The NOV CHAP is an HP-41 companion module image to the physical NoV modules produced by Diego Diaz - see http://www.clonix41.org. It greatly enhances the value of the NoV modules by adding several important features. NOV CHAP is created by Geir Isene and Ángel Martin. It is a continuation of Geir Isene's ICEBOX project which in turn was a fork of Ángel Martin's Toolbox containing lots of utilities compiled from many sources. The NOVCHAP is released under the GNU General Public License, version 3.

The NoV family of modules consists of the NoVRAM, The NoV-32 and the NoV-64.

The NoVRAM gives the user four 4K pages of HEPAX RAM (pages #8 - #B) and four 4K pages of burnable EPROM (pages #C - #F). The RAM pages are non-volatile and retain its content even when the module is removed from the calculator.

The NoV-32 adds a second block of four 4K HEPAX RAM pages (32K RAM total). Whether you use the first block of 4*4K RAM or the second block is determined by the word contained at address 4100. If the word is hXX0, the first block is selected, while the word hXX1 selects the second block (the X's can be any value).

With the NoV-64, you get four blocks of 4*4K HEPAX RAM (64K RAM total) and an additional block of 4*4K of EPROM. The address containing the configuration word is still 4100, but now you can configure the module many ways. Please refer to the manual for details. Here it suffices to say that the first value in the word selects the ROM block to be used (pages #C- #F) with the value of 1 indicating the first ROM block and the value of 2 indicates the second ROM block. The last value of the word indicates which HEPAX RAM block to fill into pages #8 - #B (0-3). Entering h102 into the address 4100 tells the NoV-64 to use the first ROM block in pages #C - #F and the third HEPAX ram block in pages #8 - #B. If the first value is zero, no ROM block is selected for pages #C - #F. It is possible to use two RAM blocks and no ROM block. If you want such a configuration, then the first value must be zero and the last two values will indicate which RAM blocks to be used (largest value first); h031 would configure the fourth RAM block for pages #C - #F while the second RAM block would fill up pages #8 - #B.

The functions of the NOVCHAP is listed in XROM number sequence and explained with both function input and output. If you find anything unclear, or if you have suggestions for improvement – please contact Geir Isene (g@isene.com).

There are several FOCAL programs in this module (marked in red font). These programs are disguised as MCODE programs and can only be copied to RAM by setting the pointer within the program (such as GTO “FLSORT” and then do a “silent copying” with COPY ALPHA ALPHA. However this may result in an unstable RAM, so do this at your own risk.

The functions come in four categories;

- NoV specific functions
- Advanced Hepax functions
- Advanced Extended Memory functions
- Utility functions

They are marked by separate headers in the module and different colors in this user manual.

The project home page is located here: http://isene.com/isene.cgi?hp-41

A special thanks goes to all those who helped bring life to the magnificent HP-41 and to those who continue to keep it alive with the creation of new programs and modules.

