Reference Guide

JIMROM – MELBROM – PPC ROM 2 – TOULROM

Rev 1.0

Contents

Introduction

The purpose of this document is to regroup in a single place the descriptions provided in the different PPC chapters local revue issues (PPC TN, PPC-T, JPC) for the different functions found in the early HP-41 microcode ROMS developed by the PPC chapters users.

Namely:

- The JIMROM developed by Jim De Arras and provided with the HHP-16K 41C Rom Emulator
- The MELBROM and the PPC ROM 2 developed by the members of the PPC Melbourne chapter based on the JIMROM 1H
- The TOULROM developed by the PPC-Toulouse members based on the PPC ROM 2C

These four early ROMS are linked as they are derived as follows:

JIMROM ➔ MELBROM ➔ PPC ROM 2 ➔ TOULROM

The list of functions and XROM numbers are based on the functions list published in the PPC revues and/or content of the currently available images of these ROMs: JIMROM 1H, MELBROM 1A, PPC ROM 2C, TOULROM 1A, TOULROM 1B, TOULROM 1C, TOULROM 1D.

The TOULROM 1B is a special case as the currently available image includes a lot of functions that are not found in the other TOULROM versions, but which are shared for most of them with the MLDL OS 1A ROM. So, these functions are not documented here, only the functions that are listed for TOULROM 1B in PPC-T N2 are documented.

Function lists

JIMROM – MELBROM – PPC ROM 2

Rom file name		Melbourne	PPC-MELB
XROM		12	12
Version	1H	1A	2C
ROM Revision		PPC1	PR2C
Date	December 1981?	February 1982	May 1982
CAT name	JIMROM 1H	MELBROM 1A	PPC ROM 2C
	X<>ROM	X<>ROM	X<>ROM
	Y<>T	Y<>T	Y<>T
	Z<>T	Z<>T	Z<>T
	Y<>Z	Y<>Z	Y<>Z
	BCDBIN	BCDBIN	BCDBIN
	GTOROM	GTOROM	GTOROM
	RXR	RXR	RXR
	RXL	RXL	RXL
	X+Y	X+Y	X+Y
	SXL	SXL	SXL
	NSTO	NSTO	NSTO
	NRCL	NRCL	NRCL
	TP	**PSIZE**	TP
	VP	VP	VP
	CODE	CODE	CODE
	DECODE	DECODE	DECODE
	EN	EN	EN
	DE	DE	DE
	CLPV	CLPV	CLPV
	MEMORY LOST	MEMORY LOST	XROMTST
	UST KIDDING	UST KIDDING	**MELBROM 1A**
			PSIZE
		PPACK	PPACK
		DIS	DIS
		2-D	2-D
		1-D	1-D
		KEY	KEY
		PKEY	PKEY
		X>Y??	X>Y??
		CHR$	CHR$
		NUM	NUM
		RDEL	RDEL
		R$	R$
		L$	L$
		LDEL	LDEL
		RHASTO	RHASTO
		CLRKEY	CLRKEY

TOULROM

Rom file name	Toulrom 1A	Toulrom 1B	Toulrom 1C	Toulrom 1D
XROM	12	12	12	12
Version	1A	1B	1C	1D
ROM Revision	TR1A	WW3M		
Date	Sept/Oct 1982	Nov/Dec 1982	Jan/Feb 1983	Mar/Apr 1983
CAT name	PPC ROM 2C	PPC ROM 2C	TOULROM1C	TOULROM1D
	TP	TP	TP	TP
	VP	VP	VP	VP
	X<>ROM	X<>ROM	X<>ROM	X<>ROM
	Y<>T	Y<>T	Y<>T	Y<>T
	Z<>T	Z<>T	Z<>T	Z<>T
	Y<>Z	Y<>Z	Y<>Z	Y<>Z
	BCDBIN	BCDBIN	BCDBIN	BCDBIN
	GTOROM	GTOROM	GTOROM	GTOROM
	RXR	RXR	RXR	RXR
	RXL	RXL	RXL	RXL
	X+Y	X+Y	X+Y	X+Y
	SXL	SXL	SXL	SXL
	NSTO	NSTO	NSTO	NSTO
	NRCL	NRCL	NRCL	NRCL
	CODE	CODE	CODE	CODE
	DECODE	DECODE	DECODE	DECODE
	EN	EN	EN	EN
	DE	DE	DE	DE
	CLPV	CLPV	CLPV	CLPV
	XROMTST	XROMTST	XROMTST	XROMTST
	MELBROM 1A	**MELBROM 1A**	"BHEX"	"BHEX"
	PSIZE	PSIZE	PSIZE	PSIZE
	PPACK	PPACK	PPACK	PPACK
	DIS	DIS	DIS	DIS
	2-D	2-D	2-D	2-D
	1-D	1-D	1-D	1-D
	KEY	KEY	KEY	KEY
	PKEY	PKEY	PKEY	PKEY
	PKEY	X>Y??	X>Y??	X>Y??
	CHR$	CHR$	CHR$	CHR$
	NUM	NUM	NUM	NUM
	RDEL	RDEL	RDEL	RDEL
	R$	R$	R$	R$
	L$	L$	L$	L$
	LDEL	LDEL	LDEL	LDEL
	RHASTO	RHASTO	RHASTO	RHASTO
	CLRKEY	CLRKEY	CLRKEY	CLRKEY
	TOULROM1A	**TOULROM1B**	CLRGX	CLRGX
	REP	REP	REP	REP
		XCAT	XCAT	XCAT
		?X=0	R?	R?
		CLRGX	ECRAN	ECRAN
				CHARGE
				X>ROM
			MEMLOST	MEMLOST

