16.1 Introduction

IS-BASIC was designed to support extension statements and functions, by keeping defined tables and pointers in RAM in well known (ie. defined) locations.

Because of paging considerations, these are always kept in the bottom 16K segment (page 0), but this is kept to a minimum, and pointers to the actual code of the extensions can contain a segment number as well as an address. This can either point to code still within page 0 (in the case of extensions loaded from cassette or disk) or to another segment (in the case of extensions in ROM or extensions loaded as an EXOS system extension).

Any extension function or statement may corrupt any of the Z80 registers except IX, which always points to the start of the variables.

Many of BASIC's variables are useful or necessary for some extensions, and these are refered to mnemonically throughout this document. The actual addresses of these are given in section 5 (Variables).

BASIC also supports many system calls which may be used by extensions, and these are also refered to mnemonically, but are described in detail in section 6 (System Calls). These system calls allow extensions to be compatible with later or earlier versions of BASIC.

Section 2 (Data Structures) describes the format of BASICs internal data structures, so that extensions can be written to use these. An extension function, for example, may need to know the format of BASICs decimal numbers. Much of the need to access BASICs data structures directly has, however, been removed since many of the system calls will perform the required operation.

An example of an extension statement and an extension function are given in section 7 (Examples).

Extensions may take the form of code in an extension ROM or code on cassette / disk.

In the case of an extension ROM, the ROM must respond to the string "BASICX" which BASIC passes around at initialisation time. At this time, the ROM should copy all the necessary code into page 0 at the address pointed to by EXTOP, do any initialisation required, and update variables EXTOP, CURLOC and MCTOP to point to the next free byte in page 0. Only page 0 will be available to BASIC at this time, so the ROM should not attempt to address into the user's program, symbol table or stack, as they do not yet exist. Similarly, none of BASICs default channels will have yet been created. The system symbol table and keyword table, however, have been initialised with the default entries, and these may be changed or added to.

Extensions loaded from cassette / disk as a relocatable module contain both the page 0 information and the code itself in one block, and when these are loaded BASIC automatically makes room for them in page 0. They are initialised by giving the offset to the initialisation routine in the file header (see appropriate EXOS document), an offset of FFFFH indicating no initialisation routine (unlikely for a BASIC extension).

Extensions loaded from cassette / disk as an EXOS system extension must not adjust the variables mentioned above, and furthermore cannot allocate themselves more space in page zero. This limitation may be overcome by loading the page 0 information as a relocatable module followed immediately by an EXOS system extension module containing the actual code and data.

