Contents

File: /home/romain/Perso/CPC/current_projects/demo.bnd5/linking/s

Summary

Labels Summary

- .search_first_letter

Details

Labels

.search_first_letter Search the very first letter of the searched string

---

File: /home/romain/Perso/CPC/current_projects/demo.bnd5/linking/s

Summary

Details

File: /home/romain/Perso/CPC/current_projects/demo.bnd5/linking/s

Summary

Details

File: /home/romain/Perso/CPC/current_projects/demo.bnd5/linking/s

Summary

Details

File: /home/romain/Perso/CPC/current_projects/demo.bnd5/linking/s

Summary

Macros Summary

- MACRO DS_PRODUCE_KEYSPACE_ROUTINE()

- MACRO DS_KEYSPACE()
- MACRO DS_PRODUCE_STOP_SOUND_ROUTINE()
- MACRO DS_WAIT_MICROSEG(duration)
- MACRO LONG_WAIT_CYCLES(cycles)
- MACRO DS_WRITE_CRTC(register,value)
- MACRO DS_CRTC_HORIZONTAL_TRANSITION_ONLY(from,to)
- MACRO DS_CRTC_VERTICAL_TRANSITION_ONLY(from,to)

Details

Macros

MACRO DS_PRODUCE_KEYSPACE_ROUTINE() Generate the routine to check if the user pressed space

MACRO DS_KEYSPACE() Generate the code to check if the user pressed space

MACRO DS_PRODUCE_STOP_SOUND_ROUTINE() Generate the routine to stop any sound. Direct access to AY is done, not to firmware

MACRO DS_WAIT_MICROSEG(duration) Wait N microseconds (max wait 1021 ms / length 4 - 7 bytes) Initial code from @Syx

MACRO LONG_WAIT_CYCLES(cycles) Macro able to wait more than 1024 nops

MACRO DS_WRITE_CRTC(register,value) WRITE_CRTC: Writes a value to a CRTC register Corrupts: BC XXX we need to keep this slow and long code for crunch and duration purpose

MACRO DS_CRTC_HORIZONTAL_TRANSITION_ONLY(from,to) Il doit être possible de déterminer ce qu'il se passe exactement et d'écrire un cas particulier de la macro si on veut.

Music MUST not play under interruption for timing reasons XXX no deeply tested yet

MACRO DS_CRTC_VERTICAL_TRANSITION_ONLY(from,to) Vertical CRTC transition from R7=from to R7=to Music MUST not play under interruption for timing reasons XXX no deeply tested yet

---

File: /home/romain/Perso/CPC/current_projects/demo.bnd5/linking/s

Summary

Details

File: /home/romain/Perso/CPC/current_projects/demo.bnd5/linking/s

Summary

Macros Summary

- MACRO MOVE_CRUNCHED_DATA(from,to)
- MACRO UNCRUNCH_PRELUDE_NOT_BACKWARD(from,to)
- MACRO UNCRUNCH_PRELUDE_BACKWARD(from,to)
- MACRO UNCRUNCH_STANDARD_BACKWARD(from,to)
- MACRO INDIRECT_UNCRUNCH_STANDARD_BACKWARD(moved_from,from,to)
- MACRO UNCRUNCH_STANDARD_NOT_BACKWARD(from,to)
- MACRO INDIRECT_UNCRUNCH_STANDARD_NOT_BACKWARD(moved_from,from,to)
- MACRO UNCRUNCH_LZ4(from,to)
- MACRO UNCRUNCH_UPKR(from,to)
- MACRO UNCRUNCH_SHRINKLER(from,to)
- MACRO INSTALL_EXOMIZER()
- MACRO INSTALL_APLIB()
- MACRO INSTALL_UPKR()
- MACRO INSTALL_LZ4()
- MACRO INSTALL_LZ48()
- MACRO INSTALL_LZ49()
- MACRO INSTALL_LZSA1()
- MACRO INSTALL_LZSA2()
- MACRO INSTALL_SHRINKLER()
- MACRO INSTALL_ZX0()
- MACRO INSTALL_ZX0_BACKWARD()
- MACRO INSTALL_ZX7()

- MACRO LOAD_N_CRUNCH(label,fname)
- MACRO UNCRUNCH(from,to)
- MACRO INDIRECT_UNCRUNCH(moved_from,from,to)
- MACRO INSTALL_UNCRUNCHER()

Details

Macros

MACRO MOVE_CRUNCHED_DATA(from,to) Sometimes there are conflicts between the crunched data areas and the uncrunched data areas. Mainly because of banks overlap or lack of free memory. To avoid this, we first move the crunched data to a safe area, then uncrunch from there.