<table>
<tr><th>XROM</th><th>NAME</th><th>SHORT DESCRIPTION</th><th>SOURCE</th></tr>
<tr><td><b>16,00</b></td><td><b>-NOV CHAP</b></td><td><b>Header/Alpha Backspace</b></td><td><b>W&W GmbH</b></td></tr>
<tr><td><b>Input:</b></td><td colspan="3">Characters in Alpha</td></tr>
<tr><td><b>Output:</b></td><td colspan="3">Alpha with the last character removed</td></tr>
<tr><td colspan="4">Main module header. This header also serves as a Alpha Backspace. To use this function, you will have to run it via the XROM function (see below), with “16,00” entered into its prompt.</td></tr>
<tr><td><b>16,01</b></td><td><b>GETN</b></td><td><b>Restore main memory</b></td><td><b>Geir Isene</b></td></tr>
<tr><td><b>Input:</b></td><td colspan="3">N/A</td></tr>
<tr><td><b>Output:</b></td><td colspan="3">N/A (“NONEXISTENT” if HEPAX data file “N” is not present)</td></tr>
<tr><td colspan="4">Restores main memory from a file named “N” in HEPAX ram (must be created manually or by the function SAVEN first). This function calls HGETA with the parameter “N” in Alpha.</td></tr>
<tr><td><b>16,02</b></td><td><b>GETNOV</b></td><td><b>Restore all HEPAX RAM blocks</b></td><td><b>Geir Isene</b></td></tr>
<tr><td><b>Input:</b></td><td colspan="3">N/A</td></tr>
<tr><td><b>Output:</b></td><td colspan="3">N/A</td></tr>
<tr><td colspan="4">This is a FOCAL program restoring all HEPAX RAM blocks from the currently selected mass storage media on an HP-IL loop.</td></tr>
<tr><td><b>16,03</b></td><td><b>N100</b></td><td><b>Write h100 to addr. 4100</b></td><td><b>Geir Isene</b></td></tr>
<tr><td><b>Input:</b></td><td colspan="3">N/A</td></tr>
<tr><td><b>Output:</b></td><td colspan="3">N/A</td></tr>
<tr><td colspan="4">Write the hex value of “100” into address 4100. This is only useful if you have a NoV-32 or a NoV-64 module. For the NoV-32, this function will activate HEPAX RAM bank #0. For the NoV-64, this function will activate ROM Bank #1 and HEPAX RAM bank #0.</td></tr>
<tr><td><b>16,04</b></td><td><b>N101</b></td><td><b>Write h101 to addr. 4100</b></td><td><b>Geir Isene</b></td></tr>
<tr><td><b>Input:</b></td><td colspan="3">N/A</td></tr>
<tr><td><b>Output:</b></td><td colspan="3">N/A</td></tr>
<tr><td colspan="4">Write the hex value of “100” into address 4100. This is only useful if you have a NoV-32 or a NoV-64 module. For the NoV-32, this function will activate HEPAX RAM bank #1. For the NoV-64, this function will activate ROM Bank #1 and HEPAX RAM bank #1.</td></tr>
<tr><td><b>16,05</b></td><td><b>N102</b></td><td><b>Write h102 to addr. 4100</b></td><td><b>Geir Isene</b></td></tr>
<tr><td><b>Input:</b></td><td colspan="3">N/A</td></tr>
<tr><td><b>Output:</b></td><td colspan="3">N/A</td></tr>
<tr><td colspan="4">Write the hex value of “102” into address 4100. This is only useful if you have a NoV-64 module. This function will activate ROM Bank #1 and HEPAX RAM bank #2.</td></tr>
<tr><td><b>16,06</b></td><td><b>N103</b></td><td><b>Write h103 to addr. 4100</b></td><td><b>Geir Isene</b></td></tr>
<tr><td><b>Input:</b></td><td colspan="3">N/A</td></tr>
<tr><td><b>Output:</b></td><td colspan="3">N/A</td></tr>
<tr><td colspan="4">Write the hex value of “103” into address 4100. This is only useful if you have a NoV-64 module. This function will activate ROM Bank #1 and HEPAX RAM bank #3.</td></tr>
<tr><td><b>16,07</b></td><td><b>N200</b></td><td><b>Write h200 to addr. 4100</b></td><td><b>Geir Isene</b></td></tr>
<tr><td><b>Input:</b></td><td colspan="3">N/A</td></tr>
<tr><td><b>Output:</b></td><td colspan="3">N/A</td></tr>
<tr><td colspan="4">Write the hex value of “200” into address 4100. This is only useful if you have a NoV-64 module. This function will activate ROM Bank #2 and HEPAX RAM bank #0.</td></tr>
</table>

16,08 N201 Write h201 to addr. 4100 Geir Isene

Input: N/A

Output: N/A

Write the hex value of “201” into address 4100. This is only useful if you have a NoV-64 module. This function will activate ROM Bank #2 and HEPAX RAM bank #1.

16,09 N202 Write h202 to addr. 4100 Geir Isene

Input: N/A

Output: N/A

Write the hex value of “202” into address 4100. This is only useful if you have a NoV-64 module. This function will activate ROM Bank #2 and HEPAX RAM bank #2.

16,10 N203 Write h203 to addr. 4100 Geir Isene

Input: N/A

Output: N/A

Write the hex value of “203” into address 4100. This is only useful if you have a NoV-64 module. This function will activate ROM Bank #2 and HEPAX RAM bank #3.

16,11 N? Write addr. In 4100 to X Geir Isene

Input: N/A

Output: Number (100, 101, 102, 103, 200, 201, 202 or 203) in X

Write the value of address 4100 to X. See N100 for more information.

