &1010 : REM CB1 interrupt flag set in
     IFR
380
390 NEXT I%
400
410 REM test for ls 8 bits
420
430 faills8=FALSE
440 FOR I% = 0 TO &FF STEP &01
450  SYS &48601,slot%,0,I% :     REM write ORB
460
470  REPEAT
480   SYS &48600,slot%,13 TO ,,data%
490  UNTIL (data% AND &02) = &02 : REM CA1 interrupt flag set in IFR
500
510  SYS &48600,slot%,1 TO ,,data% : REM read ORA
520  IF data% <> I% THEN PRINTTAB(0,10);~I%,"ls8 Fail": faills8=TRUE
     ELSE PRINTTAB(0,10);~I%,"ls8 Pass"
530
540  REPEAT
550   SYS &48600,slot%,13 TO ,,data%
560  UNTIL (data% AND &10) = &10 : REM CB1 interrupt flag set in IFR
570
580 NEXT I%
590
600 REM test for ms 8 bits
610
620 failms8=FALSE
630 FOR I% = 0 TO &FF STEP &01
640  SYS &48603,slot%,0,I% :     REM write ORB
```

```
650
660  REPEAT
670   SYS &48602,slot%,13 TO ,,data%
680  UNTIL (data% AND &02) = &02 : REM CA1 interrupt flag set in IFR
690
700  SYS &48602,slot%,1 TO ,,data% : REM read ORA
710  IF data% <> I% THEN PRINTTAB(0,12);~I%,"ms8 Fail": failms8=TRUE
     ELSE PRINTTAB(0,12);~I%,"ms8 Pass"
720
730  REPEAT
740   SYS &48602,slot%,13 TO ,,data%
750  UNTIL (data% AND &10) = &10 : REM CB1 interrupt flag set in IFR
760
770 NEXT I%
780
790 PRINT
800 IF fail THEN PRINT"16 bit failed"
```

5 BASIC Procedures and Functions

The following pages describe in alphabetical order the BASIC procedures and functions under a number of headings.

Purpose
A description of what the procedure or function does.

Syntax
A formal declaration of the procedure or function.

Parameters
A list of the parameters required, giving their type, range and, if necessary, additional information on their specification.

Results
A list of the results returned, giving their type, range and, if necessary, additional information.

Example
Example to illustrate the use of the procedure or function. For clarity the parameters of procedures are shown as constants. However, variables of the same type can be used.

Notes
Further details of the operation of the software.

rdls

Purpose
To read a register of the 65C22 connected to the Least Significant (LS) data lines of the expansion backplane.

Syntax
<data byte>=FNrdls(<slot>,<register>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15

Result
data byte (integer) - &00 to &FF

Example

```
data%=FNrdls(1,13)
```

reads the interrupt flag register (IFR), register number 13, of the 65C22 connected to the least significant data lines in slot number 1 of the expansion backplane.

Note
The 65C22 connected to the least significant data lines of the expansion backplane controls pins 2 to 20 of PL2 (Port A) and pins 2 to 20 of PL3 (Port B).

wrls

Purpose
To write to a register of the 65C22 connected to the Least Significant (LS) data lines of the expansion backplane.

Syntax
PROCwrls(<slot>,<register>,<data byte>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15 data byte (integer) - &00 to &FF

Result
None

Example
```
PROCwrls(3,0,&AA)
```

writes the data byte (hexadecimal AA) to output register B (ORB), register number 0, of the 65C22 connected to the least significant data lines in slot number 3 of the expansion backplane.

Note
The 65C22 connected to the least significant data lines of the expansion backplane controls pins 2 to 20 of PL2 (Port A) and pins 2 to 20 of PL3 (Port B).

rdms

Purpose
To read a register of the 65C22 connected to the Most Significant (MS) data lines of the expansion backplane.

Syntax
<data byte>=FNrdms(<slot>,<register>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15

Result
data byte (integer) - &00 to &FF

Example
```
data%=FNrdms(1,13)
```

reads the interrupt flag register (IFR), register number 13, of the 65C22 connected to the most significant data lines in slot number 1 of the expansion backplane.

Note
The 65C22 connected to the most significant data lines of the expansion backplane controls pins 22 to 40 of PL2 (Port A) and pins 22 to 40 of PL3 (Port B).

wrms

Purpose
To write to a register of the 65C22 connected to the Most Significant (MS) data lines of the expansion backplane.

Syntax
PROCwrms(<slot>,<register>,<data byte>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15 data byte (integer) - &00 to &FF

Result
None

Example
```
PROCwrms(3,0,&AA)
```

writes the data byte (hexadecimal AA) to output register B (ORB), register number 0, of the 65C22 connected to the most significant data lines in slot number 3 of the expansion backplane.

Note
The 65C22 connected to the least significant data lines of the expansion backplane controls pins 22 to 40 of PL2 (Port A) and pins 22 to 40 of PL3 (Port B).

rd16

Purpose
To read the corresponding registers of the 65C22' s connected to both the Least Significant (LS) and Most Significant (MS) data lines of the expansion backplane.

Syntax
<data halfword>=FNrd16(<slot>,<register>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15

Result
data halfword (integer) - &0000 to &FFFF (16 bits)

Example
```
data%=FNrd16(0,1)
```

reads output register A (ORA), register number 1, of both 65C22' s connected to the least and most significant data lines in slot number 0 of the expansion backplane. Register A controls the pins of PL2.

wr16

Purpose
To write to the corresponding registers of the 65C22' s connected to both the Least Significant (LS) and Most Significant (MS) data lines of the expansion backplane.

Syntax
PROCwr16(<slot>,<register>,<data halfword>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15 data halfword (integer) - &0000 to &FFFF (16 bits)

Result
None

Example
```
PROCwr16(0,0,&55AA)
```

writes to output register B (ORB), register number 0, of both 65C22' s connected to the least significant (hexadecimal AA) and most significant (hexadecimal 55) data lines in slot number 0 of the expansion backplane. Output register B controls the pins of PL3.

peek16

Purpose
To read the corresponding registers of the 65C22' s connected to both the Least Significant (LS) and Most Significant (MS) data lines of the expansion backplane.

Syntax
<data halfword>=FNpeek16(<address>)

Parameters
address (integer) - synchronous access address of 65C22 register

Result

data halfword (integer) - &0000 to &FFFF (16 bits)

Example
```
data%=FNpeek16(&33C7800)
```

reads register B (RB) of both 65C22' s connected to the least and most significant data lines in slot number 1, synchronous access address hexadecimal 33C7800. Register B controls the pins of PL3.

poke16

Purpose
To write to the corresponding registers of the 65C22' s connected to both the Least Significant (LS) and Most Significant (MS) data lines of the expansion backplane.

Syntax
PROCpoke16(<address>,<data halfword>)

Parameters
address - synchronous access address of 65C22 register data halfword (integer) - &0000 to &FFFF (16 bits)

Result
None

Example
```
PROCpoke16(&33CB804,&55AA)
```

writes to register A (RA) of both 65C22' s connected to the least significant (hexadecimal AA) and most significant (hexadecimal 55) data lines in slot number 2, synchronous access address hexadecimal 33CB804. Output Register A controls the pins of PL2.

rd32

Purpose
To read a word from any address in memory.

Syntax
<data word>=FNrd32(<address>)

Parameters
address (integer) - address in memory

Result
data word (integer) - &00000000 to &FFFFFFFF (32 bits)

Example
```
data%=FNrd32(&120000)
```

reads the word at address hexadecimal 120000.

Note
Use with extreme care.

wr32

Purpose
To write a word to any address in memory.

Syntax
PROCwr32(<address>,<data word>)

Parameters
address (integer) - address in memory data word (integer) - &00000000 to &FFFFFFFF (32 bits)

Result
None

Example
```
PROCwr32(&150000,&55AA55AA)
```

writes the word (hexadecimal 55AA55AA) to address hexadecimal 150000.

Note
Use with extreme care.

6 Using the 16 Bit Parallel I/O Expansion Card from FORTRAN 77

To enable the use of the Expansion Card from FORTRAN 77 an interface library is supplied on the software distribution disc. The 16BitPIO Module must be loaded in order to use the Library (see Chapter 3).

Compiling and Linking

Compile the main FORTRAN program by typing

```
*f77 myprog
```

Then link with the Interface Library by typing

```
*link aof.myprog aof.IF16BitPIO $.Library.lib.f77 -image myprog
```

Example Program

The following test program illustrates the use of the Expansion Card from FORTRAN 77. Note that a special cable connecting PL2 and PL3 is required to use it.

