MPL Maintenance

		Input	Output	Stack
MPINIT	Initialize the MPL. Creates two buffers RES and TMP with ID 33 and 44.	X - Number of registers per MPN	None	unaltered
MPUNL	frees up space by deleting the MPL buffers	none	None	unaltered
MPL?	Checks if the MPL is loaded. Skip/no if not, NoSkip/yes otherwise	none	None	unaltered
MPNOR?	Delivers to X the number of registers per MPN for the current MPL	none	X - #of reg per MPN	lifted
MPNOD?	Delivers to X the number of digits per MPN for the current MPL	none	X - # of dig per MPN	lifted

MPN Maintenance

		Input	Output	Stack
MPNEW	Creates a new MPN, setting its exponent, sign, first 10 digits and starting reg. Each MPN is identified by its starting reg for all MPL functions	Z - exponent and sign btw+/- 2048 Y - sign and up to first 10 digits. The decimal is always after the first digit. No exponent X - first reg for MPN.	none	unaltered
MPNEW0	Create a new MPN and set it to +0 exp+0	X - MPN(X)	none	unaltered
MPSTO	Sets 10 digits at a time for an MPN. User is in charge of picking the right register. Leading 0s will be automatically padded.	Y - register to store digits into X - 10 digits to store	none	unaltered
MPGET	Get first 10 digits and exponent of MPN	X - MPN (first reg)	Y - exponent X - first 10 digits	lifted
MPGETM	Get first 10 digits of MPN	X - MPN (first reg)	X - first 10 digits	lifted
MPGETX	Get exponent of MPN	X - MPN (first reg)	X - exponent	lifted
MPVIEW	MPN-Viewer. View MPN(X) in X, 10 digits at a time. R/S for next 10 digits, / for previous, <- to exit. 'RAD' signifies last reg	X - MPN(X)	none	unaltered

MP Arithmetic

		Input	Output	Stack
MP+	Add two MPN's from X and Y and place result into MPN in L. MPN(L) can be the same as MPN(X) or MPN(Y)	Y - MPN(Y) X - MPN1(X) L - MPN(L) - Result	X - MPN(L) Result L - MPN(X)	dropped
MP-	Subtract MPN(X) from MPN(Y) and place result into MPN in L. MPN(L) can be the same as MPN(Y) but not as MPN(X)	Y - MPN(Y) X - MPN1(X) L - MPN(L) - Result	X - MPN(L) Result L - MPN(X)	dropped
MP*	Multiply MPN(X) by MPN(Y) and store result in MPN(L). MPN(L) can be the same as MPN(X) or MPN(Y)	Y - MPN(Y) X - MPN1(X) L - MPN(L) - Result	X - MPN(L) Result L - MPN(X)	dropped
MP/	Divide MPN(Y) by MPN(X) and store result in MPN(L). MPN(L) can ***not*** be the same as either MPN(X) or MPN(Y)	Y - MPN(Y) X - MPN1(X) L - MPN(L) - Result	X - MPN(L) Result L - MPN(X)	dropped
MP1/X	Calculate 1/MPN(X) and store in MPN(L). MPN(L) can not be the same as MPN(X)	X - MPN(X) L - MPN(L) - Result	X - MPN(L) Result L - MPN(X)	dropped
MPX^2	Calculate the square of MPN(X) and store in MPN(L). MPN(L) can be the same as MPN(X)	X - MPN(X) L - MPN(L) - Result	X - MPN(L) Result L - MPN(X)	dropped
MPY^X	Exponentiate MPN(Y) by the normal number NN(X) and store result in MPN(L). MPN(L) can not be the same as MPN(X)	Y - MPN(Y) X - NN(X) L - MPN Result	X - MPN Result L - NN(X)	dropped

MPN Manipulation

		Input	Output	Stack
MPSEXP	Set the exponent of the existing MPN(Y). This can be used for fast multiplication or division by powers of 10	Y - MPN(Y) X - New exponent +/-2048	X - MPN(Y) L - New Exponent	dropped
MPCHS	Change the sign of MPN(X)	X - MPN(X)	none	unaltered
MPCLX	Clear the MPN(X) (i.e. +0.0 e0)	X - MPN(X)	none	unaltered
MPCPY	Copy existing MPN(Y) onto existing MPN(X), replacing its content	Y - Source MPN(Y) X - Target MPN(X)	X - Target MPN(X) L - Source MPN(X)	dropped

