
# INFORMATION FOR MACHINE CODE USERS!

Published by Dragon Data Ltd., March 83

## CASSETTE CONNECTIONS

The Dragon 32 may be connected to most audio cassette recorders but some problems may be encountered if the recorder has an automatic level control. To connect your own lead you will require a 5 pin DIN audio plug. The connections are as follows:

| | |
|---|---|
| PIN 1 | Remote (to switch recorder motor on/off) |
| PIN 2 | Ground |
| PIN 3 | Remote |
| PIN 4 | Cassette input to Dragon |
| PIN 5 | Output from Dragon to Cassette |

Whilst we do not recommend a specific cassette recorder for the Dragon 32, the following recorders have been found to work well with the Dragon:-

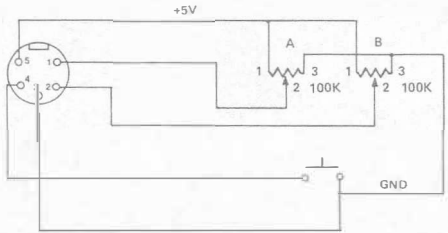
Prinz TR12 and TR15; and Realistic recorder.

Should problems arise with saving and loading check that heads are clean; remove MIC lead whilst loading and EAR lead whilst saving; start recording after plastic header label; move Dragon from close vicinity of television.

## JOYSTICK CONTROLLERS

Joysticks are now available but other joysticks may be connected through the joystick ports using a 5 pin DIN plug. The connections are as follows:

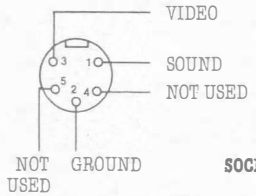
| | |
|---|---|
| PIN 1 | X wiper |
| PIN 2 | Y wiper |
| PIN 3 | 0 volts |
| PIN 4 | Fire button Input |
| PIN 5 | +5 volts |




## MONITOR CONNECTIONS

The Dragon 32 is compatible with composite video signal monitors. To connect a PAL monitor to the Dragon 32, using a 5 pin DIN wire the plug as follows:-

| | |
|---|---|
| PIN 1 | Sound Signal |
| PIN 2 | Ground |
| PIN 3 | Video Signal |
| PIN 4 | Blank |
| PIN 5 | Blank |




**FRONT VIEW OF SOCKET ON DRAGON**

## PRINTERS

The Dragon 32 will accept a wide range of printers currently available. The printer port is the parallel I/O type for centronics interfaces. The cable connections are detailed on Page 2 of the Additional Information booklet supplied with the machine. The connector required is a 20 way cable mounting socket.

The following printers have been found to work well with the Dragon 32:

Seikosha GP100A and GP250A; Epson MX80; NEC PC8023; Amber 2400.

## DISK DRIVES

The Disk Drive system initial package will consist of the following:

**Disk Controller**
Which will fit into the existing cartridge port.

**Disk Unit**
Which will be in a case housing one disk drive and also, a power supply which is then fed out to the 13 Amp mains socket.

The power supply in the disk unit is capable of feeding two drives and a maximum of four drives may be operated from the disk controller.

The disk system is designed to work with either a Dragon 32 or an upgraded 64k machine. With the 32k Dragon a disk operating system will be provided on board the disk controller which will work in conjunction with Microsoft Basic. This DOS may only be used on a 32k system but, it would also be suitable for use by users working in assembly language, in conjunction with an Editor/Assembler program loaded via the tape or disk systems. The operating system for the 64k machine will be OS-9, which will be resident on disk.

## 64k UPGRADE

Due to the format of the memory map, in order to use the upper 32k of the memory map for RAM, the BASIC ROMS will have to be disabled. If this is done, however, there will be no operating system available to the user; i.e., the program will crash or alternatively the screen will be filled with garbage. Therefore, the 64k RAM facility may only be used in conjunction with the disk, having loaded the operating system OS-9 into the lower region of RAM, or alternatively, by users who write their own machine code operating program to switch the upper RAM in under their own control.

## MACHINE CODE

Machine language code can be put into memory using an Editor/Assembler or by a BASIC program using the POKE command. Routines entered in the latter manner can be accessed by using the DEFUSR and USR commands, see pages 134/135 of the manual supplied with the machine.
The Editor/Assembler will be available on cassette and cartridge.

The cartridge version will include a debugger and both versions will enable the user to input programs in Assembley language.