```
      PROGRAM PIOTEST1
C     3rd August 1988
C     Archimedes Fortran 77 - automatic test for 16 bit Parallel I/O
C     Expansion Card - variable slot position
C
      INTEGER slot,data
      CHARACTER a
      LOGICAL fail,faills8,failms8
C
      PRINT '($,1A)',CHAR(12)
C
      PRINT *, 'DIP Switches all OFF?'
      PRINT *
      PRINT *, 'S3 - S8 position B?'
      PRINT *
      PRINT '($,5A)', ' Y/N '
      READ *, a
      IF (a.EQ.'Y') GOTO 10
      PRINT *, 'Switch off computer and set up correctly'
      STOP
C
   10 PRINT *
      PRINT '($,31A)', ' Expansion Card in slot number '
      READ *,slot
C
C     test for 16 bits
C
      fail=.FALSE.
C               DDRB outputs
      CALL IFWR16(slot,2,?IFFFF)
C               DDRA inputs
      CALL IFWR16(slot,3,?I0000)
C             PCR CA2 and CB2 handshake output modes
C             CA1 and CB1 control negative transitions
      CALL IFWR16(slot,12,?I8888)
C               ACR PA latch enable
      CALL IFWR16(slot,11,?I0101)
```

```
C
      DO 20,I=0,?IFFFF,?I0101
C               write ORB
      CALL IFWR16(slot,0,I)
   30 data=IFRD16(slot,13)
C               CA1 interrupt flag set in IFR
      IF (IAND(data,?I0202).NE.?I0202) GOTO 30
C               read ORA
      data=IFRD16(slot,1)
        IF (data.NE.I) THEN
        PRINT '($,3A)',CHAR(31),CHAR(0),CHAR(8)
        PRINT *, I,' 16 Fail'
        fail=.TRUE.
        ELSE
        PRINT '($,3A)',CHAR(31),CHAR(0),CHAR(8)
        PRINT *, I,' 16 Pass'
        END IF
   40 data=IFRD16(slot,13)
C              CB1 interrupt flag set in IFR
      IF (IAND(data,?I1010).NE.?I1010) GOTO 40
   20 CONTINUE
C
C   test for ls 8 bits
C
      faills8=.FALSE.
C
      DO 50,I=0,?IFF,?I01
C               write ORB
      CALL IFWRLS(slot,0,I)
    0 data=IFRDLS(slot,13)
C               CA1 interrupt flag set in IFR
      IF (IAND(data,?I02).NE.?I02) GOTO 60
C               read ORA
      data=IFRDLS(slot,1)
        IF (data.NE.I) THEN
        PRINT '($,3A)',CHAR(31),CHAR(0),CHAR(10)
        PRINT *, I,' ls8 Fail'
        faills8=.TRUE.
        ELSE
        PRINT '($,3A)',CHAR(31),CHAR(0),CHAR(10)
        PRINT *, I,' ls8 Pass'
        END IF
   70 data=IFRDLS(slot,13)
C              CB1 interrupt flag set in IFR
      IF (IAND(data,?I10).NE.?I10) GOTO 70
   50 CONTINUE
C
C   test for ms 8 bits
C
      failms8=.FALSE.
C
      DO 80,I=0,?IFF,?I01
C               write ORB
      CALL IFWRMS(slot,0,I)
   90 data=IFRDMS(slot,13)
C               CA1 interrupt flag set in IFR
      IF (IAND(data,?I02).NE.?I02) GOTO 90
C               read ORA
      data=IFRDMS(slot,1)
        IF (data.NE.I) THEN
        PRINT '($,3A)',CHAR(31),CHAR(0),CHAR(12)
```