Microcode Functions

1-D

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N. A	12,26	12,27	12,27	12,27	12,27	12,27

Does the same as 2-D, but appends only the decoding of the last digit of X.

(This is a much longer routine than 2-D, since it needs a modification of the called mainframe routine, not available in ROM 0).

– PPC TN Supplement to Technical Notes #10, page 3 –

2-D

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N. A	12,25	12,26	12,26	12,26	12,26	12,26

This decodes the last two digits of X and appends the result to the contents of alpha. (Calls the append routine at 0F09, after using the routine at 0FDD.)

– PPC TN Supplement to Technical Notes #10, page 3 –

BCDBIN

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,6	12,6	12,6	12,8	12,8	12,8	12,8

This function accepts as arguments, values from 0 to 999 (decimal) in X, translates them in their hexadecimal counterparts, and places them in the three last digits od the register. Thus 10, XEQ, alpha, BCDBIN, alpha, replaces X by the NNN 00 00 00 00 00 00 0A, writing the old X value into LAST X. The argument 999 yields the value 00 00 00 00 00 03 E7 in X, with 999 in LAST X. Using other functions, any decimal number may be translated to its hex counterpart in X. With DECODE this can provide a very fast conversion program.

– PPC Technical Notes #10, page 67 –

CHARGE

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	12,44

Equivalent to "LB", load a byte in memory. Assign CHARGE to a key, then when you need to load as synthetic instruction in the program memory press the key in user mode, you get on the display CHARGE _ _ _ then you enter the instruction decimal code. The new instruction is inserted above the program step displayed.

– PPC-T N4, page 24 –

CHR$

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N. A	12,30	12,31	12,31	12,31	12,31	12,31

With a decimal in X, the corresponding character is appended to alpha.

– PPC TN Supplement to Technical Notes #10, page 3 –

CLPV

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,20	12,30	12,31	12,31	12,31	12,31	12,31

Clear Private Status.

CLRGX

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N. A	N. A	N. A	N. A	12,43	12,39	12,39

Clear a specified block of registers. This is a function from the 41CX.

CLRKEY

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N. A	12,37	12,38	12,38	12,38	12,38	12,38

Clears all key assignments, including global label assignments.

(It does not clear the key code from global labels, however, and these assignments could be reactivated by a card read.)

– PPC TN Supplement to Technical Notes #10, page 3 –

CODE

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,16	12,16	12,16	12,16	12,16	12,16	12,16

This is a very good version of the famous routine originally invented by Bill Wickes, as was the next. It takes the hex coding alpha characters from the alpha register and places their named digits in the X register. Y, Z, T and L are all preserved, as, also, are the content of alpha. The function, then may be used again later for the same NNN (or normal number, for that matter), or the various alpha functions can be used to manipulate the used content of alpha, for NNN making, and other characters in the remaining ten-or fourteen, if the contents of P are counted. (I think P is preserved.)