MACRO UNCRUNCH_PRELUDE_NOT_BACKWARD(from,to) Add some tests to check that the selected data is valid for non backward crunchers

Parameters:

from: start address of the crunched data (the very first byte) to: address where the uncrunched data will be stored (the very first byte)

MACRO UNCRUNCH_PRELUDE_BACKWARD(from,to) Add some tests to check that the selected data is valid for backward crunchers

Parameters:

from: start address of the crunched data (the very first byte AND NOT the very last byte) to: address where the uncrunched data will be stored (the very first byte AND NOT the very last byte)

MACRO UNCRUNCH_STANDARD_BACKWARD(from,to) Request to uncrunch data from {from} to {to} with the selected cruncher. Add some tests to check that the selected data is valid. Parameters:

- {from}: start address of the crunched data (the very first byte)
- {to}: address where the uncrunched data will be stored (the very first byte)

MACRO INDIRECT_UNCRUNCH_STANDARD_BACKWARD(moved_from,from,to) Indirect uncrunch is used when the crunched data has been move elsewhere. As such the from labels are not correct anymore.

Parameters: - {moved_from}: address where the crunched data has been moved - {from}: start address of the crunched data (the very first byte) BEFORE it has been moved - {to}: address where the uncrunched data will be stored (the very first byte)

MACRO UNCRUNCH_STANDARD_NOT_BACKWARD(from,to)
Request to uncrunch data from {from} to {to} with the selected cruncher. Add some tests to check that the selected data is valid. Parameters:

- {from}: start address of the crunched data (the very first byte)
- {to}: address where the uncrunched data will be stored (the very first byte)

MACRO INDIRECT_UNCRUNCH_STANDARD_NOT_BACKWARD(moved_from,from,to)
Indirect uncrunch is used when the crunched data has been moved elsewhere. As such the from labels are not correct anymore.

Parameters: - {moved_from}: address where the crunched data has been moved - {from}: start address of the crunched data (the very first byte) BEFORE it has been moved - {to}: address where the uncrunched data will be stored (the very first byte)

MACRO UNCRUNCH_LZ4(from,to) Request to uncrunch data from {from} to {to} with LZ4 cruncher. Add some tests to check that the selected data is valid. Parameters:

- {from}: start address of the crunched data (the very first byte)
- {to}: address where the uncrunched data will be stored (The very first byte)v

MACRO UNCRUNCH_UPKR(from,to) Request to uncrunch data from {from} to {to} with UPKR cruncher. Add some tests to check that the selected data is valid. Parameters:

- {from}: start address of the crunched data (the very first byte)
- {to}: address where the uncrunched data will be stored (The very first byte)

MACRO UNCRUNCH_SHRINKLER(from,to) Request to uncrunch data from {from} to {to} with SHRINKLER cruncher. Add some tests to check that the selected data is valid. Parameters:

- {from}: start address of the crunched data (the very first byte)
- {to}: address where the uncrunched data will be stored (The very first byte)

MACRO INSTALL_EXOMIZER() Install the uncruncher for exomizer

MACRO INSTALL_APLIB() Install the uncruncher for aplib

MACRO INSTALL_UPKR() Install the uncruncher for upkr

MACRO INSTALL_LZ4() Install the uncruncher for lz4

MACRO INSTALL_LZ48() Install the uncruncher for LZ48

MACRO INSTALL_LZ49() Install the uncruncher for LZ49

MACRO INSTALL_LZSA1() Install the uncruncher for LZSA1

MACRO INSTALL_LZSA2() Install the uncruncher for LZSA2

MACRO INSTALL_SHRINKLER() Install the uncruncher for shrinkler

MACRO INSTALL_ZX0() Install the uncruncher for zx0

MACRO INSTALL_ZX0_BACKWARD() Install the uncruncher for zx0 backward

MACRO INSTALL_ZX7() Install the uncruncher for zx7

MACRO LOAD_N_CRUNCH(label,fname) Load a data file and crunch it with the selected cruncher. The file is crunched with the selected cruncher at the label `label` Sub labels .start is synonyme, .next is the byte after last crunched byte, .length is the crunched size