```
        PRINT *, I,' ms8 Fail'
        failms8=.TRUE.
        ELSE
        PRINT '($,3A)',CHAR(31),CHAR(0),CHAR(12)
        PRINT *, I,' ms8 Pass'
        END IF
  100 data=IFRDMS(slot,13)
C                CB1 interrupt flag set in IFR
      IF (IAND(data,?I10).NE.?I10) GOTO 100
   80 CONTINUE
C
      PRINT *
      IF (fail) THEN
        PRINT *,'16 bit Failed'
        ELSE
        PRINT *,'16 bit Passed'
      END IF
C
      IF (faills8) THEN
        PRINT *,'ls8 bit Failed'
        ELSE
        PRINT *,'ls8 bit Passed'
      END IF
C
      IF (failms8) THEN
        PRINT *,'ms8 bit Failed'
        ELSE
        PRINT *,'ms8 bit Passed'
       END IF
      STOP
      END
```

7 Interface Library for FORTRAN 77

The following pages describe in alphabetical order the FORTRAN 77 subroutines and functions under a number of headings.

Purpose
A description of what the subroutine or function does.

Syntax
A formal declaration of the subroutine or function.

Parameters
A list of the parameters required, giving their type, range and, if necessary, additional information on their specification.

Results
A list of the results returned, giving their type, range and, if necessary, additional information.

Example
Example to illustrate the use of the subroutine or function. For clarity the parameters of subroutines are shown as con stants. However, variables of the same type can be used.

Notes
Further details of the operation of the software.

IFRDLS

Purpose
To read a register of the 65C22 connected to the Least Significant (LS) data lines of the expansion backplane.

Syntax
<data byte>=IFRDLS(<slot>,<register>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15

Result
data byte (integer) - &00 to &FF

Example
```
data=IFRDLS(1,13)
```

reads the interrupt flag register (IFR), register number 13, of the 65C22 connected to the least significant data lines in slot number 1 of the expansion backplane.
Note
The 65C22 connected to the least significant data lines of the expansion backplane controls pins 2 to 20 of PL2 (Port A) and pins 2 to 20 of PL3 (Port B).

IFWRLS

Purpose
To write to a register of the 65C22 connected to the Least Significant (LS) data lines of the expansion backplane.

Syntax
IFWRLS(<slot>,<register>,<data byte>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15 data byte (integer) - &00 to &FF

Result
None

Example
```
CALL IFWRLS(3,0,?IAA)
```

writes the data byte (hexadecimal AA) to output register B (ORB), register number 0, of the 65C22 connected to the least significant data lines in slot number 3 of the expansion backplane.

Note
The 65C22 connected to the least significant data lines of the expansion backplane controls pins 2 to 20 of PL2 (Port A) and pins 2 to 20 of PL3 (Port B).

IFRDMS

Purpose
To read a register of the 65C22 connected to the Most Significant (MS) data lines of the expansion backplane.

Syntax
<data byte>=IFRDMS(<slot>,<register>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15

Result
data byte (integer) - &00 to &FF

Example
```
data=IFRDMS(1,13)
```

reads the interrupt flag register (IFR), register number 13, of the 65C22 connected to the most significant data lines in slot number 1 of the expansion backplane.

Note
The 65C22 connected to the most significant data lines of the expansion backplane controls pins 22 to 40 of PL2 (Port A) and pins 22 to 40 of PL3 (Port B).

IFWRMS

Purpose
To write to a register of the 65C22 connected to the Most Significant (MS) data lines of the expansion backplane.

Syntax
IFWRMS(<slot>,<register>,<data byte>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15 data byte (integer) - &00 to &FF

Result
None

Example
```
CALL IFWRMS(3,0,?IAA)
```

writes the data byte (hexadecimal AA) to output register B (ORB), register number 0, of the 65C22 connected to the most significant data lines in slot number 3 of the expansion backplane.

Note
The 65C22 connected to the least significant data lines of the expansion backplane controls pins 22 to 40 of PL2 (Port A) and pins 22 to 40 of PL3 (Port B).

IFRD16

Purpose
To read the corresponding registers of the 65C22' s connected to both the Least Significant (LS) and Most Significant (MS) data lines of the expansion backplane.

Syntax
<data halfword>=IFRD16(<slot>,<register>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15

Result
data halfword (integer) - &0000 to &FFFF (16 bits)

Example
```
data=IFRD16(0,1)
```

reads output register A (ORA), register number 1, of both 65C22' s connected to the least and most significant data lines in slot number 0 of the expansion backplane. Register A controls the pins of PL2.

IFWR16

Purpose
To write to the corresponding registers of the 65C22' s connected to both the Least Significant (LS) and Most Significant (MS) data lines of the expansion backplane.

Syntax
IFWR16(<slot>,<register>,<data halfword>)

Parameters
slot (integer) - expansion card slot number 0, 1, 2 or 3 register (integer) - 65C22 register number 0 to 15 data halfword (integer) - &0000 to &FFFF (16 bits)

Result
None

Example
```
CALL IFWR16(0,0,?I55AA)
```

writes to output register B (ORB), register number 0, of both 65C22' s connected to the least significant (hexadecimal AA) and most significant (hexadecimal 55) data lines in slot number 0 of the expansion backplane. Output register B controls the pins of PL3.

IFPEEK16

Purpose
To read the corresponding registers of the 65C22' s connected to both the Least Significant (LS) and Most Significant (MS) data lines of the expansion backplane.

Syntax
<data halfword>=IFPEEK16(<address>)

Parameters
address (integer) - synchronous access address of 65C22 register

Result
data halfword (integer) - &0000 to &FFFF (16 bits)

Example
```
data=IFPEEK16(?I033C7800)
```

reads register B (RB) of both 65C22' s connected to the least and most significant data lines in slot number 1, synchronous access address hexadecimal 033C7800. Register B controls the pins of PL3.

IFPOKE16

Purpose
To write to the corresponding registers of the 65C22' s connected to both the Least Significant (LS) and Most Significant (MS) data lines of the expansion backplane.

Syntax
IFPOKE16(<address>,<data halfword>)

Parameters
address - synchronous access address of 65C22 register data halfword (integer) - &0000 to &FFFF (16 bits)

Result
None

Example
```
CALL IFPOKE16(?I033CB804,?I55AA)
```

writes to register A (RA) of both 65C22' s connected to the least significant (hexadecimal AA) and most significant (hexadecimal 55) data lines in slot number 2, synchronous access address hexadecimal 033CB804. Output Register A controls the pins of PL2.

IFRD32

Purpose
To read a word from any address in memory.

Syntax
<data word>=IFRD32(<address>)

Parameters
address (integer) - address in memory

Result
data word (integer) - &00000000 to &FFFFFFFF (32 bits)

Example
```
data=IFRD32(?I120000)
```

reads the word at address hexadecimal 120000.

Note
Use with extreme care.

IFWR32

Purpose
To write a word to any address in memory.

Syntax
IFWR32(<address>,<data word>)

Parameters
address (integer) - address in memory data word (integer) - &00000000 to &FFFFFFFF (32 bits)

Result
None

Example
```
CALL IFWR32(?I150000,?I55AA55AA)
```

writes the word (hexadecimal 55AA55AA) to address hexadecimal 150000.

Note
Use with extreme care.

8 Using the 16 Bit Parallel I/O Expansion Card from Assembler

Read the Archimedes Programmer' s Reference Manual, particularly the sections on Fundamental Operating System Concepts and Appendix A on ARM Assembler, before writing programs in assembler. General information on communicating with the operating system is given in the Programmer' s Reference Manual.

The 16BitPIO software is a relocatable module which extends the operating system. The Module must be loaded from disc (see Chapter 3).

The facilities of the Expansion Card are accessed through the II16BitPIO SWI' s (Software Interrupts) allocated by Acorn Computers. These SWI' s are listed in Appendix VI.

None of the II16BitPIO SWI' s generate or return errors. Therefore, error generating SWI' s always return with the V flag clear.

Parameters are passed and results returned in the processor' s registers. Appendix VI gives the allocation of parameters and results to the processor' s registers.

All SWI' s share the same structure.

On entry
R0 Expansion Card slot number
R1 65C22 register number
R2 data (for write SWI' s)

On exit
R0 preserved
R1 preserved
R2 data (for read SWI' s)

To enable one of the 65C22' s to interrupt the ARM (Acorn RISC Machine) processor both the 65C22' s interrupt must be enabled by setting the appropriate bit in its IER (Interrupt Enable Register) and the Expansion Card Interrupt Request, bit 5 of IRQ MASK B of the IOC (Input Output Controller), must also be set. IRQ MASK B of the IOC is at address hexadecimal 03200028.

Example Program Using the II16BitPIO SWI's

The following test program illustrates the use of the Expansion Card from Assembler. Note that a special cable connecting PL2 and PL3 is required to use it.

```
   10 REM 11th August 1988
   20 REM automatic test for 16 bit Parallel I/O Expansion Card
   30 REM written in BASIC Assembler
   40 REM accesses the Expansion Card through SWI's
   50
   60 CLS
   70 PRINT"DIP Switches all OFF?"
   80 PRINT
   90 PRINT"S3 - S8 position B?"
  100 PRINT
  110 INPUT"Y/N "a$:     a$ = CHR$(ASC(a$)AND&DF)
  120 IF a$<>"Y" THEN PRINT"Switch off computer and set up
correctly":     END
  130
  140 PRINT
  150 INPUT"Expansion Card in slot number "slot%
```

```
160 PRINT
170 REM test for 16 bits
180
190 REM REGISTERS
200
210 R0=0
220 R1=1
230 R2=2
240 R3=3
250 R4=4
260 R5=5
270 R6=6
280 R7=7
290 R14=14
300 R15=15
310
320 DIM code% 1023
330
340 FOR i%=0TO2 STEP 2
350  P%=code%
360  [OPTi%
370  LDR R0,slot ; R0 slot
380  MOV R1,#2 ; R1 register
390  LDR R2,DDRBoutputs ; R2 data
400  SWI "II16BitPIO_wr16" ; write to DDRB
410  MOV R1,#3
420  LDR R2,DDRAinputs
430  SWI "II16BitPIO_wr16"
440  MOV R1,#12
450  LDR R2,PCR
460  SWI "II16BitPIO_wr16"
470  MOV R1,#11
480  LDR R2,ACR
490  SWI "II16BitPIO_wr16"
500  LDR R3,start ; set up loop parameters
510  LDR R4,increment
520  LDR R5,finish
530  LDR R7,true
540  .test3
550  MOV R1,#0
560  MOV R2,R3
570  SWI "II16BitPIO_wr16" ; write to ORB
580  MOV R1,#13
590  LDR R6,CA1mask
600  .test1
610  SWI "II16BitPIO_rd16"
620  AND R2,R2,R6 ; CA1 interrupt flag set in IFR?
630  CMP R2,R6
640  BNE test1 ; no
650  MOV R1,#1
660  SWI "II16BitPIO_rd16" ; read from ORA
670  CMP R2,R3 ; read data = write data?
680  STRNE R7,fail ; no
690  MOV R1,#13
700  LDR R6,CB1mask
710  .test2
720  SWI "II16BitPIO_rd16"
730  AND R2,R2,R6 ; CB1 interrupt flag set in IFR?
740  CMP R2,R6
750  BNE test2 ; no
760  ADD R3,R3,R4 ; increment loop counter
```

```
 770  CMP R3,R5 ; finished
 780  BLS test3 ; no
 790  MOVS R15,R14
 800  ;
 810  ; VARIABLES
 820  ;
 825  ALIGN
 830  .slot EQUD &0
 840  .start EQUD &0
 850  .increment EQUD 0
 860  .finish EQUD &0
 870  .fail EQUD &0
 880  ;
 890  ; CONSTANTS (Literals)
 900  ;
 905  ALIGN
 910  .DDRBoutputs EQUD &0000FFFF
 920  .DDRAinputs EQUD &00000000
 930  .PCR EQUD &00008888
 940  .ACR EQUD &00000101
 950  .CA1mask EQUD &00000202
 960  .CB1mask EQUD &00001010
 970  .true EQUD &FFFFFFFF
 980  ;
 990  ]
1000 NEXT i%
1010
1020 REM set up variables
1030 slot!0=slot%
1040 start!0=0
1050 increment!0=&01
1060 finish!0=&FFFF
1070 fail%=FALSE
1080 fail!0=fail%
1090 CALL code%
1100 fail%=fail!0
1110 IF fail% THEN PRINT"16 Bit failed" ELSE PRINT"16 Bit passed"
1120 END
```

Example Program Directly Addressing the Hardware

The alternative relocatable module 16BitPIOft, described in Chapter 3 which ' wedges' in between the SWI Exception Vector and the operating system SWI entry results in fast reading and writing of the 65C22 registers. However, the fastest input and output can be achieved by directly addressing the hardware. Normally this is only worthwhile when inputting or outputting blocks of data which require multiple reads and writes of the 65C22 registers.

The following program illustrates the use of the Expansion Card from Assembler. Note that a special cable connecting PL2 and PL3 is required to use it.