– PPC Technical Notes #10, page 69 –

DE

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,19	12,19	12,19	12,19	12,19	12,19	12,19

The inverse of EN, again a MALMAC version of Jake's original, and milliseconds, rather than parts of a minute. The stack and L are unchanged, but alpha is entirely overwritten by the ten letters. (Zeroes are translated by spaces. Thus CLX, DE clears alpha, and places ten spaces there. Useful for formatting!! 2 bytes can replace about 14.)

– PPC Technical Notes #10, page 69 –

DECODE

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,17	12,17	12,17	12,17	12,17	12,17	12,17

The inverse of CODE, placing into alpha the alpha coding of the 14 digits of X. The stack and L are preserved, but the whole of alpha is cleared, to be replaced by the decoding of X. When encountered in a running program, alpha is not printed, but when SST'ed, or executed from the board, alpha is printed. The functions ends, that is, with an AVIEW, though with advantages, relative to the printer. The AVIEW is not effected at all in a running program.

– PPC Technical Notes #10, page 69 –

DIS

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N. A	12,24	12,25	12,25	12,25	12,25	12,25

Richard wrote this to speed up the disassembler program, improved very much, published in this issue of TN. (see p.71) Given an address in the last four digits of X, in he manner of X<>ROM, it not only fetches the ten bit word from that address in the manner of that function, with the incremented address in Last X, printing the same format as X<>ROM when executed in RUN mode from the keyboard (but not in program mode), but also formatting in alpha the address, the 244 format of the word from that address and the 442 format. Using this disassembler aid, we now can list the contents of a ROM or of an EROM at the rate of one word every two seconds. If the mnemonics are not needed, the listing is printed as fast as the printer can operate. (LBL 01, DIS, PRA, LAST X, GTO 01) In addition, DIS places the decimal equivalents of the 244 format into the stack, ready for entry at LBL 16 of Richard's program "RC" mark III as listed in TN 10 on p.52. Given a bit of address prompting, the new disassembler is virtually driven by DIS.

– PPC TN Supplement to Technical Notes #10, page 3 –

ECRAN

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	N.A.	N.A.	N.A.	N.A.	12,43	12,43

This program reads X, executes BCDBIN and writes the result to the display. It can be used to see the character associated to a given character code with the following user program:

```
01◆LBL "ECRAN"
  02 CF 21

  03◆LBL 01
  04 CLA
 05 "       "
  06 1
  07 +
  08 ARCL X
  09 "├=="
  10 AVIEW
  11 ECRAN
  12 PSE
  13 GTO 01
  14 .END.
```

– PPC-T N4, page 24 –

EN

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,18	12,18	12,18	12,18	12,18	12,18	12,18

This was a surprise! It is a MALMAC version, and very fast too, of Jake Schwartz' alpha packing routine, of which a revision by Gerard Westen is in this issue, which Richard Collett and I had a go at in the last issue, and which Gerard has compacted even further than Richard and I had. (In the Walmolen Report for the next issue. Lovely stuff!) This takes the last ten characters in alpha and packs them into an alpha string in Jake's manner. As far as I have examined (not far at all) it uses Jake's five bit sequences, and not the pattern Richard and I used. This overwrites the contents of X with the alpha string, but the rest of the stack is undisturbed.

– PPC Technical Notes #10, page 69 –

GTOROM

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,7	12,7	12,7	12,9	12,9	12,9	12,9

The GTOROM function takes the address from digits 6, 5,4 and 3 of X, and jumps to that location, continuing execution from there

– PPC TN Supplement to Technical Notes #10, page 3 –

KEY

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	12,27	12,28	12,28	12,28	12,28	12,28

Places in X the code corresponding to any key currently being pressed while the program is running. (N.b. NOT the key code. We are unsure so far exactly where to get that from.)

– PPC TN Supplement to Technical Notes #10, page 3 –

L$

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	12,34	12,35	12,35	12,35	12,35	12,35

Places the leftmost character from alpha into X as an alpha string.

– PPC TN Supplement to Technical Notes #10, page 3 –

LDEL

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	12,34	12,35	12,35	12,35	12,35	12,35

Deletes the leftmost character from alpha.

– PPC TN Supplement to Technical Notes #10, page 3 –

MEMLOST

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.

This function is not listed in CAT2 and has no XROM #. It is associated to the MEMLOST vector at the end of the TOULROM 1C and 1D. It is auto executed upon Memory Lost: does CF 28, FIX 2 and SIZE 026.