16,12 NBS NoV Block Switch Geir Isene

Input: NS prompts for the NoV bank number (100-103,200-203 – see N100)

Output: N/A (“DATA ERROR” if input value is not in the ranges 000-003, 100-103,200-203. “NON EXISTENCE” if HGETA is not found, “CALC OFF” if ROM block is switched)

This function switches the block to the configuration you enter at the prompt and then restores Main Memory to the file named “N” in the new HEPAX RAM block. If you switch the ROM block (the first of the three digits you enter at the prompt is different than the current value at the address 4100), it gives a brief message, “CALC OFF” to remind you that you must turn the calc off and on again for the ROM block switch to take effect.

16,13 NX Write X to addr. 4100 Geir Isene

Input: Number (000, 001, 002, 003, 100, 101, 102, 103, 200, 201, 202 or 203) in X

Output: N/A

Write the value in X into address 4100. See N100 for more information.

16,14 SAVEN Save main memory Geir Isene

Input: N/A

Output: N/A (“NONEXISTENT” if HEPAX data file “N” is not present)

Saves main memory from a file named “N” in HEPAX ram. This function calls HSAVEA with the parameter “N” in Alpha.

16,15 SAVENOV Restore all HEPAX RAM blocks Geir Isene

Input: N/A

Output: N/A

This is a FOCAL program saving all HEPAX RAM blocks from the currently selected mass storage media on an HP-IL loop.

16,16 -ADV HEPAX Header

Input: N/A

Output: N/A

No function, only a header

16,17 ?JUMP Calculate MCODE jump code VM Electronics

Input: N/A

Output: MCODE jump code

This is the program “JUMP” from the HEPAX manual (page 128). The program prompts for the jump type: “TYPE (0-3)” where 0 = ?NC XQ, 1 = ?C XQ, 2 = ?NC GO, 3 = ?C GO. It then prompts for the target address and returns the MCODE codes for the jump.

16,18 DISSST Disassemble MCODE programs VM Electronics

Input: N/A

Output: MCODE disassembled one line at a time

This is the program “DISSST” from the HEPAX manual (page 67). The program prompts for the start and end address to be disassembled. It then shows the MCODE program disassembled, one line at a time, just keep pressing R/S until the end address.

16,19 GTOADDR Goto address (and run) Geir Isene

Input: ROM address in NNN in X

Output: N/A

Jumps directly to the ROM address given in X, in NNN (Non-Normalized Number).
DO NOT USE THIS unless you know what you are doing.
This is a dangerously powerful function.

16,20 HFIX Fix HEPAX RAM page Geir Isene

Input: Next HEPAX RAM page in Y, previous HEPAX RAM page in X

Output: N/A

This function puts a HEPAX RAM page back into the HEPAX chain (if it was intentionally removed such as with HKILL or the chain was unintentionally broken). If you have HEPAX ram in pages #8, #9, #A and #B and #A has been removed from the chain, put 11 (#B) into Y and 9 into X and XEQ”HFIX”. You will be prompted for the page to repair (enter 10). The chain is now fixed. If the page to repair is #8, put 9 into Y and 0 into X (start of chain). If the page to repair is #B, put 0 into Y (end of chain) and 10 into X.

16,21 HGETAS HEPAX ASCII file to XM Geir Isene

Input: File name in Alpha

Output: N/A

A focal program that retrieves an ASCII file from HEPAX RAM to Extended Memory.

16,22 HGETD HEPAX data file to XM Geir Isene

Input: File name in Alpha

Output: N/A

A focal program that retrieves an data file from HEPAX RAM to Extended Memory.

16,23 HKILL Remove HEPAX ram page Geir Isene

Input: Prompting function, prompts for HEPAX RAM page to remove from HEPAX chain

Output: N/A

Takes out a HEPAX RAM page from the HEPAX chain.

16,24 HRESZFL Resize a HEPAX ASCII file Geir Isene

Input: File name in Alpha, target file size in X

Output: N/A

Resizes the file named in Alpha to the size in X. HRESZFL uses no Extended Functions, while HRSZFL2 uses Extended functions and is shorter and faster.

16,25 HRSZFL2 Resize a HEPAX ASCII file Geir Isene

Input: File name in Alpha, target file size in X

Output: N/A

Resizes the file named in Alpha to the size in X. HRSZFL2 uses Extended Functions and is shorter and faster than its counterpart HRESZFL.