```
 10 REM 11th August 1988
 20 REM automatic test for 16 bit Parallel I/O Expansion Card
 30 REM written in BASIC Assembler
 40 REM directly accesses the Expansion Card
 50
 60 CLS
 70 PRINT"DIP Switches all OFF?"
 80 PRINT
 90 PRINT"S3 - S8 position B?"
100 PRINT
110 INPUT"Y/N "a$:     a$ = CHR$(ASC(a$)AND&DF)
```

```
 120 IF a$<>"Y" THEN PRINT"Switch off computer and set up correctly":
END
 130
 140 PRINT
 150 INPUT"Expansion Card in slot number "slot%
 160 PRINT
 170 REM test for 16 bits
 180
 190 REM REGISTERS
 200
 210 R0=0
 220 R1=1
 230 R2=2
 240 R3=3
 250 R4=4
 260 R5=5
 270 R6=6
 280 R7=7
 290 R14=14
 300 R15=15
 310
 320 DIM code% 1023
 330
 340 FOR i%=0TO2 STEP 2
 350  P%=code%
 360  [OPTi%
 370  SWI "OS_EnterOS" ; enter supervisor mode
 380  LDR R0,baseaddress ; R0 points to base address for 16 bit access
 390  LDR R2,DDRBoutputs ; read constant
 400  MOV R2,R2,ASL#16 ; adjust for system to expansion backplane data
bus mapping
 410  STR R2,[R0,#8] ; write to DDRB (address of register 2 = base
address + (2x4))
 420  LDR R2,DDRAinputs
 430  MOV R2,R2,ASL#16
 440  STR R2,[R0,#12] ; write to DDRA
 450  LDR R2,PCR
 460  MOV R2,R2,ASL#16
 470  STR R2,[R0,#48] ; write to PCR
 480  LDR R2,ACR
 490  MOV R2,R2,ASL#16
 500  STR R2,[R0,#44] ; write to ACR
 510  LDR R3,start ; set up loop parameters
 520  LDR R4,increment
 530  LDR R5,finish
 540  LDR R7,true
 550  .test3
 560  MOV R2,R3,ASL#16
 570  STR R2,[R0,#0] ; write to ORB
 580  LDR R6,CA1mask
 590  .test1
 600  LDR R2,[R0,#52]
 610  AND R2,R2,R6 ; CA1 interrupt flag set in IFR?
 620  CMP R2,R6
 630  BNE test1 ; no
 640  LDR R6,rd16mask
 650  LDR R2,[R0,#4] ; read from ORA
 660  AND R2,R2,R6 ; clear top 16 bits
 670  CMP R2,R3 ; read data = write data ?
 680  STRNE R7,fail ; no
 690  LDR R6,CB1mask
```

