Derby Technical Guide

Released 7/11/85

1 INTRODUCTION

This document is a brief description for software developers of the main features of the Derby microcomputer.

* * * * * * * * * * * * * * * * * *

2 HARDWARE SPECIFICATION

The Derby is a Z80A based microcomputer with 128K of bank switched Ram and 32K Rom memory. Its external interfaces and outputs comprise.

1) Cassette port	As for Spectrum Plus
2) TV output	
3) Expansion bus	
4) RS-232/Midi-out port	Derby specific
5) RGB-monitor output	
6) Keypad	
7) TV sound	

A major consideration of the Derby design has been to make it as far as possible software and hardware compatible with the Spectrum. Where information is not given the system will behave in the same fashion as the Spectrum eg screen layout, cassette data format.

* * * * * * * * * * * * * * * * * *

3 MEMORY MAP

The Derby contains 32K Rom and 128K Ram arranged in 16K byte pages.

The two ROM pages (0-1) are mapped into the bottom 16K (0-3FFF) of the Z80 memory map.

The eight RAM pages (0-7) are mapped into the top 16K (C000-FFFF) of the memory map, RAM page 5 is also mapped to the range 4000-7FFF while RAM page 2 is mapped to the range 8000-BFFF.

c is thus possible, though not very useful, to have the same RAM page mapped into two different address spaces within the Z80.

The RAM pages are divided into a video contended section, pages 4 to 7, and a non contended section pages 0 to 3.

The Derby has two possible hardware screen bases, screen 0 and screen 1. Screen 0 is used by the system and resides in RAM page 5 corresponding to the normal Spectrum screen. The system software does not support the use of the second screen base, screen 1, which is realistically available only to machine code applications programs. The second screen resides in RAM page 7 and thus maps, when paged into the Z80 address space, onto memory address C000.

	Physical Memory			Z80 Memory	
	==========			==========	
RAM	page 7	screen 1	contended	Paging	Address
	page 6		contended	Ram page 0-7	C000-FFFF
	page 5	screen 0	contended	Ram page 2	8000-BFFF
	page 4		contended	Ram page 5	4000-7FFF
	page 3		uncontended	Rom page 0-1	0000-3FFF
	page 2		uncontended		
	page 1		uncontended		
	page 0		uncontended		
ROM	page 1	Spectrum			
	page 0	Derby			

* * * * * * * * * * * * * * * * * *

4 I/O MAP

The Spectrum i/o addresses reserve A4-A0 for use by Sinclair Research. These are active low and decoded by the presence of a single signal. Unused addresses in the range A4-A0 should be set high.

Line	Address	Function
A0	FE	Spectrum keyboard, cassette, loudspeaker and border.
A1	FD	Derby paging, screen selection, sound and i/o.
A2	FB	ZX Printer
A3,A4	F7/EF	Interface One

/he following addresses are only found on the Derby.

7FFD	D2-D0	Ram Page select
	D3	Screen select
	D4	Rom select
	D5	lock (become a Spectrum)
BFFD	Sound chip -	data register write (register dependent)
FFFD	Sound chip -	data read (register dependent)
	-	address write 0000XXXX, where XXXX is the register selected 0-F

* * * * * * * * * * * * * * * * * *

5 SOUND CHIP

The sound generator on the Derby is the General Instruments AY-3-8912. This device has three sound channels and an 8-bit i/o port which is used to control the RS232/MIDI port and keypad.

The sound chip contains sixteen registers which are selected by writing first to the address write port with the number of the register and then writing or reading from the data write or data read ports. Once a register has been selected with an address write, it can be accessed repeatedly by data read/writes until a new register is selected.

The basic clock input to the sound chip is at 1.7734(476)MHz accurate 1 in 10+4

The registers have the following properties

R0 - Fine Tone Control for Channel A
R1 - Coarse Tone Control for Channel A

The tone is a 12-bit value taken from the sum of R1,D3-D0 and R0,D7-D0. The extra bits in R1,D7-D4 are unused. The sound clock is divided by 16 to get the basic frequency unit for the Tone and Noise registers. With a twelve bit counter range frequencies in a range from 27Hz to 100kHz can be generated.

R2 - Fine Tone Control for Channel B
R3 - Coarse Tone Control for Channel B

R4 - Fine Tone Control for Channel C
R5 - Coarse Tone Control for Channel C

R6 - Noise Generator Control, D4-D0

* * * * * * * * * * * * * * * * * *

7 SOFTWARE ORGANISATION

The Derby Rom contains the routines associated with the program editor, music control (PLAY), RS-232 and Ram disk. The Spectrum Rom contains the Basic language and the keypad routines.

