Squash

Introduction and Overview

This module provides general compression and decompression facilities of a lossless nature through a SWI interface. The algorithm is 12-bit LZW, however, this may change in future releases.

The interface is designed to be restartable, so that compression or decompression can occur from a variety of locations. Operations involving file I/O can easily be constructed from the operations provided.

This module is not available in RISC OS 2.

The module is used by the Squash application to generate files of type Squash (&FCA). The format of these files is documented in the chapter entitled *FILEFORMATS.HTML#27821*.

Errors

The following errors can be returned by the Squash module:

Error number	Error text
&921	Bad address for module Squash
&922	Bad input for module Squash
&923	Bad workspace for module Squash
&924	Bad parameters for module Squash

SWI calls

Squash_Compress (SWI &42700)

Provides general compression of a lossless nature

On entry

R0 = flags:

Bit(s)	Meaning
0	Start new operation; otherwise continue existing operation (using existing workspace contents)
1	End of the input; otherwise more input exists after this
2	Reserved, must be zero
3	Return the work space size required and the maximum output size in bytes (all other bits must be 0); otherwise perform compression
4-31	Reserved, must be zero

R1 = input size (-1 to not return maximum output size) - if bit 3 of R0 is set; or workspace pointer - if bit 3 of R0 is clear
R2 = input pointer, word aligned - if bit 3 of R0 is clear
R3 = number of bytes of input available - if bit 3 of R0 is clear
R4 = output pointer, word aligned - if bit 3 of R0 is clear
R5 = number of bytes of output space available - if bit 3 of R0 is clear

On exit

R0 = required work space size - if bit 3 of R0 set on input; else output status - if bit 3 of R0 clear on input:

Value	Meaning
0	Operation completed
1	Operation ran out of input data (R3 = 0)
2	Operation ran out of output space (R5 < 12)

R1 = Maximum output size (-1 if we don't know or wasn't asked) - if bit 3 of R0 set on input; else preserved - if bit 3 of R0 clear on input
R2 = Updated to show first unused input byte - if bit 3 of R0 clear on input
R3 = Updated to show number of input bytes not used - if bit 3 of R0 clear on input
R4 = Updated to show first unused output byte - if bit 3 of R0 clear on input
R5 = Updated to show number of output bytes not used - if bit 3 of R0 clear on input

Interrupts

Interrupts are undefined
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This call provides general compression of a lossless nature. It acts as a filter on a stream of data. The call returns if either the input or the output is exhausted.

It is recommended that you use the following facility to determine the maximum output size rather than attempting to calculate it yourself:

Call the SWI first with bit 3 of R0 set and the input size placed in R1. The maximum output size is then calculated and returned on exit in R1. You can use this value to allocate the required amount of space and call the SWI again setting the registers as appropriate.

The algorithm used by this module is 12-bit LZW, as used by the UNIX

'compress' command (with -b 12 specified). If future versions of the module use different algorithms, they will still be able to decompress existing compressed data.

If bits 0 and 1 of R0 are clear, and the output is definitely big enough, a fast algorithm will be used.

The performance of compression on an 8Mhz A420 with ARM2 is approximately as follows:

Operation	Speed
Store to store	24 Kbytes per second
Fast case	68 Kbytes per second

where 'Fast case' is store to store, with all input present, and with an output buffer large enough to hold all output.

Related SWIs

SWI Squash_Decompress (on page 5)

Squash_Decompress (SWI &42701)

Provides general decompression of a lossless nature

On entry

R0 = flags:

Bit(s)	Meaning
0	Start new operation; otherwise continue existing operation (using existing workspace contents)
1	End of the input; otherwise more input exists after this
2	You may assume that the output will all fit in this buffer (allows a faster algorithm to be used, if bits 0 and 1 are both 0)
3	Return the work space size required and the maximum output size in bytes (all other bits must be 0); otherwise perform compression
4-31	Reserved, must be zero

R1 = input size (-1 to do not return maximum output size) - if bit 3 of R0 is set; or workspace pointer - if bit 3 of R0 is clear
R2 = input pointer - if bit 3 of R0 is clear
R3 = number of bytes of input available - if bit 3 of R0 is clear
R4 = output pointer - if bit 3 of R0 is clear
R5 = number of bytes of output space available - if bit 3 of R0 is clear

On exit

R0 = required work space size - if bit 3 of R0 set on input; else output status - if bit 3 of R0 clear on input:

Value	Meaning
0	Operation completed
1	Operation ran out of input data (R3 < 12)
2	Operation ran out of output space (R5 = 0)

R1 = Maximum output size (-1 if we don't know or wasn't asked) - if bit 3 of R0 set on input; else preserved - if bit 3 of R0 clear on input
R2 = Updated to show first unused input byte - if bit 3 of R0 clear on input
R3 = Updated to show number of input bytes not used - if bit 3 of R0 clear on input
R4 = Updated to show first unused output byte - if bit 3 of R0 clear on input
R5 = Updated to show number of output bytes not used - if bit 3 of R0 clear on input

Interrupts

Interrupts are undefined
Fast interrupts are enabled

Processor mode

Processor is in SVC mode

Re-entrancy

SWI is re-entrant

Use

This SWI provides general decompression of a lossless nature.

Note: The current algorithm cannot predict what the size of the decompressed output will be. This means that, currently, -1 is always returned on exit in R1. In future releases this may change; it is therefore recommended that you call the SWI first with bit 3 of R0 set and the input size placed in R1.

In the case where R3 < 12, the unused input must be resupplied.

The performance of decompression on an 8Mhz A420 with ARM2 is approximately as follows:

Operation	Speed
Store to store	48 Kbytes per second
Fast case	280 Kbytes per second

where 'Fast case' is store to store, with all input present, and with an output buffer large enough to hold all output.

Related SWIs

SWI Squash_Compress (on page 2)

Document information

Maintainer(s): RISCOS Ltd <developer@riscos.com>

History:

Revision	Date	Author	Changes
1		ROL	Initial version