MPN - NN Interaction

		Input	Output	Stack
MPM2N	Convert the MPN(X) into a normal number. Exponent is modulo 100	X - MPN(X)	X - Normal Number	
MPN2M	Convert a normal number NN(Y) into the existing MPN(X)	Y - Normal Number NN(Y) X - Target MPN(X)	none	unaltered
MPM+N	Add the normal number NN(X) to the MPN(Y) and store the result in MPN(L). MPN(L) can be MPN(Y)	Y - MPN(Y) X - normal number NN(X) L - MPN(L) - Result	X - MPN(L) L - NN(X)	dropped
MPM-N	Subtract the normal number NN(X) to the MPN(Y) and store the result in MPN(L). MPN(L) can be MPN(Y)	Y - MPN(Y) X - normal number NN(X) L - MPN(L) - Result	X - MPN(L) L - NN(X)	dropped
MPM*N	Multiply the MPN(Y) by the normal number NN(X) and store the result in MPN(L). MPN(L) can be the same as MPN(Y)	Y - MPN(Y) X - normal number NN(X) L - MPN(L) - Result	X - MPN(L) L - NN(X)	dropped

Comparison Functions

		Input	Output	Stack
MPX=0?	Check if the MPN(X) is equal to 0 in both mantissa and exponent. Skip/No if not, otherwise No Skip/Yes	X - MPN(X)	none	unaltered
MPX=1?	Check if the MPN(X) is equal to 1 in the mantissa. The exponent is ignored. Skip/No if not, otherwise No Skip/Yes	X - MPN(X)	none	unaltered

System Functions - handle with care

		Input	Output	Stack
MPLCNR	Change the Number of Registers for the current MPL	none	none	unalterd
MPNCNR	Change the Number of Registers for the current MPN	none	none	unalterd

Error Msg	Functions	Explanation
MPL Level		
MPN SIZE=0	MPINIT	The desired size in registers for a MPN is invalid. It needs to be 1<MPN Size<100. Please enter a valid size and retry.
MPN SIZE>99	MPINIT	The desired size in registers for a MPN is invalid. It needs to be 1<MPN Size<100. Please enter a valid size and retry.
MPN SIZE<1	MPINIT	The desired size in registers for a MPN is invalid. It needs to be 1<MPN Size<100. Please enter a valid size and retry.
MPN SIZE<0	MPINIT	The desired size in registers for a MPN is invalid. It needs to be 1<MPN Size<100. Please enter a valid size and retry.
NO MPL	All MP functions	No MPL loaded. Please initialize the MPL with MPINIT and a proper MPN size
NO MPL(RES)	All MP functions	No MPL loaded. In particular the internal RES buffer was not found. Please initialize the MPL with MPINIT and a proper MPN size
NO MPL(TMP1)	All MP functions	No MPL loaded. In particular the internal TMP buffer was not found. Please initialize the MPL with MPINIT and a proper MPN size
MPN-MPL MISM	All MP functions	One or more of the selected MPNs use a different number of registers than the currently active MPL. Please unload (MPUNL) the current MPL and initialize a new one with the right number of registers per MPN
MPN Address - General		
INVALID MPN	All MP functions	One or more of the selected MPNs are invalid. This happens most often by inadvertendly using a register number which is not the first register of a MPN. Please check your MPN entries (including L) and retry
MPN Address Specific		
X-MPN INVALID	All MP functions	The number in x does not represent a valid MPN (e.g. negative, smaller than 0, etc)
X NO MPN	All MP functions	The number in X does not represent a MPN. Most likely it was an inintended input (e.g. switching X and Y regs)
MPN Memory Location		
MPN TOO LONG	MPNEW	There is not enough room to store all registers for this MPN. Please check the current SIZE setting, start at a lower register number and try again
X TOO LONG	All MP functions	There is not enough room to reach all registers for the MPN in X. Most likely you entered a wrong register number or changed the SIZE after enetering the MPN
Exponent Values		
EXP INVALID	MPNEW, MPSEXP	The desired exponent is invalid (e.g. a number <1 or >2048, not integer)
EXP >+/- 2048	MPNEW, MPSEXP	The desired exponent is > than +2048 or <-2048. Attention: Overflow/Underflow treatment not fully implemented yet!
MPN Values		
MPN-VAL>1e10	MPNEW, MPN2M, MPSTO	The desired 10 digits for the MPN are > than 1e10 (i.e. they have more than 10 digits or an exponent
MPN-VAL<0	MPNEW, MPN2M, MPSTO	The desired 10 digits for the MPN are < than 1 (i.e. they have a 0 as the only digit left of the comma)
Special		
FO IN C1stR	MP*	Odd nybble trying to load in first register Carry treatment in MP*. Please record your steps to reproduce this error and contact the author via the Forum