– PPC-T N3, page 28 –

NRCL

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,13	12,13	12,13	12,15	12,15	12,15	12,15

Non-Normalizing recall function. Takes address from X and recalls, without normalisation, overwriting the content of X by that of the recalled register. Y, Z and T are unchanged and LAST X is overwritten by the used value of X

– PPC Technical Notes #10, page 68 –

NSTO

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,13	12,13	12,13	12,15	12,15	12,15	12,15

Non-Normalizing store function. Stores the contents of Y into the register addressed by the contents of X. As is the case with the usual indirect storing, the stack is left unchanged.

– PPC Technical Notes #10, page 68 –

NUM

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	12,31	12,32	12,32	12,32	12,32	12,32

This is the inverse of CHR$, but does not delete the last character from alpha.

– PPC TN Supplement to Technical Notes #10, page 3 –

PKEY

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	12,28	12,29	12,29	12,29	12,29	12,29

Stops execution of program until a key is pressed, then returns the same code as KEY to X.

– PPC TN Supplement to Technical Notes #10, page 3 –

PPACK

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	12,23	12,24	12,24	12,24	12,24	12,24

Programmable PACK. This calls the PACK routine at address 11E7 in ROM 1.

– PPC TN Supplement to Technical Notes #10, page 3 –

PSIZE

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	12,14	12,23	12,23	12,23	12,23	12,23

Programmable Size.

– Paul Lind - PPC Technical Notes #9, page 90 & #10, page 27 –

R$

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	12,33	12,34	12,34	12,34	12,34	12,34

Places the rightmost character from alpha into X as an alpha string.

– PPC TN Supplement to Technical Notes #10, page 3 –

R?

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	N.A.	N.A.	N.A.	N.A.	12,43	12,43

R? returns in X the number of registers available between .END. and the first assignment buffer. These registers can be used by SIZE (or PSIZE), for new programs or new assignments.

– PPC-T N3, page 29 –

RDEL

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	12,32	12,33	12,33	12,33	12,33	12,33

Delete rightmost character in alpha.

– PPC TN Supplement to Technical Notes #10, page 3 –

REP

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	N.A.	N.A.	12,40	12,40	12,40	12,40

Copy the value in X to Y, Z and T.

– PPC-T N1, page 14 –

RHASTO

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	12,36	12,37	12,37	12,37	12,37	12,37

Right-Hand ASTO: Alpha-stores the rightmost six characters from alpha in X.

– PPC TN Supplement to Technical Notes #10, page 3 –

RXL

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,9	12,9	12,9	12,11	12,11	12,11	12,11

This, as the function name suggests, rotates the 14 digits of the X register to the left. Provided that the leftmost digits are zeroes, this is the equivalent of multiplying, hexadecimally by 10_{16}. Remember, though, that there is only a maximum range of that 14 digits. The biggest hex number that can be accommodated is FFFFFFFFFFFFF, and because of the limitation of these two routines, this is not readily used.

– PPC Technical Notes #10, page 68 –

RXR

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,8	12,8	12,8	12,10	12,10	12,10	12,10

As the mnemonic suggests, this rotates the digits of the X register to the right. Thus with ! in X RXR results in the NNN 00 01 00 00 00 00 00, for the value1 in X has the digits/bytes 01 00 00 00 00 00 00. This, in hexadecimal terms, is equivalent to diving by 10_{16}. To be such, the exponent digits need to be zeroes, of course. 10, RXR produces the alpha string 10 10 00 00 00 00 00, and a further RXR produces the normal number 1.010… -1, RXR, of course generates 9.1000…

– PPC Technical Notes #10, page 67 –

SXL

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,11	12,11	12,11	12,13	12,13	12,13	12,13

Shift X left one bit.

– PPC Technical Notes #10, page 68 –

TP

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,14	N.A.	12,14	12,2	12,2	12,2	12,2

This is an attempt at a programmable TONE function. It uses values held in X & Y, where X determines the duration, and Y apparently should determine the pitch. Values from 0 to 999are accepted in X and Y, the sign of X is ignored, but a – sign of Y causes the TONE flag in the microprocessor to remain set, so that after execution with such a value in Y, pressing any key causes a faint, high pitched tone. Subsequent execution with a positive value removes this. There were only two tone pitches produced, one for values of Y below about 300, and one above, though pitch seemed erratic. Occasionally I got low tones, but I don't know quite why. The duration runs from a short tick for the value 1 in X, to a prolonged tone for 999. The pitch was usually higher than TONE 9. Even numbers in Y produced fainter tones than odd. Fractions in X usually caused a crash, but sometimes lower pitches. However, 999 in X and Y produced a low tone. I suspect this was an unsuccessful attempt to write a routine which would have the duration determined by the value in X, and the pitch by that in Y, but I am not at all sure.