16,26 HSAVEAS XM ASCII file to HEPAX RAM Geir Isene

Input: File name in Alpha

Output: N/A

A focal program that saves an ASCII file from Extended Memory to HEPAX RAM.

16,27 HSAVED XM ASCII file to HEPAX RAM Geir Isene

Input: File name in Alpha

Output: N/A

A focal program that saves a data file from Extended Memory to HEPAX RAM.

16,28 READWRD Read MCODE WORD Geir Isene

Input: ROM address in X (NNN)

Output: ROM address word in Y (NNN)

Takes an address in X (in NNN format - use HEX>NNN to take an address in ALPHA and convert it to NNN format in X) and returns the NNN value from that address in Y (use NNN>HEX to get the hex value in ALPHA).

The address in the X register is incremented by one (makes it easy to view the ROM instruction-by-instruction).

16,29 SCHWRD Search for MCODE WORD Geir Isene

Input: ROM address (NNN) in X, WORD to search for in Y (NNN)

Output: ROM address where WORD is found in Y (or end of page)

Executing SCHWRD will start the search immediately after the address you entered into X. It will return the address into X where it finds the first occurrence of the search word. You can then execute SCHWRD again to find the next occurrence etc.

SCHWRD will stop when it reaches the end of the block and return the start address of the next block (by again executing SCHWRD, it will continue into the next block).

16,30 WRITWRD Write MCODE WORD Geir Isene

Input: ROM address (NNN) in X, WORD to write in Y (NNN)

Output: N/A

Takes an address in X and the value to write to that address in Y (both in NNN). This can only be used to write to EPROM RAM.

16,31	XMBACKUP	Backup XM files to HEPAX RAM	

Input: N/A

Output: N/A

There is no native HEPAX function to back up whole or parts of Extended Memory. This FOCAL program makes such a backup possible. XMBACKUP uses two XM ASCII files as “control files” - “BKAS” for XM ASCII files to be backed up to HEPAX RAM and “BKD” for XM data files to be backed up to HEPAX RAM. These control files can be manually edited to include the ASCII or data files you want to have backed up to HEPAX RAM, or you can let the program collect all ASCII and data files in XM and include them as entries into the “BKAS” and “BKD” files respectively.

The program presents a menu corresponding to program labels A,a B,b C: “**B,R AS,D COL**”:

LBL A: “**B**” = Backup the XM files listed in the control files “BKAS” and “BKD” to HEPAX RAM.

LBL a: “**R**” = Restore the files listed in the control files “BKAS” and “BKD” from HEPAX to XM.

LBL B: “**AS**” = Edit “BKAS” to include or remove ASCII files to be backed up/restored.

LBL b: “**D**” = Edit “BKD” to include or remove data files to be backed up/restored.

LBL C: “**COL**” = Collect all XM ASCII files as entries into “BKAS” and data files into “BKD”

16,32	XMRESTR	Restore XM files from HEPAX	Geir Isene

Input: N/A

Output: N/A

A shortcut to “LBL a” in XMBACKUP for easy restore of selected files from HEPAX RAM to XM.

16,33	-ADV XM	Header/CLA to comma	W&W GmbH

Input: String with a comma in Alpha

Output: Clears Alpha up to comma

This header also serves as a “CLA to comma”.

16,34	ARCLCHR	Recall character from XM ASCII	Håkan Thörngren

Input: N/A

Output: Character in current XM ASCII file to Alpha

Copies next character in current XM ASCII file to Alpha.

16,35	FILE	XM ASCII file management	Geir Isene

Input: N/A

Output: N/A

FILE handles the contents of ASCII files. The program first shows the name of the current file and then the main menus corresponding to labels A-E and then shifted labels a-e:

“+. .+ +1 +X ED” and “**- .+a -1 S ><**”

LBL A: “**+.**” = Insert record (in Alpha) before current record

LBL B: “**.+**” = Append record (in Alpha) after current record

LBL C: “**+1**” = Jump one record forward

LBL D: “**+X**” = Jump the specified number of records (in X) forward (or back if X is negative)

LBL E: “**ED**” = Edit current file

LBL a: “**-**” = Delete current record

LBL b: “**.+a**” = Append Alpha to current record

LBL c: “**-1**” = Jump one record backward

LBL d: “**S**” = Sort file alphabetically

LBL e: “><” = Trim file (i.e. run FLSZ-)