```
 700  .test2
 710  LDR R2,[R0,#52]
 720  AND R2,R2,R6 ; CB1 interrupt flag set in IFR?
 730  CMP R2,R6
 740  BNE test2 ; no
 750  ADD R3,R3,R4 ; increment loop counter
 760  CMP R3,R5 ; finished?
 770  BLS test3 ; no
 780  MOV R0,R15 ; return to user mode
 790  TEQP R0,#3
 800  MOVNV R0,R0 ; no operation
 810  MOVS R15,R14
 820  ;
 830  ; VARIABLES
 840  ;
 845  ALIGN
 850  .baseaddress EQUD &0
 860  .start EQUD &0
 870  .increment EQUD 0
 880  .finish EQUD &0
 890  .fail EQUD &0
 900  ;
 910  ; CONSTANTS (Literals)
 920  ;
 925  ALIGN
 930  .DDRBoutputs EQUD &0000FFFF
 940  .DDRAinputs EQUD &00000000
 950  .PCR EQUD &00008888
 960  .ACR EQUD &00000101
 970  .CA1mask EQUD &00000202
 980  .CB1mask EQUD &00001010
 990  .rd16mask EQUD &0000FFFF
1000  .true EQUD &FFFFFFFF
1010  ;
1020  ]
1030 NEXT i%
1040
1050 REM set up variables
1060 baseaddress!0=&33C3800+&4000*slot%
1070 start!0=0
1080 increment!0=&01
1090 finish!0=&FFFF
1100 fail%=FALSE
1110 fail!0=fail%
1120 CALL code%
1130 fail%=fail!0
1140 IF fail% THEN PRINT"16 Bit failed" ELSE PRINT"16 Bit passed"
1150 END
```

9 The 65C22 Versatile Interface Adapter

The 65C22 Versatile Interface Adapter (VIA) provides

two 8-bit bi-directional peripheral data ports with input data latching

three bi-directional peripheral control lines

one input peripheral control line

two programmable 16-bit counter/timers

an 8-bit shift register for serial to parallel and parallel to serial conversion.

Control of peripheral devices is handled primarily through the two 8-bit bidirectional ports. Each of these lines can be programmed to act as either an input or an output, although in this application, all lines must be programmed to the same direction because of the use of the SN74LS245 octal bus transceivers. Several peripheral I/O lines can be controlled directly from the interval timers for generating programmable- frequency square waves and for counting externally generated pulses. To facilitate control of the chip, the internal registers have been organised into an interrupt flag register, an interrupt enable register and a pair of function control registers.

9.1 Registers

The 65C22 has 16 internal registers as shown in Table 9.1.

9.2 Peripheral Interface

This section contains a description of the peripheral data and control lines which are used to drive peripheral devices under control of the internal 65C22 registers. The operation of these peripheral lines is described in detail in subsequent sections.

Register Number	Register Designation	Description Write	Read
0	ORB/IRB	Output Register "B"	Input Register "B"
1	ORA/IRA	Output Register "A"	Input Register "A"
2	DDRB	Data Direction Register "B"	
3	DDRA	Data Direction Register "A"	
4	T1C-L	T1 Low-Order Latches	T1 Low-Order Counter
5	T1C-H	T1 High-Order Counter	
6	T1L-L	T1 Low-Order Latches	
7	T1L-H	T1 High-Order Latches	
8	T2C-L	T2 Low-Order Latches	T2 Low-Order Counter
9	T2C-H	T2 High-Order Counter	
10	SR	Shift Register	
11	ACR	Auxiliary Control Register	
12	PCR	Peripheral Control Register	
13	IFR	Interrupt Flag Register	
14	IER	Interrupt Enable Register	
15	ORA/IRA	Same as Reg 1 Except No "Handshake"	

Table 9.1

9.2.1 Peripheral A Port (PA0 - PA7)

The Peripheral A port consists of eight lines which can be individually programmed to act as an input or an output under control of a Data Direction Register (DDRA). The level of output pins is controlled by an Output Register (ORA) and input data can be latched into an Input Register (IRA) under control of the CA1 line.

All of these modes of operation are controlled by the system processor or through the internal control registers.

9.2.2 Peripheral A Control Lines (CA1, CA2)

The two peripheral A control lines act as interrupt inputs or as a handshake pair, one input and one output. Each line controls an internal interrupt flag with a corresponding interrupt enable bit. In addition, CA1 controls the latching of data on Peripheral A Port input lines. The various mode of operation are controlled by the system processor through the internal control registers.

9.2.3 Peripheral B Port (PB0 - PB7)

The Peripheral B port consists of eight bi-directional lines which are controlled by an Output Register (ORB) and a Data Direction Register (DDRB) in the same manner as the PA port. The polarity of the PB7 output signal can be controlled by one of the interval timers while the second timer can be programmed to count pulses on the PB6 pin.

9.2.4 Peripheral B Control Lines (CB1, CB2)

The Peripheral B control lines act as interrupt inputs or as a handshake pair, one input and one output. As with CA1 and CA2, each line controls an interrupt flag with a corresponding interrupt enable bit. In addition, these lines act as a serial port under control of the Shift Register.

9.3 Port A Registers, Port B Registers

Three registers are used in accessing each of the 8-bit peripheral ports. Each port has a Data Direction Register (DDRA, DDRB) for specifying whether the peripheral pins are to act as inputs or outputs. A "0" in a bit of the Data Direction Register causes the corresponding peripheral pin to act as an input. A "1" causes the pin to act as an output.

When the pin is programmed to act as an output, the voltage on the pin is controlled by the corresponding bit of the Output Register (ORA, ORB). A "1" in the Output Register causes the pin to go high, and a "0" causes the pin to go low. Data written into Output Register bits corresponding to pins programmed to act as inputs will be unaffected. Reading a peripheral port causes the contents of the Input Register (IRA, IRB) to be transferred onto the Data Bus. With input latching disabled, IRA will always reflect the data on the PA pins. With input latching enabled (ACR, bit 0), setting the CA1 Interrupt Flag (IFR1) by an active transition on CA1, will cause IRA to latch the contents of the Port A pins until the Interrupt Flag is cleared.

The IRB register operates in a similar manner. However, for output pins, the corresponding IRB bit will reflect the con tents of the Output Register bit instead of the actual pin. This allows proper data to be read into the processor if the output pin is not allowed to go to full high voltage eg driving transistors. If input latching is enabled on Port B, setting the CB1 Interrupt Flag will cause IRB to latch this combination of input data and ORB data until the Interrupt Flag is cleared.

9.3.1 Handshake Control

The 65C22 allows positive control of data transfers between the system processor and peripheral devices through the operation of "handshake" lines. Port A lines (CA1, CA2) handshake data on both a read and a write operation while the Port B lines (CB1, CB2) handshake on a write operation only.

9.3.2 Read Handshake

Positive control of data transfers from peripheral devices into the system processor can be accomplished effectively using "Read" handshaking. In this case, the peripheral device must generate "Data Ready" to signal the processor that valid data is present on the peripheral port. This signal normally interrupts the processor, which then reads the data, causing generation of a "Data Taken" signal. The peripheral device responds by making new data available. This process continues until the data transfer is complete.

In the 65C22, automatic "Read" handshaking is possible on the Peripheral A port only. The CA1 interrupt pin accepts the "Data Ready" signal and CA2 generates the "Data Taken" signal. The Data Ready signal will set an internal flag which may interrupt the processor or which can be polled under software control. The Data Taken signal can be either a pulse or a DC level which is set low by the system processor and is cleared by the Data Ready signal.

9.3.3 Write Handshake

The sequence of operations which allows handshaking data from the system processor to a peripheral device is very similar to that described in Section A for Read Handshaking. However, for "Write" handshaking, the processor must generate the "Data Ready" signal (through the 65C22) and the peripheral device must respond with the "Data Taken" signal. This can be accomplished on both the PA port and the PB port on the 65C22. CA2 or CB2 acts as a Data Ready output in either the DC level or pulse mode and CA1 or CB1 accepts the "Data Taken" signal from the peripheral device, setting the interrupt flag and clearing the "Data Ready" output.

9.4 Timer 1

Interval Timer T1 consists of two 8-bit latches and a 16-bit counter. The latches are used to store data which is to be loaded into the counter. After loading, the counter decrements at system clock rate (2 MHz). Upon reaching zero, an interrupt flag will be set, and I _R _Q _ will go low if enabled. The timer will then disable any further interrupts, or will automatically transfer the contents of the latches into the counter and will continue to decrement. In addition, the timer can be instructed to invert the output signal on peripheral pin PB7 each time it "times-out". Each of these modes is discussed separate ly below.

9.4.1 Writing The Timer 1 Registers

The operations which take place when writing to each of the four Timer 1 addresses are as shown in Table 9.4.1.

Note that the processor does not write directly into the low-order counter (T1C-L). Instead, this half of the counter is loaded automatically from the low-order latch when the processor writes into the high-order counter. In fact, it may not be necessary to write to the low-order counter in some applications since the timing operation is triggered by writing to the high-order counter.

Register Number	Operation
4	Write into low-order latch
	Write into high-order latch