MACRO UNCRUNCH(from,to) Uncrunch data from {from} to {to} with the selected cruncher Parameters: - from: start address of the crunched data (the very first byte). It is better to use a label (with its sublabels) than an address - to: start address where the uncrunched data will be stored (the very first byte)

MACRO INDIRECT_UNCRUNCH(moved_from,from,to) Indirect uncrunch data from {from} to {to} with the selected cruncher Parameters: - moved_from: address where the crunched data has been moved - from: start address of the crunched data (the very first byte) was initially before being moved. It is better to use a label (with its sublabels) than an address - to: start address where the uncrunched data will be stored (the very first byte)

TODO: implement all the case. ATM I have only implemented what I used

MACRO INSTALL_UNCRUNCHER() Install the uncruncher for the selected cruncher

---

File: /home/romain/Perso/CPC/current_projects/demo.bnd5/linking/s

Summary

Details

File: /home/romain/Perso/CPC/current_projects/demo.bnd5/linking/s

Summary

Macros Summary

- MACRO DS_INITIALIZATION()
- MACRO DS_SELECT_A_MEMORY_SPACE(space)
- MACRO DS_LOAD_FILE_IN_EXTRA_PAGE(filename,filename_size,address,loading_options)
- MACRO **DS_LOAD_FILE_IN_EXTRA_PAGE_INNER**(filename,filename_size,address,restore_
- MACRO DS_LOAD_FILE_IN_MAIN_PAGE(filename_addr,filename_size,load_address,loading_optio
- MACRO DS_LOAD_AND_RUN_FILE_IN_MAIN_PAGE(filename,filename_size,load_address,exec_a
- MACRO DS_MAKE_FIRMWARE_QUIET()
- MACRO DS_COPY_MEMORY_FROM_EXTRA_PAGE_TO_MAIN_PAGE(from,to,page,size)
- MACRO DS_BACKUP_FIRMWARE()

Equs Summary

- DS_FIRST_AVAILABLE_ADDRESS EQU &300

Details

Macros

MACRO DS_INITIALIZATION() This macro should be called at the very beginning of the demo. it aims at checking the kind of device use to load

MACRO DS_SELECT_A_MEMORY_SPACE(space) Select a memory space for the demo Completly relocatable code

MACRO DS_LOAD_FILE_IN_EXTRA_PAGE(filename,filename_size,address,loading_option
Request the demosystem to load a file. This macro must be called when being in space &c0 The process to load a file is the following: - memory conf. in &C1 - restore the system - memory conf in &C2 - load the file - backup the system - memory conf in C1 - restore the normal stack - memory conf in &C0

In case of failure the program is stucked In case of success: - page 1 contains the loaded file - bc contains the length of the file - de contains the expected destination of the file

Input: - filename: address to the filename - address: loading address IN THE EXTRA page

MACRO DS_LOAD_FILE_IN_EXTRA_PAGE_INNER(filename,filename_size,address,restor
Private macro. Do not use directly Load a file and optionnaly restore &C0

MACRO DS_LOAD_FILE_IN_MAIN_PAGE(filename_addr,filename_size,load_address,loadi
Load first the file in extra page. Once done, move it, in the main memory. This is a slow process as each byte has to pass over a buffer TODO handle a fast copy for bytes NOT in &4000-&7fff space as they can skip the buffer TODO handle case where the memory is overriden by the loaded file

MACRO DS_LOAD_AND_RUN_FILE_IN_MAIN_PAGE(filename,filename_size,load_addre
Load first the file in extra page. Once done, move it, in the main memory. This is a slow process as each byte has to pass over a buffer. Then launch it TODO handle a fast copy for bytes NOT in &4000-&7fff space as they can skip the buffer TODO handle case where the memory is overriden by the loaded file

MACRO DS_MAKE_FIRMWARE_QUIET() Ensure the demosystem will not write anywhere It is probably mandatory using C1 as any print will destroy the demosystem

MACRO DS_COPY_MEMORY_FROM_EXTRA_PAGE_TO_MAIN_PAGE(from,to,page,siz
Copy something loaded in the extra memory into the main memory

MACRO DS_BACKUP_FIRMWARE() Backup the firmware. Has to be called one time at the beginning of the demo.

Equs

DS_FIRST_AVAILABLE_ADDRESS EQU &300 The minmum address available to load a file in the main memory.