– PPC Technical Notes #10, page 70 –

VP

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,15	12,15	12,15	12,3	12,3	12,3	12,3

A variable pause. The duration is determined by the value in X, which accepts values from 0 up to 999. There is no pause at all for 0, a normal duration pause for 1, 20 seconds for the value 30, and so on. This is very useful for keying in either numerical values, or for keying in alpha. For me, hesitating, and fumbling, the usual PSE is too short. I have only one digit of a multi-digit number in, before it is off and away. This works nicely.

– PPC Technical Notes #10, page 70 –

X+Y

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,10	12,10	12,10	12,12	12,12	12,12	12,12

This effects a hexadecimal addition of the contents of the X and Y registers (or a binaty addition, depending on your point of view), and leaves Y, Z and T unaffected. When the sign digit exceeds hex F, the leftmost digit is lost. This is valid, then for up to 13 digits, or until the sign digit becomes F.

– PPC Technical Notes #10, page 68 –

X<>ROM

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,2	12,2	12,2	12,4	12,4	12,4	12,4

With an address in the four last digits of the X register, readily generated with the aid of the other functions, in hex digits, execution of this function displaces those digits three to the left, inserting in the three gaps the 244 component of the microcode/machine language word at that address. Thus, with the X register containing the digits 00 00 00 00 00 10 BD, X<>ROM replaces this by 00 00 00 01 0B D3 45, the last three from 10BD (page 0, line B, word D – or word no BD of page 0 of ROM 1). In general, if X contains 00 00 00 00 00 ab cd, X<>ROM returns 00 00 00 0a bc dn nn, where nnn is the word of the ROM at address abcd. After execution, the previous value in X* is preserved in LAST X,* while Y, Z and T are untouched.

– PPC Technical Notes #10, page 67 –

X>ROM

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	12,45

Copy the value in X to MLDL.

– PPC-T N5, page 31 & JPC 14, page 20 –

X>Y??

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	12,29	12,30	N.A.	12,30	12,30	12,30

This does a full binary compare of the contents of X and U – a genuine test, testing true for X greater, hexadecimally, than Y for all digits. Thus, any positive number comes out as less than any alpha string, and any alpha string as less than any negative number. It will need to be used with care with alpha strings, of course, for any alpha string of n characters will test as greater than an alpha string of less than n characters.

– PPC TN Supplement to Technical Notes #10, page 3 –

XCAT

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	N.A.	N.A.	N.A.	12,41	12,41	12,41

XCAT begins the CAT 2 with the module which you indicate by his XROM number in X.

– PPC-T N2, page 14 –

XROMTST

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	N.A.	12,21	12,21	12,21	12,21	12,21

Verify ROM checksum. For example with the PPC ROM2C, with the number of XROM in X (12), executing XROMTST displays "12 PR-2C TST" for a second or two, then "12 PR-2C OK" if in order. If there is no ROM of the addressed number plugged in, the message is instead "NO ROM nn" where nn is the no. addressed. The message for numbers over 999 is "NON EXISTENT".

– PPC Technical Notes #12, page 25 –

Y<>T

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,3	12,3	12,3	12,5	12,5	12,5	12,5

This does exactly what one would expect … exchanging Y and T. It is readily effected by normal functions, of course, but needs three lines, and four bytes (X<>Y, X<>T, X<>Y). I suspect that it was one of the first which Jim wrote when developing the system. An easy one for practise, and readily checked for errors. Of course, this will execute faster than the three line alternative.

– PPC Technical Notes #10, page 67 –

Y<>Z

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,4	12,4	12,4	12,6	12,6	12,6	12,6

Exchange Y and Z.

– PPC Technical Notes #10, page 67 –

Z<>T

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	12,5	12,5	12,5	12,7	12,7	12,7	12,7

Exchange Z and T.

– PPC Technical Notes #10, page 67 –

?X=0

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N.A.	N.A.	N.A.	N.A.	12,42	N.A.	N.A.

No description available. May be part of the 2 programs listed at non-working in PPC-T N2.