5	Write into high-order counter Transfer low-order latch into low order counter. Reset T1 interrupt flag (IFR6)
6	Write low-order latch
7	Write high-order latch Reset T1 interrupt flag (IFR6)

Table 9.4.1

The second set of addresses allows the the processor to write into the latch register without affecting the count-down in progress. This is discussed in detail below.

9.4.2 Reading the Timer 1 Registers

For reading the Timer 1 registers, the four addresses relate directly to the four registers as shown in Table 9.4.2.

Register Number	Operation
4	Read T1 low-order counter Reset T1 interrupt flag (IFR6)
5	Read T1 high-order counter
6	Read T1 low-order latch
7	Read T1 high-order latch

Table 9.4.2

9.5 Timer 1 Operating Modes

Two bits are provided in the Auxiliary Control Register (ACR) to allow selection of the T1 operating modes. These bits and the four possible modes are as shown in Table 9.5.

ACR7 Output Enable	ACR6 "Free-Run" Enable	Mode
0	0	Generates a single time-out interrupt each time T1 is loaded. PB7 is disabled.
0	1	Generates continuous interrupts. PB7 is disabled.
1	0	Generate a single interrupt and an output pulse on PB7 for each T1 load operation
1	1	Generate continuous interrupts and a square wave output on PB7.

Table 9.5

9.5.1 Timer 1 One-Shot Mode

The interval timer one-shot mode allows generation of a single interrupt for each timer load operation. As with any interval time, the delay between the "write T1C-H" operation and generation of the processor interrupt is a direct function of the data loaded into the timing counter. In addition to generating a single interrupt, Timer 1 can be programmed to produce a single negative pulse on the PB7 peripheral pin. With the output enabled (ACR7=1) a "write T1C-H" operation will cause PB7 to go low. PB7 will return high when Timer 1 times out. The result is a single programmable width pulse.

NOTE The PB7 output enable function will over-ride bit 7 of the Data Direction Register B. PB7 will act as an output if DDRB7=1 or if ACR7=1.

In the one-shot mode, writing into the high-order latch has no effect on the operation of Timer 1. However, it will be necessary to ensure that the low-order latch contains the proper data before initiating the countdown with a "write T1C-H" operation. When the processor writes into the high-order counter, T1L-H will also copy the data, the T1 interrupt flag will be cleared, the contents of the low-order latch will be transferred into the low-order counter, and the timer will begin to decrement at system clock rate. If the PB7 output is enabled, this signal will go low on the phase two following the write operation. When the counter reaches zero, the T1 interrupt flag will be set, the I _R _Q _ pin will go low (interrupt enabled), and the signal on PB7 will go high. At this time the counter will continue to decrement at system clock rate. This allows the system processor to read the contents of the counter to determine the time since interrupt. However, the T1 interrupt flag cannot be set again unless a "write TIC-H" operation has taken place.

9.5.2 Timer 1 Free-Running Mode

The most important advantage associated with the latches in T1 is the ability to produce a continuous series of evenly spaced interrupts and the ability to produce a square wave on PB7 whose frequency is not affected by variations in the processor response time. This is accomplished in the "free-running" mode.

In the free-running mode (ACR6=1), the interrupt flag is set and the signal on PB7 is inverted each time the counter reaches zero. However, instead of continuing to decrement from zero after a time-out, the timer automatically transfers the con tents of the latch into the counter (16 bits) and continues to decrement from there. The interrupt flag can be cleared by writing TIC-H, by reading TIC-L or by writing directly into the flag as described below. However, it is not necessary to rewrite the timer to enable setting the interrupt flag on the next time-out.

Rewriting the counter will always re-initialise the time-out period. In fact, the time-out can be

prevented completely if the processor continues to rewrite the timer before it reaches zero. Timer 1 will operate in this manner if the processor writes into the high-order counter (TIC-H). However, by loading the latches only, the processor can access the timer during each counting-down operation without affecting the time-out in progress. Instead, the data loaded into the latches will determine the length of the next time-out period. This capability is particularly valuable in the free-running mode with the output enabled. In this mode, the signal on PB7 is inverted and the interrupt flag is set with each time-out. By responding to the interrupts with the new data for the latches, the processor can determine the period of the next half cycle during each half cycle of the output signal on PB7. In this manner, very complex pulse width modulated waveforms can be generated.

9.6 Timer 2

Timer 2 operates as an interval timer (in the "one-shot" mode only), or as a counter for counting negative pulses on the PB6 peripheral pin. A single control bit is provided in the Auxiliary Control Register (ACR) to select between these two modes. This timer is comprised of a "write-only" low-order latch (T2L-L), a "read-only" low-order counter and a read/write high-order counter. The counter registers act as a 16-bit counter which decrements at system clock rate (2 MHz).

Timer 2 addressing can be summarised as in Table 9.6.

Register Number	Write	Read
8	Write T2L-L	Read T2C-L Clear Interrupt Flag
9	Write T2C-H Transfer T2L-L to T2C-L Clear Interrupt Flag	Read T2C-H

Table 9.6

9.6.1 Timer 2 Interval Timer Mode

As an interval timer, T2 operates in the "one-shot" mode similar to Timer 1. In this mode, T2 provides a single interrupt for each "write T2C-H" operation. After timing out, the counter will continue to decrement. However, setting of the interrupt flag will be disabled after initial time-out so that it will not be set by the counter continuing to decrement through zero. The processor must rewrite T2C-H to enable setting of the interrupt flag. The interrupt flag is cleared by reading T2C-L or by writing T2C-H.

9.6.2 Timer 2 Pulse Counting Mode

In the pulse counting mode, T2 serves primarily to count a pre-determined number of negative-going pulses on PB6. This is accomplished by first loading a number into T2. Writing into T2C-H clears the interrupt flag and allows the counter to decrement each time a pulse is applied to PB6. The interrupt flag will be set when T2 reaches zero. At this time the counter will continue to decrement with each pulse on PB6. However, it is necessary to rewrite T2C-H to allow the interrupt flag to set on subsequent down-counting operations.

9.7 Shift Register

The Shift Register (SR) performs serial data transfers into and out of the CB2 pin under control of an internal modulo-8 counter. Shift pulses can be applied to the CB1 pin from an external source or, with the proper mode selection, shift pulses generated internally will appear on the CB1 pin for controlling shifting in external devices.

The control bits which allow control of the various shift register operating modes are located in the Auxiliary Control Register (ACR). These bits can be set and cleared by the system processor to select one of the operating modes discussed in the following paragraphs.

9.7.1 Shift Register Input Modes

Auxiliary Control Register ACR4 selects the shift register input or output mode. There are three input modes and four output modes, differing primarily in the source of the pulses which control the shifting operation. With ACR4=0 the input modes are selected by ACR3 and ACR2 as shown in Table 9.7.1.

ACR4	ACR3	ACR2	
0	0	0	Shift Register Disabled
0	0	1	Shift in under Control of Timer 2
0	1	0	Shift in at system clock rate
0	1	1	Shift in under Control of External Input Pulses

Table 9.7.1

All Shift Register inputs are sampled into the Shift Register during the system clock low immediately following the detection of the shift clock rising transition. This detection occurs during system clock high.

9.7.2 Mode 000 - Shift Register Disabled

The 000 mode is used to disable the Shift Register. In this mode the microprocessor can write or read the SR, but the shifting operation is disabled and operation of CB1 and CB2 is controlled by the appropriate bits in the Peripheral Control Register (PCR). In this mode the SR Interrupt Flag is disabled (held to a logic 0).

9.7.3 Mode 001 - Shift in Under Control of Timer 2

In this mode the shifting rate is controlled by the low-order eight bits of T2. Shift pulses are generated on the CB1 pin to control shifting in external devices. The time between transitions of this output clock is a function of the system clock (2 MHZ) period and the contents of the low-order T2 latch.

The shifting operation is triggered by writing or reading the Shift Register. Data are shifted first into the low-order bit of SR and are then shifted into the next-higher-order bit of the Shift Register on the trailing edge of each clock pulse. As shown in Figure 6-10, the input data should change on the negative edge of the clock pulse. These data are loaded into the Shift Register during the system clock cycle following the positive edge of the clock pulse. After eight clock pulses, the Shift Register Interrupt Flag will be set.

9.7.4 Mode 010 - Shift in at System Clock Rate