16,36 FILEMAN File Management Geir Isene

Input: N/A

Output: N/A

The program first shows the name of the current file and then the main menus corresponding to labels A-E and then shifted labels a-e:

"**C:A D G:A D SK**" and "**P CL S:A D FL**"

LBL A: “**C:A**” = Create ASCII file (name in Alpha)

LBL B: “**D**” = Create Data file (name in Alpha)

LBL C: “**G:A**” = Get/retrieve ASCII file from HEPAX memory (name in Alpha)

LBL D: “**D**” = Get/retrieve Data file from HEPAX memory (name in Alpha)

LBL E: “**SK**” = Select file (name in Alpha) and set record to 0 (i.e. execute a SEEKPTA)

LBL a: “**P**” = Purge file (name in Alpha)

LBL b: “**CL**” = Clear file (name in Alpha)

LBL c: “**S:A**” = Save ASCII file to HEPAX memory (name in Alpha)

LBL d: “**D**” = Save Data file to HEPAX memory (name in Alpha)

LBL e: “**FL**” = Run the "FILE" program

16,37 FLHD File header Ángel Martin

Input: File name in Alpha, or blank for current file.

Output: Address of file header in X

FLHD returns the address where the file header is located in XMEM. Handy for peeking or poking around, for instance using PEEKR and POKER.

16,38 FLSZ+ Make room in XM ASCII file Geir Isene

Input: File name in Alpha

Output: N/A

Ensures the XM ASCII file has room for 4 more records.

16,39 FLSZ- Trim XM ASCII file Geir Isene

Input: File name in Alpha

Output: N/A

Trims the XM ASCII file to minimum size.

16,40 FLSORT Sort XM ASCII file Geir Isene

Input: File name in Alpha

Output: N/A

Alphabetically sorts an XM ASCII file.

16,41 FLTYPE Get XM File Type Ángel Martin

Input: File Name in Alpha, or blank for current file.

Output: File type in X

Returns the file type X:

0 = No current file selected

1 = Program file

2 = Data file

3 = ASCII file

4 = Matrix file (CCD, Advantage)

5 = Buffer file (CCD)

6 = XM Contents File (SKWID, PANAME)

16,42 GETBUF Reads Buffer from XM File Håkan Thörngren

Input: File Name in Alpha.

Output: Buffer is restored into buffer area.

Restores the buffer from the XM file. If a buffer with the same id# already exists it'll show "DUP BUF". The Buffer id# is stored into the file, therefore it's not needed as an input.

16,43 RENMFL Rename XM file Ángel Martin

Input: "OLDNAME,NEWNAME" in Alpha

Output: N/A

With Alpha containing the old filename and the new filename separated by a comma, RENMFL will rename the XM file (for any file type).

16,44 RESTCHK Restore checksum Håkan Thörngren

Input: XM program file name

Output: N/A

Restores the checksum byte of an XM program file (in case it has been broken).

16,45 RETPFL Change the file type of XM file Ángel Martin

Input: Filename in Alpha, file type in X

Output: N/A

Changes the file type of file to the type specified in X. See FLTYPE for file types.

04,46 SAVEBUF Saves Buffer in XM File Håkan Thörngren

Input: File Name in Alpha, Buffer id# in X

Output: Buffer saved as XM File, with File type = 5

On power ON, the calculator will zero out the buffers' first ID# and let the various ROMs reclaim them (by putting back the first ID#. If no ROM reclaims a particular buffer, it is lost.

See the David Assembler manual for more information.

16,47 SKPTACR SEEKPTA or create file Geir Isene

Input: File name in Alpha

Output: N/A

Sets the file pointer of XM ASCII file to 0, or if file does not exist, creates it with size = 10.

16,48 XMFILE? Get current XM file name Geir Isene

Input: N/A

Output: Current XM file name in Alpha

Retrieves the name of the current XM file.

16,49 -UTILS Header/GTO .END. Ken Emery

Input: N/A

Output: N/A

This header doubles as a “GTO .END.” - i.e. Sets the program pointer at .END.

16,50 A<>RG _ _ Exchange Alpha with Regs Ángel Martin

Input: N/A

Output: N/A

Exchanges Alphas MNOP with the 4 registers starting with the register prompted for.