Goal	Steps	Notes
I - Initialize the MPL		
		Initialize the MPL for use with 3 registers per MPN or 25 digits.
Step 1	3, Enter, XEQ 'MPINIT'	If you get an Error, most likely you have a MPL loaded already. Unload it with MPLUNL (you might get an error that you can ignore however). Now redo Step 1
Step 2	XEQ 'MPL?'	Check that the MPL is loaded, You should see 'YES' in the display
Step 3	XEQ 'MPNOR?'	If you forget how many registers per MPN, use the 'MPNOR?' command to bring back to X the number of registers per MPN
Step 4	25 XEQ 'PSIZE'	make sure we have at least 25 registers available.
II - Create the first MPN-1		
		Create a new MPN starting in register 1 with a value of 1.1234 56789 01234 00998 87766 exp +123
Step 1	123, Enter, 11234, Enter, 1, XEQ 'MPNEW'	This creates a MPN starting in register 1, an exponent of +123 and the first 5 digits to be 1.1223
Step 2	2, Enter, 5678901234, XEQ 'MPSTO'	This enters the next 10 digits of the MPN
Step 3	3, Enter, 99887766, XEQ 'MPSTO'	This enters the last 10 digits of the MPN. Note how the leading 0s are not typed in. They will be automatically added by the MPL
III - Confirm that we have entered the MPN correctly		
Step 1	1 XEQ 'MPVIEW'	This will show us the first 5 digits in the LCD. The lit 0 annunciator signifies that we are looking at the first register. Confirm that it shows 1.1223
Step 2	press R/S	This will show the next 10 digits. Confirm that it shows '34455,66778'. If it does not simply repeat step II-2
Step 3	press R/S	This will show the last 10 digits. Confirm that it shows '00998,87766'. If it does not, simply repeat step II-3. Not that the 'Reached lAst Digit' annunciator is on to signify that we are watching the last register for this MPN
Step 4	press R/S or <-	This will end the view routine, leaving the last show digits in the display.You can simply delete them with <-, the X-register will not be affected.
Step 5	XEQ 'MPGETX'	make sure that you still have '1' in the X-register. If you inadvertendly deleted it, enter it again before XEQ 'MPGETX'. This will get you the exponent of this MPN into the X-register. Confirm that it is +123. You can always change an exponent via MPSEXP (see QRG or later on in this Step-by-Step guide
IV - Create two MPNs, MPN-4 and MPN-7 with a value of 0		
Step 1	4, XEQ 'MPNEW0'	This creates a MPN starting at register 4 and initializes it with a value of 0. Note how we make sure that we don't overlap MPN's - our first MPN starts at register 1 and goes until register 3
Step 2	XEQ 'MPX=0?'	Confirm that the MPN is really 0 via the MPX=0? Function. You should see 'YES' in the display. If not, please repeat steps III
Step 3	7, XEQ 'MPNEW0'	This creates a MPN starting at register 7 and initializes it with a value of 0. Note how we make sure that we don't overlap MPN's
Step 4	XEQ 'MPX=0?'	Confirm that the MPN is really 0 via the MPX=0? Function. You should see 'YES' in the display. If not, please repeat steps III
V - Square the MPN-1 and leave the result in MPN-4		
Step 1	4 STO L, 1, XEQ 'MPX^2'	MPN-4 is placed in the L-register and MPN-1 in the X-register. MPX^2 will square MPN-1 and place the result in MPN-4
Step 2	XEQ 'MPVIEW'	MPX^2 left the number for the result MPN (4 in this case) in the X-registers so that we can easily continue to work with the result. Here we just use MPVIEW to look at the result.
VIII - Divide MPN-4 by MPN-1, storing the result in MPN-7		
Step 1	7 STO L, 4 Enter, 1, XEQ 'MP/'	This divides MPN-4 by MPN-1 and stores the result in MPN-7. Remember that MPN-4 was calculated by squaring MPN-1. We will use this in the next step.
IX - Subtract MPN-1 from MPN-7, storing the result in MPN-4		
Step 1	4 STO L, 7 Enter, 1, XEQ 'MP-'	This will subtract MPN-1 from MPN-7 and store the result in MPN-4. See how we re-use MPN-4 to save memory. This is not dissimilar to how we constantly try to re use the stack as much as possible to avoide having to use extra storage registers.
X - Confirm that the result is 0		
Step 1	XEQ 'MPX=0?'	The previous 'MP-' leaves 4 in the X-register which we can use directly again to check that the result is 0 indeed, as it should be (MPN1^2 / MPN1 - MPN1 = 0)

This concludes this simple tutorium. You have learned how to create MPNs and do simple arithmetic with them. Get yourself familiar especially with the different ways of creating a MPN (MPNEW, MPNEW0, MPN2M) and the various arithmetic functions to manipulate them (MP+, MP-, MP*, MP/, MP1/X,MPX^2, MPY^X,MPCHS, MPCLX, MPCPY)