In this mode the shift rate is a direct function of the system clock frequency (2 MHz). CB1 becomes an output which generates shift pulses for controlling external devices. The shifting operation is triggered by reading or writing the Shift Register. Data is shifted first into bit 0 and are then shifted into the next-higher-order bit of the Shift Register on the positive edge of each clock pulse. After nine clock pulses, the Shift Register Interrupt Flag will be set, and the output clock pulses on CB1 will stop.

9.7.5 Mode 011 - Shift in Under Control of External Source

In this mode CB1 becomes an input. This allows an external device to load the shift register at its own pace. The shift register counter will interrupt the processor each time 8 bits have been shifted in. However, the shift register counter does not stop the shifting operation; it acts simply as a pulse counter. Reading or writing the Shift Register resets the Interrupt Flag and initialises the SR counter to count another eight pulses.

Note that data is shifted during the first system clock cycle following the positive edge of the CB1 shift pulse. For this reason, data must be held stable during the first full cycle following CB1 going high.

9.8 Shift Register Output Modes

The four shift register output modes are selected by setting the input/output control bit (ACR4) to a

logic 1 and then selecting the specific output mode with ACR 3 and ACR2. In each of these modes the shift register shifts data out of bit 7 to the CB2 pin. At the same time the contents of bit 7 are shifted back into bit 0. As in the input modes, CB1 is used either as an output to provide shifting pulses or as an input to allow shifting from an external pulse. The four modes are as shown in Table 9.8.

ACR4	ACR3	ACR3	Mode
1	0	0	Shift Out – Free-Running Mode Shift Rate Controlled by T2
1	0	1	Shift Out – Shift Rate Controlled by T2
1	1	0	Shift Out at System Clock Rate
1	1	1	Shift Out Under Control of an External Pulse

Table 9.8

All shift register outputs are set during system clock low immediately following the detection of the shift clock falling transitions. This occurs during system clock high.

9.8.1 Mode 100 - Free-Running Output
This mode is very similar to mode 101 in which the shifting rate is set by T2. However, in mode 100 the SR Counter does not stop the shifting operation. Since the Shift Register bit 7 (SR7) is recirculated back into bit 0, the eight bits loaded into the Shift Register will be clocked onto CB2 repetitively. In this mode the shift register counter is disabled. 9.8.2 Mode 101 - Shift Out Under Control of Timer 2

In this mode the shift rate is controlled by Timer 2. However, with each read or write of the Shift Register the SR Counter is reset and 8 bits are shifted onto CB2. At the same time, eight shift pulses are generated on CB1 to control shifting in external devices. After the eight shift pulses, the shifting is disabled, and the SR Interrupt Flag is set. If the Shift Register is reloaded before the last time-out, the shifting will continue.

9.8.5 Mode 110 - Shifting Out at System Clock Rate
In this mode the shift register operation is similar to that of mode 101. However, the shifting rate is a function of the system clock rate and is independent of T2. Timer 2 resumes its normal function as an independent interval timer.

9.8.4 Mode 111 - Shift Out Under Control of an External Pulse
In this mode, shifting is controlled by pulses applied to the CB1 pin by an external device. The SR counter sets the SR Interrupt flag each time it counts eight pulses but it does not disable the shifting function. Each time the system processor writes or reads the shift register, the SR Interrupt flag is reset and the SR Counter is initialised to begin counting the next eight shift pulses on pin CB1. After eight shift pulses, the interrupt flag is set. The system processor can then load the shift register with the next byte of data.

9.9 Interrupt Control

Controlling interrupts within the 65C22 involves three principal operations; flagging the interrupts, enabling interrupts and signalling to the processor that an active interrupt exists within the chip. Interrupt flags are set by interrupting conditions which exist within the chip or on inputs to the chip. These flags normally remain set until the interrupt has been serviced. To determine the source of an interrupt, the system processor must examine these flags in order from highest to lowest priority. This is accomplished by reading the flag register into the processor accumulator, shifting this register either right or left and then using conditional branch instructions to detect an active interrupt. Associated with each interrupt flag is an interrupt enable bit. This bit can be set or cleared by the processor to enable interrupting the processor from the corresponding interrupt flag. If an interrupt flag is set to a logic 1 by an interrupting condition, and the corresponding interrupt enable bit is set to a 1, the Interrupt Request Output (I _R _Q _) will go low interrupting

the system processor.

In the 65C22, all the interrupt flags are contained in one register. In addition, bit 7 of this register will be read as a logic 1 when an interrupt exists within the chip. This permits very convenient polling of several devices within a system to locate the source of an interrupt.

	7	6	5	4	3	2	1	0
Interrupt Flag Register	IRQ	T1	T2	CB1	CB2	SR	CA1	CA2
Interrupt Enable Register	Set/ clear control	T1	T2	CB1	CB2	SR	CA1	CA2

Table 9.9

9.10 Interrupt Flag Register (IFR)

Bit 7 of the IFR indicates the status of the I _R _Q _ output. This bit corresponds to the logic function : IRQ = IFR6 x IER6 + IFR5 x IER5 + IFR4 x IER4 + IFR3 x IER3 + IFR2 x IER2 + IFR1 x IER1 + IFR0 x IER0.
Note: x = logic AND, + = Logic OR.

Bits six through zero are latches which are set and cleared as shown in Table 9.10.

Bit	Set by	Cleared by
0	Active transition of the signal on the CA2 pin	Reading or writing the A Port Output Register (ORA) using address 0001
1	Active transition of the signal on the CA1 pin	Reading or writing the A Port Output Register (ORA) using address 0001
2	Completion of eight shifts	Reading or writing the Shift Register
3	Active transition of the signal on the CB2 pin	Reading or writing the B Port Output Register
4	Active transition of the signal on the CB1 pin	Reading or writing the B Port Output Register
5	Time-out of Timer 2	Reading T2 low order counter or writing T2 high order counter
6	Time-out of Timer 1	Reading T1 low order counter or writing T1 high order latch

Table 9.10

In addition to the clearing operations shown in the table, individual bits in the IFR can be cleared by writing into the register. A logic 1 in the data word written into the IFR will clear the corresponding interrupt flag. A zero in this word will leave the corresponding flag untouched. Setting the flag occurs only from interrupting conditions within the chip.

The IFR bit 7 is not a flag. Therefore, this bit is not directly cleared by writing a logic 1 into it. It can only be cleared by clearing all the flags in the register or by disabling all the active interrupts as discussed in the next section.

9.11 Interrupt Enable Register (IER)

For each interrupt flag in the IFR, there is a corresponding bit in the IER. The system processor can set or clear selected bits in this register to facilitate controlling individual interrupts without affecting others.

If bit 7 is 0 during a write operation, each 1 in bits 6 through 0 clears the corresponding bit in the IER. For each 0 in bits 6 through 0 the corresponding bit is unaffected. If bit 7 is a 1 during a write operation each 1 in bits 6 through 0 sets the corresponding bit in the IER. For each 0 int bits 6 through 0 the corresponding bit is unaffected.

In addition to setting and clearing IER bits, the processor can read the contents of this register. Bit 7 will be read as a logic 0.

9.12 Function Control

Control of the various functions and operating modes within the 65C22 is accomplished primarily through two register, the Peripheral Control Register (PCR), and the Auxiliary Control Register (ACR). The PCR is used primarily to select the operating mode for the four peripheral control pins. The ACR selects the operating mode for the interval timers (T1, T2) and the serial port (SR).

9.13 Peripheral Control Register

The Peripheral Control Register is organised as shown in Table 9.13.

Bit No	7	6	5	4	3	2	1	0
Function	CB2 Control			CB1 Control	CA2 Control			CA1 Control

Table 9.13

Each of these functions is discussed in detail below.

9.13.1 CA1 Control

Bit 0 of the PCR selects the active transition of the input signal applied to the CA1 interrupt input pin.. If this bit is a logic 0, the CA1 interrupt flag will be set by a negative transition (high to low) of the signal on the CA1 pin. IF PCR0 is a logic 1, the CA1 interrupt flag will be set by a positive transition (low to high) of this signal.

9.13.2 CA2 Control

The CA2 pin can be programmed to act as an interrupt input or as a peripheral control output. As an input, CA2 operates in two modes, differing primarily in the methods available for resetting the interrupt flag. Each of these two input modes can operate with either a positive or a negative active transition as described above for CA1.

In the output mode, the CA2 will perform either a "Read" or a "write" handshake operation. The CA2 operating modes are selected as shown in Table 9.13.2.

PCR3	PCR2	PCR1	Mode
0	0	0	CA2 Negative Edge Interrupt (IFR0/ORA Clear) Mode -- Set CA2 interrupt flag (IFR0) on a negative transition of the input signal. Clear IFR0 on a read or write of the Peripheral A Output Register (ORA) or by writing logic 1 into IFR0.
0	0	1	CA2 Negative Edge Interrupt (IFR0 Clear) Mode -- Set IFR0 on a negative transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag. Clear IFR0 by writing logic 1 into IFR0.