16,51 ARCLIP _ _ Ángel Martin

Input: N/A (prompting function – prompts for register)

Output: Adds integer part of register content to Alpha

Function prompts for a register number (accepts IND but not ST) and adds the integer part of that register to the contents of Alpha.

16,52 BUF? Ángel Martin

Input: Buffer ID# in X

Output: Yes/No, skips program line if False.

A conditional test to see if a specific buffer ID# exists. Here is a list of possible buffer Ids:

Buffer id#	*Module/Eprom*	*Reason*
1	David Assembler	MCODE Labels already existing
2	David Assembler	MCODE Labels referred to
3	Eramco RSU-1B	ASCII file pointers
4	Eramco RSU-1A	Data File Pointers
5	CCD Module, Advantage	Seed, Word Size, Matrix Name
6	Extended IL (Skwid)	Accessory ID of current device
7	Extended IL (Skwid)	Print Cols, number & width
8	Complex Stack	Ángel Martin's 41Z ROM
10	Time Module	Alarms information
11	Plotter Module	Data and barcode parameters
12	IL Development, CMT-200	IL buffer and monitoring
13	CMT-300	Status Info
14	Advantage	INTEG & SOLVE scratch
15*	Mainframe	Key Assignments
(*) KA area isn’t really a buffer.		

16,53 FDATA _ Function Data Klaus Huppertz

Input: Function name in Alpha (prompt)

Output: FAT address and XROM value in Alpha

Shows the FAT address and XROM value (the one used for key assignments) of the function input into the function’s Alpha prompt. It works equally for mainframe functions, User Code programs in RAM, and MCODE functions in ROM.

Despite being an Alpha prompt function when invoked from the keyboard, FDATA is also programmable: when in a program, the function name will be taken from the Alpha register!

16,54 HEX>NNN Code Ken Emery

Input: HEX value in Alpha

Output: NNN in X

This ia an improved version of the well-known CODE functions. The function is well-known and has been around for a long time, included already in the PPC ROM (routine “HN”).

16,55 NNN>HEX Decode Clifford Stern

Input: NNN in X

Output: HEX value in Alpha

This is an improved version of the well-known DECODE functions. The function is well-known and has been around for a long time, included already in the PPC ROM (routine “NH”).

NNN>HEX will decode the NNN in X into the HEX code in Alpha, and (contrary to other implementations of this function) without leading zeros (i.e. no left-padding).

16,56 PEEKR Ken Emery

Input: Register number in X

Output: Register content in X

The content of register N (absolute register address given in X) is returned to X.

16,57 POKER Ángel Martin

Input: Content to be stored in X, Reg# in Y

Output: N/A

Puts content of X into the absolute register number given in Y.

16,58 RCLB Mark Power

Input: Reg# in M – ZENROM convention

Output: Recalls to X the contents of byte which address is in M/

Like PEEKB, but using the M register as input (ZENROM like)

16,59 ST<>RG _ _ Exchange Stack with Regs Ángel Martin

Input: N/A

Output: N/A

Exchanges Stack with the 4 registers starting with the register prompted for.

16,60 STOB Mark Power

Input: Reg# in M – ZENROM convention

Output: Stores the X contents in byte which address is in M

16,61 X<>B Mark Power

Input: Reg# in M – ZENROM convention

Output: Swaps contents of X and the byte which address is in M

16,62 X>$ VM Electronics

Input: Content in X

Output: Content in X with mantissa sign (nybble 13) changed to 1

Changes the mantissa sign (nibble 13) into a "1", indicating ALPHA.

16,63	XROM	XEQ ROM	Clifford Stern

Input: XROM number (prompt)

Output: N/A

A very special prompting function. Allows direct entry of any function included in a plug-in module, by introducing its XROM number first and then the function number.

This allows access to ROM header functions, such us "–NOV CHAP", (XROM 16,00) – doubles as an Aplha Backspace. Note that while XROM is not programmable, the function called can be entered into a program, thus it isn't necessary that the ROM be present to introduce its corresponding functions.

NOV CHAP home page: http://isene.com/isene.cgi?hp-41

Want other functions in the NOV CHAP?

Please e-mail me at g@isene.com and ask for functionality – your wish may come true :-)

Geir Isene

Oslo, 2011-04-04