Ram page 7 contains the ram storage for the editor and ram disk directory.

Ram page 0 is the ram used for program storage by Basic.

* * * * * * * * * * * * * * * * * *

8 ROM VECTORS

In the Derby Rom there are a number of useful Rom vectors, suitable for use by machine code programs. No other routines in the DERBY Rom should be used by programmers as the contents and organisation of the rom are liable to change.

Routine	Address	Function
KPSCAN	118H	Scan the Derby keypad
MSHOOK	11AH	Play music strings
MIDI_SEND	11EH	Send a MIDI character
RSIN	121H	Receive an RS232 character
OUT_T	124H	Send a token to RS232
OUT_T2	127H	Send a character to RS232
SCRDMP	12AH	Epson compatible RS232 screen dump

These routines will corrupt all unused registers

```
****************************************
*    KPSCAN - DERBY ROM 118H           *
****************************************
```

call this routine to service the keypad

NB interrupts must be disabled on entry

exit conditions are as follows :

ROW01, ROW23, and ROW45 are left holding the instantaneous keypad image

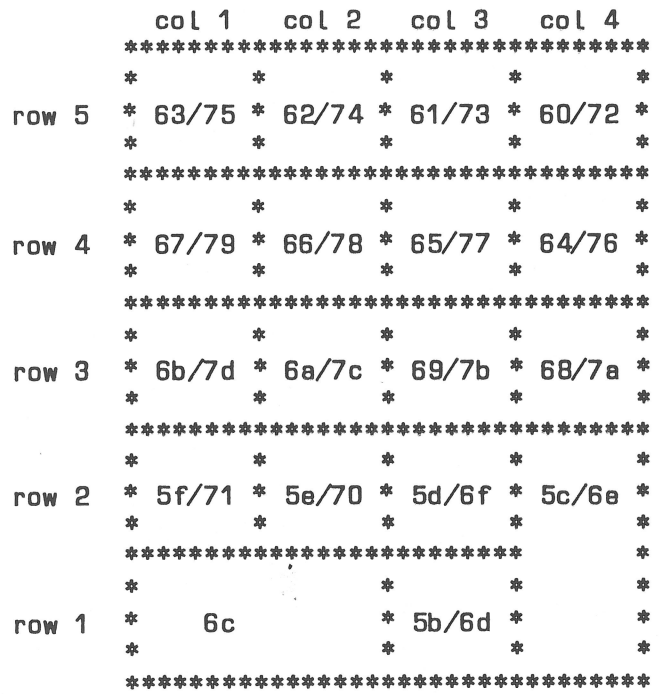
z flag is clear: something's wrong
- no pp device is connected

- no keypad is connected
- more than 1 key is pressed (not including 0/SHIFT)

z flag is set: everything's ok,
an intermediate key code is returned in E
if no key was pressed then E is 10000000
else E is 0kkkkkkk

intermediate key codes are returned as follows :

calculator legend	row	col	key only	key and 0/SHIFT
0	1	1		6c (108)
.	1	3	5b (91)	6d (109)
ENTER	2	4	5c (92)	6e (110)
3	2	3	5d (93)	6f (111)
2	2	2	5e (94)	70 (112)
1	2	1	5f (95)	71 (113)
)	5	4	60 (96)	72 (114)
(	5	3	61 (97)	73 (115)
*	5	2	62 (98)	74 (116)
/	5	1	63 (99)	75 (117)
-	4	4	64 (100)	76 (118)
9	4	3	65 (101)	77 (119)
8	4	2	66 (102)	78 (120)
7	4	1	67 (103)	79 (121)
+	3	4	68 (104)	7a (122)
6	3	3	69 (105)	7b (123)
5	3	2	6a (106)	7c (124)
4	3	1	6b (107)	7d (125)



****
* MSHOOK - DERBY ROM 11BH *
****

This entry point allows access to the music system from machine code. It requires strings to be set up in the same form as the Basic PLAY command.

The data structure required is described below.

Index registers

The music routines in the DERBY ROM use two control blocks accessed indirectly via IY and IX.