0	1	0	CA2 Positive Edge Interrupt (IFR0/ORA Clear) Mode -- Set CA2 interrupt flag on a positive transition of the CA2 input signal. Clear IFR0 with a read or write of the Peripheral A Output Register.
0	1	1	CA2 Positive Edge Interrupt (IFR Clear) Mode -- Set IFR0 on a positive transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag. Clear IFR0 by writing logic 1 into IFR0.
1	0	0	CA2 Handshake Output Mode -- Set CA2 output low on a read or write of the Peripheral A Output Register. Reset CA2 high with an active transition on CA1.
1	0	1	CA2 Pulse Output Mode -- CA2 goes low for one cycle following a read or write of the Peripheral A Output Register.
1	1	0	CA2 Output Low Mode -- The CA2 output is held low in this mode.
1	1	1	CA2 Output High Mode -- The CA2 output is held high in this mode.

Table 9.13.2

In the interrupt (IFR0 clear) input mode, writing or reading the ORA register has no effect on the CA2 interrupt flag. This flag must be cleared by writing a logic 1 into the appropriate IFR bit. This mode allows the processor to handle interrupts which are independent of any operations taking place on the peripheral data ports.

The handshake and pulse output modes have been described previously. Note that the timing of the output signal varies slightly depending on whether the operation is initiated by a read or a write. 9.13.3 CB1 Control

Control of the active transition of the CB1 input signal operates in exactly the same manner as that described above for CA1. If PCR4 is a logic 1, the CB1 interrupt flag (IFR4) is a logic 1, the CB1 interrupt flag (IFR4) will be set by a negative transition of the CB1 input signal and cleared by a read or write of the ORB register. If PCR4 is a logic one, IFR4 will be set by a positive transition of CB1.

If the Shift Register function has been enabled, CB1 will act as an input or output for the shift register clock signals. In this mode, the CB1 interrupt flag will still respond to the selected transition of the signal on the CB1 pin.

PCR7	PCR6	PCR5	Mode
0	0	0	CB2 Negative Edge Interrupt (IFR3/ORB Clear) Mode -- Set CB2 interrupt flag (IFR3) on a negative transition of the CB2 input signal. Clear IFR3 on a read or write of the Peripheral B Output Register (ORB) or by writing a logic 1 into IFR3.
0	0	1	CB2 Negative Edge Interrupt (IFR3 Clear) Mode -- Set IFR3 on a negative transition of the CB2 input signal. Reading or writing ORB does not clear the interrupt flag. Clear IFR3 by writing logic 1 into IFR3.
0	1	0	CB2 Positive Edge Interrupt (IFR3/ORB Clear) Mode -- Set CB2 input signal. Clear the CB2 interrupt flag on a read or write of the ORB or by writing logic 1 into

			IFR3.
0	1	1	CB2 Positive Edge Interrupt (IFR3 Clear) Mode -- Set IFR3 on a positive transition of the CB2 input signal. Reading or writing ORB does not clear the CB2 interrupt flag. Clear IFR3 by writing logic 1 into IFR3.
1	0	0	CB2 Handshake Output Mode -- Set CB2 low on a write ORB operation. Reset CB2 high with an active transition of the CB1 input signal.
1	0	1	CB2 Pulse Output Mode -- Set CB2 low for one cycle following a write ORB operation
1	1	0	CB2 Manual Output Low Mode -- The CB2 output is held low in this mode.
1	1	1	CB2 Manual Output High Mode -- The CB2 output is held high in this mode.

Table 9.13.4

9.13.4 CB2 Control

With the serial port disabled, operation of the CB2 pin is a function of the three high-order bits of the PCR. The CB2 modes are very similar to those described previously for CA2. These modes are selected as shown in Table 9.13.4.

9.14 Auxiliary Control Register

Many of the functions in the ACR have been discussed previously. However, a summary of this register is presented here as a convenient reference. The ACR is organised as shown in Table 9.14.

Bit No	7	6	5	4	3	2	1	0
Function	T1 Control		T2 Control	Shift Register Control			PB Latch Enable	PA Latch Enable

Table 9.14

9.14.1 PA Latch Enable

The 65C22 provides input latching on both the PA and PB ports. In this mode, the data present on the peripheral A input pins will be latched within the chip when the CA1 interrupt flag is set. Reading the PA port will result in these latches being transferred into the system processor. As long as the CA1 interrupt flag is set, the data on the peripheral pins can change without affecting the data in the latches. This input latching can be used with any of the CA2 input or output modes.

It is important to note that on the PA port, the system processor always reads the data on the peripheral pins (as reflected in the latches). For output pins, the processor still reads the latches. This may or may not reflect the data currently in the ORA. Proper system operation requires careful planning on the part of the system designer if input latching is combined with output pins on the peripheral ports.

Input latching is enabled by setting bit 0 in the ACR to a logic 1. As long as this bit is a 0, the latches will directly reflect the data on the pins. 9.14.2 PB Latch Enable

Input latching on the PB port is controlled in the same manner as that described for the PA port. However, with the peripheral B port the input latch will store either the voltage on the pin or the contents of the Output Register (ORB) depending on whether the pin is programmed to act as an input or an output. As with the PA port, the system processor always reads the input latches.

9.14.3 Shift Register Control

The Shift Register operating mode is selected as shown in Table 9.14.3.

ACR4	ACR3	ACR2	Mode
0	0	0	Shift register disabled
0	0	1	Shift in under control of Timer 2
0	1	0	Shift in under control of o2 pulses
0	1	1	Shift in under control of external clock pulses
1	0	0	Free-running output at rate determined by Timer 2
1	0	1	Shift out under control of Timer 2
1	1	0	Shift out under control of the o2 pulses
1	1	1	Shift out under control of external clock pulses

Table 9.14.3

9.14.4 T2 Control

Timer 2 operates in two modes. If ACR5 = 0, T2 acts as an interval timer in the one-shot mode. If ACR5 = 1, Timer 2 acts to count a predetermined number of pulses on pin PB6.

9.15.5 T1 Control

Timer 1 operates in the one-shot or free-running mode with the PB7 output control enabled or disabled. These modes are selected as shown in Table 9.14.5

ACR7	ACR6	Mode
0	0	One-Shot Mode -- Output to PB7 disabled
0	1	Free-Running Mode -- Output to PB7 disabled
1	0	One-Shot Mode -- Output to PB7 enabled
1	1	Free-Running Mode -- Output to PB7 enabled

Table 9.14.5

Appendix I

Expansion Card DIP Switch Settings

Switch	Connector	Pin Number	Designation	Direction On	Direction Off
SW1	PL2	4	LCA2	In	Out
SW2	PL2	6-20	LPA0-7	Out	In
SW3	PL2	24	MCA2	In	Out
SW4	PL2	26-40	MPA0-7	Out	In
SW5	PL3	2	LCB1	Out	In
SW6	PL3	4	LCB2	In	Out
SW7	PL3	6-20	LPB0-7	In	Out
SW8	PL3	22	MCB1	Out	In
SW9	PL3	24	MCB2	In	Out
SW10	PL3	26-40	MPB0-7	In	Out
Note					
	PL2	2	LCA1	Always In	
	PL3	22	MCA1	Always In	

Default setting SW1 - SW10 all off making PL2 a 16 bit input port and PL3 a 16 bit output port.

A VERY IMPORTANT NOTE WHEN PROGRAMMING
Following a hardware reset all peripheral lines of the 65C22' s are configured as inputs. Before programming a line as an output make absolutely sure that the direction of the corresponding buffer, as determined by the DIP switch, is as an output or damage to the IC' s may result.

Appendix II

Expansion Card Option Selection Links

Link	Position A	Position B	Default
S1			A
	Simple Expansion Card I.D.	See following table	
S2			A
S3	LCA1 and MCA1 both from PL2 pin 2	LCA1 from PL2 pin 2 and MCA1 from PL2 pin 22	B
S4	PL2 pins 1 and 3 to +5V	PL2 pins 1 and 3 to 0V	B
S5	PL2 pins 21 and 23 to +5V	PL2 pins 21 and 23 to 0V	B
S6	LCB1 and MCB1 both from PL3 pin 2	LCB1 from PL3 pin 2 and MCB1 from PL3 pin 22	B
S7	PL3 pins 1 and 3 to +5V	PL3 pins 1 and 3 to 0V	B
S8	PL3 pins 21 and 23 to +5V	PL3 pins 21 and 23 to 0V	B
S9	expansion card not present	expansion card present	B

Simple Expansion Card I.D.

I.D.	Link S1	Link S2
12 (&C)	B	B
13 (&D)	A	B
14 (&E)	B	A
15 (&F)	A	A

Note - the simple Expansion Card I.D. ranges from 12 (&C) to 15 (&F) as the BD(5) and BD(6) data lines of the expansion backplane are pulled up setting I.D.[2] and I.D.[3] both to 1 giving a offset of 12 (&C). With version 1.2 of the operating system of the Archimedes Computer the simple Expansion Card I.D. is not relevant and, therefore, can be left as the default setting.

Appendix III

Expansion Card Rear Panel Connectors

Pin Designations

PL2 (Upper Connector)

Number	Description	Number	Description
1	0V or +5V (see Appendix II)	2	LCA1
3	0V or +5V (see Appendix II)	4	LCA2
5	0V	6	LPA0
7	0V	8	LPA1
9	0V	10	LPA2
11	0V	12	LPA3
13	0V	14	LPA4
15	0V	16	LPA5
17	0V	18	LPA6
19	0V	20	LPA7
21	0V or +5V (see Appendix II)	22	MCA1
23	0V or +5V (see Appendix II)	24	MCA2
25	0V	26	MPA0
27	0V	28	MPA1
29	0V	30	MPA2
31	0V	32	MPA3
33	0V	34	MPA4
35	0V	36	MPA5
37	0V	38	MPA6
39	0V	40	MPA7

PL3 (Lower Connector)

Number	Description	Number	Description
1	0V or +5V (see Appendix II)	2	LCB1
3	0V or +5V (see Appendix II)	4	LCB2
5	0V	6	LPB0
7	0V	8	LPB1
9	0V	10	LPB2
11	0V	12	LPB3
13	0V	14	LPB4
15	0V	16	LPB5
17	0V	18	LPB6
19	0V	20	LPB7
21	0V or +5V (see Appendix II)	22	MCB1
23	0V or +5V (see Appendix II)	24	MCB2
25	0V	26	MPB0
27	0V	28	MPB1
29	0V	30	MPB2
31	0V	32	MPB3
33	0V	34	MPB4
35	0V	36	MPB5
37	0V	38	MPB6
39	0V	40	MPB7

Appendix IV

Expansion Card Identification Byte

```
BD[0]   IRQ       0 = not req IRQ
                  1 = req IRQ

BD[1]    P        0 = expansion card present
                  1 = expansion card not present

BD[2]    FIQ      always 0 = not req IFQ

BD[6:3]  ID[3:0]  Simple Expansion card I.D.

BD[7]    A        always 1 = other manufacturer than Acorn
```