## MEMORY MAP OF DRAGON

| | |
|---|---|
| 0-$FF | Direct Page: Used by BASIC |
| $100-$1FF | Page 1: I/O drivers, Extended BASIC |
| $200-$3FF | Buffers for cassette, etc., |
| $400-$5FF | Text Screen — default area |
| $600-$7FFF | Graphics Screen/Program/Variable storage |
| $8000-$BFFF | BASIC ROMS |
| $C000-$DFFF | Cartridge ROM Locations |
| $FF00-03 | PIO 0 |
| $FF20-23 | PIO 1 |
| $FFC0-DF | SAM chip register |
| $FFF0-FF | Reset vectors |

Reset vectors are actually mapped from BFF0

| $BFF0 | | | **Default settings** |
|---|---|---|---|
| F2 | SW13 | 100 | |
| F4 | SW12 | 103 | |
| F6 | FIRQ | 10F | Initialise Cartridge |
| F8 | IRQ | 10C | Update clock, PLAY, etc., |
| FA | SWI | 106 | |
| FC | NMI | 109 | |
| FE | Reset | B3B4 | Initialise/Warm Start BASIC |

| PIO 0 | Bit | Function |
|---|---|---|
| | 0-6 | Keyboard Row input |
| A side (FF00) | 7 | Joystick Comparator input |
| | 0,1 | Joystick switch input |
| | CA2 | MUX Least significant byte select |
| B side (FF02) | 0-7 | Keyboard Column output<br>Printer output |
| | CB1 | IRQ — vertical sync |
| | CB2 | MUX M.S.B. select output |

**PIO 1**

| | | | |
|---|---|---|---|
| | 0 | Cassette data input | |
| | 1 | Printer strobe | |
| A side (FF20) | 2-7 | Digital-Analogue converter | |
| | CA1 | Printer acknowledge IRQ (not used) | |
| | CA2 | Cassette relay control | |
| B side (FF22) | 0 | Printer busy input | |
| | 1 | Single bit sound output | |
| | 2 | RAM size select sensing | |
| | 3 | CSS | |
| | 4 | GM0 / I/E | video controller control lines |
| | 5 | GM1 | |
| | 6 | GM2 | |
| | 7 | A/G | |
| | CB1 | Cartridge port FIRQ | |
| | CB2 | Sound enable output (to T.V.) | |

**CASSETTE I/O**

| | |
|---|---|
| JSR $8015 | Turn on cassette relay |
| JSR $8018 | Turn off cassette relay |
| JSR $801B or JSR [$A00C] | Prepare cassette for writing |
| $90/91 | Leader byte count |
| $95/96 | Cassette motor delay |
| JSR $801E | Put out a byte to cassette from A |

This is best used as part of BLKOUT (i.e. not directly used). JSR [$A008] will write out a block of data, complete with checksum, once the cassette has been prepared by JSR $8015. Parameters to be set up for BLKOUT.

| | |
|---|---|
| $7C | Block type = 0: file header<br>1: data, FF: End of file |
| $7D | Number of bytes to be put out |
| $7E/F | Base address of bytes to be put to cassette |
| JSR $8021 or JSR [$A004] | Prepares the cassette for data input, getting into BIT sync. |
| JSR $8024 | Inputs the next eight data bits as a byte in A. |
| JSR $8027 | Gets the next bit in from cassette into carry. |

BLKIN: JSR [$AØØ6]

Having been set up by JSR $8021, waits for $3C from tape to get into BYTE sync then reads in the data following into the memory, pointed to by $7E, and does a check sum on the result, also reading block type and byte count. If all O.K. Zero flag set.

| | |
|---|---|
| $81 | Error code: cleared if read and verified correctly. |

As an example of the use of these routines see February ''Personal Computer World'' for a cassette verify.

| | |
|---|---|
| JSR $8006 or JSR [$AØØØ] | Polls the keyboard and returns the character code in accumulator A. If no new key has been pressed, A is cleared, else the ASCII key code is returned (as in the manual) after rollover has been taken into account. |
| $150-159 | Keyboard rollover table — writing $FF to these locations will cause a sort of auto repeat (once for each clearing of the table). |
| $149 | Alpha lock flag (Default $FF). |
| JSR $8012 | Updates the joystick readings stored in $15A-D. |
| JSR $8009 | Blinks the cursor when the count has fallen to Ø. |
| $8F | Blink count. |
| JSR $800C | Writes the character from A to the text screen, scrolling if necessary. The current cursor position is updated to point to the next location of the screen. |
| $88 & 89 | Point to the next location for screen output. |

Default screen address (Text) $400 — $5FF

| | |
|---|---|
| JSR $800F | Writes out the character in A to the lineprinter. |
| $99 | Lineprinter ''comma field'' width |
| $9A | Last ''comma field'' width. |
| $9B | Line printer width. |
| $9C | Line printer: print head position. |
| $148 | (Buffer full) auto line feed flag Default FF ( = on). If Ø then carriage return will be printed at end of line. |
| $14A-14F | End of line termination sequence. |