IY points to a control block which contains 'system' information about all the strings that are currently being interpreted. This block must be at least 60 bytes long. The variables in this block are :

Offset

```
CT_CHAN0        EQU     0               ; value of IX for channel 0
CT_CHAN1        EQU     CT_CHAN0+2      ; --------||------------ 1
CT_CHAN2        EQU     CT_CHAN1+2      ; --------||------------ 2
CT_CHAN3        EQU     CT_CHAN2+2
CT_CHAN4        EQU     CT_CHAN3+2
CT_CHAN5        EQU     CT_CHAN4+2
CT_CHAN6        EQU     CT_CHAN5+2
CT_CHAN7        EQU     CT_CHAN6+2
CT_FLAGS        EQU     CT_CHAN7+2      ; Flags
                                        ; bit i set if string i finished
CT_Q0           EQU     CT_FLAGS+1      ; queue pointer for channel 0
CT_Q1           EQU     CT_Q0+2         ; --------||------------  1
CT_Q2           EQU     CT_Q1+2         ; --------||------------  2
CT_Q3           EQU     CT_Q2+2
CT_Q4           EQU     CT_Q3+2
CT_Q5           EQU     CT_Q4+2
CT_Q6           EQU     CT_Q5+2
CT_Q7           EQU     CT_Q6+2
CT_CHAN         EQU     CT_Q7+2         ; temp store for chan indicator
CT_TEMP         EQU     CT_CHAN+1       ; temp store for copy of flags
CT_QTEMP        EQU     CT_TEMP+1       ; temp storage for pointer to Q ptr
CT_EVENT        EQU     CT_QTEMP+2      ; length of current event in T states
CT_TEMP0        EQU     CT_EVENT+2      ; no of dec BC loops for 1 T state
CT_ENV          EQU     CT_TEMP0+2      ; current envelope shape/cycle byte
CT_MIXT         EQU     CT_ENV+1        ; temporary mixer mask
CT_CODE         EQU     CT_MIXT+1       ; RAM code for tempo adjustment
```

IX points to a buffer for the string currently being processed. For 8 strings there will be 8 of these buffers and the higher level software switches the value of IX between them. Note that the values of IX are stored at the start of the IY control buffer. An IX buffer must be at least 55 bytes long. The buffer variables are :

```
MV_CURR         EQU     0               ; current MIDI note
MV_MIDI         EQU     MV_CURR+1       ; MIDI channel number (byte)
MV_CHAN         EQU     MV_MIDI+1       ; channel number (0,1 or 2) (byte)
MV_OCTAVE       EQU     MV_CHAN+1       ; current octave (byte)
MV_VOL          EQU     MV_OCTAVE+1     ; current volume (byte)
MV_NOTE         EQU     MV_VOL+1        ; current note code (byte)
MV_ADD          EQU     MV_NOTE+1       ; pointer to next char (word)
MV_END          EQU     MV_ADD+2        ; pointer to end of string (word)
MV_REPEAT       EQU     MV_END+2        ; pointer to last repeat (word)
MV_FLAG         EQU     MV_REPEAT+2     ; misc flags (byte)
MV_OPEN         EQU     MV_FLAG+1       ; open bracket stack(byte+5 words)
MV_CLOSE        EQU     MV_OPEN+11      ; close bracket stack(byte+5 words)
MV_PEND         EQU     MV_CLOSE+11     ; notes in queue (byte)
MV_QUEUE        EQU     MV_PEND+1       ; start of queue (20 words)
```

The string interpreter

In order to provide string interpretation for machine code programmers an entry point to the code has been provided at the global MUSIC_HOOK. On entry to this point the calling code must have set up a control block at IY and music buffers for each string to be interpreted (up to a maximum of 8 strings). The control block must have the following parameters set :

```
(IY+CT_CHAN0)   must contain the value of IX for the first string.
(IY+CT_CHAN1)   ----------------------------------2nd---------.
                        etc up to 8 strings if necessary.
(IY+CT_FLAGS)   must have reset bits for strings to be played and
                set bits for absent strings
                Bit 0 is the first string etc.
```

On entry the code will set up the default tempo to 120 crotchets per min.

Each music buffer must have the following parameters set :

(MV_MIDI) OFFH if MIDI output is not required otherwise the MIDI channel number (0..15)

(MV_CHAN) The channel number for this string. The first string is channel 0 and so on.

(MV_OCTAVE) The default octave for the base of the 2 octave range. A value 5 gives note code c as middle C.

(MV_VOL) The volume for the GI chip on this channel (0..15)

(MV_NOTE) The default note type (5=crotchet)

(MV_ADD) 16 bit address of the first code in the string. This value should also be copied to (MV_REPEAT) if repeat is required to start from the beginning of the string.

(MV_END) Pointer to the next byte after the end of the string.

(MV_OPEN) Must contain 0 (byte)

(MV_OPEN+1) Must contain the 16 bit start address of the string.

(MV_CLOSE) Must contain OFFH.

On entry to the code interrupts must be disabled. The code will execute a RET on correct termination of all strings. Any errors will jump to the internal error handling routine and control will be lost. The routine will corrupt all normal registers. IY will be returned as the correct pointer to the system variables. The alternate register set is unaffected.

****
* MIDI_SEND DERBY ROM 11E *
****

This routine outputs a byte to the RS-232 port acting as a MIDI-OUT port

The MIDI output routine is accessed via a call to the global address MIDI_SEND. The byte to be sent is in A and is sent immediately. The routine corrupts A,BC,DE and L. Interrupts must be off to ensure correct timing.

****
* RSIN DERBY ROM 121 *
****

RS-232 Receive character routine

If a character is received then C-flag is set and the character is returned in A, else no character C-flag clear

The receive system expects 8-data bits, no parity and 1 stop-bit.

Corrupts all registers

****
* OUT_T DERBY ROM 124 *
****

RS-232 Send a token routine

This routine takes as input a token in A

The baud rate is set up by poking into the Derby system variable BAUD a value equivalent to a bit time in T-states divided by 26

This routine sends data in the format, 8-data, no-parity, 2 stop bits.

Corrupts all registers

```
*****************************************
*  OUT_T2       DERBY ROM  127           *
*****************************************
```

RS-232 Send a character routine

This routine takes as its input a character in A

The baud rate is set up by poking into the Derby system variable BAUD a value equivalent to a bit time in T-states divided by 26

This routine sends data in the format, 8-data, no-parity, 2 stop bits.

Corrupts all registers

```
*****************************************
*  SCRDMP       DERBY ROM  12A           *
*****************************************
```

Dump the screen image to an Epson-compatible printer

Corrupts all registers

This takes no parameters and sends a bit image of the current screen to an Epson compatible printer over the RS-232 interface.

The baud rate is set up by poking into the Derby system variable BAUD a value equivalent to a bit time in T-states divided by 26

* * * * * * * * * * * * * * * * * *

9 SPECTRUM/DERBY DIVERGENCES

The Derby runs in two distinct modes, the first and start-up state is as a 128K machine, the second is as a 48K Spectrum. When running in 48K mode the only detectable difference is that previously unused space in the Spectrum rom now contains the keypad scanning routines. In Derby mode the buffer for the ZX printer is used for extra system variables, so programs that use this area for code space may not function. On the edge connector interface the Z80 clock signal is not connected.

Derby System Variables

These are the new variables associated with 128K mode, they reside in the printer bufffer. The most useful of these to third parties will be BAUD, which allow you to set up the RS-232 speed and ROW01,ROW23,ROW45 which give access to the keypad.

Variable	Address	Function
SWAP	5B00	ROM swapping subroutines
YOUNGER	5B14	
ONERR	5B1D	

PIN	5B2F	Printer channel Input routine
POUT	5B34	Printer channel Output routine
POUT2	5B4A	
TARGET	5B58	Address of subroutine in old ROM
RETADDR	5B5A	Return address in new ROM
BANKM	5B5C	Copy of last byte output to bank
RAMRST	5B5D	RST 8 instruction
RAMERR	5B5E	Error number or old ROM
BAUD	5B5F	Bit period in T states / 26
SERFL	5B61	Second-character-received-flag and data
COL	5B63	Current column from 1 to width
WIDTH	5B64	Paper column width
TVPARS	5B65	Number of parameters exepected by RS232
FLAGS3	5B66	Bit 0 ... Calculator/Edit mode Bit 1 ... BASIC line changed Bit 2 ... Silicon File open for write Bit 3 ... Silicon/cassette SLVM Bit 4 ... Load Bit 5 ... Save Bit 6 ... Merge Bit 7 ... Verify
N_STR1	5B67	SLVM Name

SLVM header blocks

HD_00	5B71	Type code
HD_0B	5B72	Length of block
HD_0D	5B74	Start of block
HD_0F	5B76	Program length
HD_11	5B78	Line number
SC_00	5B7A	Second set for LOAD,VERIFY,MERGE
SC_0B	5B7B	
SC_0D	5B7D	
SC_0F	5B7F	
XLOC	5B71	Screen dump variables
YLOC	5B72	(Dual use of variables)
OLDSP	5B81	Old SP when TSTACK is used
SFNEXT	5B83	Pointer to last (empty) entry in directory
SFSPACE	5B85	Number of bytes left (17 bit)

The following variables return a keypad image when KPSCAN is called

ROW01	5B88	pdmm1111 - present, device, micro, row1
ROW23	5B89	22223333 - row2, row3
ROW45	5B8A	44445555 - row4, row5
SYNRET	5B8B	Return address for ONERR
LASTV	5B8D	Last value printed by calculator
TSTACK	5BFF	Temporary stack used when memory paging

There are a number of minor physical and electrical improvements that will take place between the development machines (Spanish) and the UK production version.

The Z80 clock signal is brought out to the edge connector (Whoops).