Printer end of line sequence.

| | | |
|---|---|---|
| 14A | 1 | Number of characters to be printed in E.O.L. sequence. |
| $14B | $ØD | Return |
| $14C | $ØA | Line feed |
| $14D | Ø | null |
| $14E | Ø | null |

## OTHER USEFUL LOCATIONS

| | |
|---|---|
| $19 | Beginning of Text. |
| $1B | Beginning of Simple Variable Space. |
| $1D | Beginning of Array pointer table. |
| $IF | End of storage in use ( = 1st free location) set to [$1B] on clearing. |
| $21 | Top of stack (from which stack grows down). |
| $23 | Top of string free place. |
| $27 | Highest RAM available to BASIC. |
| $2D | Points to statement to be Executed. |
| $2F | Text pointer for BASIC warm start on reset |
| $31 | DATA line number. |
| $33 | DATA memory pointer. |
| $35 | INPUT pointer. |
| $68 | Current line number. |
| $6F | Current device. Ø = VDU, – 1 = cassette, – 2 = printer. |
| $70 | End of file flag: Ø = character found. |
| $71 | RESTART FLAG. If [$71] ≠ $55 then a full cold start will be performed on RESET. |
| $72/73 | Restart vector. If [$71] = $55 and $72 points to a NOP then restart will be to that NOP (else coldstart). |
| $78 | Cassette status: Ø = closed, 1 = input, 2 = open for output. |
| $87 | Last key pressed (but may well have been cleared by BREAK check). |
| $9D/E | EXEC address. |
| $9F | Start of a short self modifying routine to read in the next useful character for BASIC, continued in ROM. |
| $A6/7 | Text pointer (part or self modifying code). Points to current active byte. |
| $BØ | Address of start of USR function address table. |
| $B2 | Foreground colour. |
| $B3 | Background colour. |
| $B4 | Active colour. |
| $B6 | Graphics mode. |
| $B7/8 | Top address of current graphics screen. |
| $B9 | Count of bytes in a row of graphics. |
| $BA/B | Base address of current graphics screen. |
| $BD/E | Current X position. |
| $BF/CØ | Current Y position. |
| $112-4 | Timer $114 cycles 0-$FF in about 5s. |
| $120 | Start of "STUBØ" — Used to set up entry points for BASIC keywords. |
| $120 | Number of normal reserved words. |
| $121 | Address of normal reserved words list. |
| $123 | Address of statement despatch table. |

$125 Number of function reserved words (tokens will be preceded by $FF).
$126 Address of function reserved word list.
$128 Address of function despatch table.

$12A STUB 1: normally a dummy to make end of stubs. Layout as for STUBØ except:-

$12D address of statement despatch routine.
$132 address of function despatch routine.

$134 STUB 2 (as above).
$134-147 Are normally used for the USR function address.

$15E-1AF are subroutines (length 3 bytes) used by the BASIC ROM. By default they return immediately with a RTS, but these can be modified to jump to your own code to provide extensions to the BASIC. Some of the more useful ones.

| Address | ROM routine Called from | |
|---|---|---|
| $167 | Input a character. | |
| $16A | Output a character. | |
| $182 | Read an input line. | |
| $18B | Evaluate an expression. | |
| $18E | User error trap<br>System error trap | error handler |
| $194 | RUN | |
| $19A | Read in next statement (after returning, the keyboard is polled for break, so increasing the return address by 4 will disable BREAK). | |

$1A3 Crunch BASIC line for storing.
$1A6 Decrunch BASIC line for output.

To access machine code from BASIC use A = USRnn(X). (Where nn is a two digit number in the range Ø-9). The entry point required must first be set up by DEFUSRn = XXXX. XXXX is the position of the machine code subroutine, e.g. to call a subroutine at $B7BA you could use DEFUSR3 = &HB78A then A = USRØ3(X).

If the byte pointed to has value Ø then the value passed was zero, otherwise it represents the binary exponent plus 128. The next four bytes represent the absolute value of the mantissa in normalised form. [X] + 5 points to a duplicate of the most significant byte of the mantissa, except the top bit is cleared if the number is positive or set for negative. Variables are stored with this byte in place of the M.S.B.

To get the integer value of the floating point accumulator JSR $8B2D, which returns the value of the F.A.C. in the D register.

To return an integer in the D register as a floating point number terminate the subroutine with JMP $8C37 and the value will be passed back as the value of the USR function.

Strings can be accessed using the VARPTR function. The pointer will be stored in the floating accumulator and will point to a five byte string descriptor. The first byte is the length and the third and fourth contain the address of the start of the string.


Dragon Data Ltd., Kenfig Industrial Estate, Margam, Port Talbot, West Glam.