– PPC-T N2, page 14 –

User code Functions

BHEX

Module	JIMROM 1H	MELBROM 1A	PPC ROM 2C	TOULROM 1A	TOULROM 1B	TOULROM 1C	TOULROM 1D
XROM#	N. A	N. A	N. A	N. A	N. A	12,22	12,22

User code from Richard Collett (Australia): give the 2716 code from the 244 instruction code for burning EPROMs.

Once a Microcode routine has been written and checked, the next chore is to unpick the ten-bit words of the machine instructions into a 244 format, taking the 44 part to burn into the 8 bit EPROM (easy), and taking the 2 part for four sequential words to produce the eight bits for the other EPROM.

The BHEX routine, using the functions we had had from the start in JIMROM 1H, was written, and eased the task considerably.

User Instructions:

1. Read in program. (Printer must be connected and switched on.)
2. SIZE to 001 or greater.
3. XEQ "BHEX"
4. On the prompt "ADD ?", key in the starting address in the alpha counterpart of the hex digits – as 0 to 9, A to F.
5. R/S. See the two-bit ROM address for the two most significant bits of the first four machine code words, accompanied by two short TONE 57 bips. The correct starting address is printed.
6. Key in 0 to 3 for the first two bits of the first ten-bit word, ENTER↑, key in 0 to 3 for the first two bits of the second ten bit word, ENTER↑, etc., finally keying in the two bits for the fourth ten bit word's most significant bits as 0 to 3.
7. R/S. The output print has the format:

   a:b:c:d xxxx HH

   where a, b, c and d are the above four pairs of most significant bits of the four successive words, as decimal numbers from 0 to 3, xxxx is the two bit EPROM address where the two hex digits, HH are to be burned in.

8. After the print is complete, the two short TONE's sound again – go to step 6 if more words are to be coded.
9. When burning in the two bit EPROM, use the RHS of the resulting printout as the guide to keying into the controlling computer the pairs of hex digits at successive addresses, as given on their left.

– PPC Technical Notes #12, page 55 & PPC-T N3, page 28 –

```
EPROM BOX HEX DIGITS FOR TWO BIT EPROM  Richard Collett (4523)

 01*LBL "BHE
X"
 02 SF 12
 03 FIX 0
 04 CF 29
 05 ADV
 06 "ADD ?"
 07 AON
 08 PROMPT
 09 PRA
 10 CF 12
 11 AOFF
 12 CODE
 13 SXL
 14 SXL
 15 RXR
 16 SXL
 17 SXL
 18 STO 00
19*LBL 00
20 CLX
21 NRCL
22 DECODE
23 CLX
24 STO \
25 "⊢   "
26 STO \
27 TONE 7
28 TONE 7
29 PROMPT
30 CLA
31 ARCL T
32 "⊢:"
33 ARCL Z
34 "⊢:"
35 ARCL Y
36 "⊢:"
37 ARCL X
38 "⊢   "
39 ACA
40 ST+ X
41 ST+ X
42 +
43 ST+ X
44 ST+ X
45 +
46 ST+ X
47 ST+ X
48 +
49 ENTER↑
50*LBL 01
51 CLX
52 NRCL
53 4
54 BCDBIN
55 X+Y
56 STO 00
57 RDN
58 SXL
59 SXL
60 RXR
61 DECODE
62 CLX
63 STO \
64 "⊢   "
65 STO \
66 ACA
67 RDN
68 BCDBIN
69 DECODE
70 CLX
71 STO \
72 "⊢   "
73 STO \
74 ACA
75 PRBUF
76 GTO 00
77 .END.
```

Figure 1 PPC TN #12 original "BHEX" listing

```
01*LBL "BHEX"
02 FIX 0
03 CF 29
04 ADV
05 "BITS      2bAd"
06 "⊢   CODE"
07 PRA
08 ADV
09 "ADD?"
10 AON
11 PROMPT
12 PRA
13 AOFF
14 CODE
15 SXL
16 SXL
17 RXR
18 SXL
19 SXL
20 STO 00
21 CLX
22 NRCL
23 DECODE
24 CLX
25 STO \
26 "⊢   "
27 STO \
28 PROMPT
29 CLA
30 ARCL T
31 "⊢:"
32 ARCL Z
33 "⊢:"
34 ARCL Y
35 "⊢:"
36 ARCL X
37 "⊢   "
38 ACA
39 ST+ X
40 ST+ X
41 +
42 ST+ X
43 ST+ X
44 +
45 ST+ X
46 ST+ X
47 +
48 ENTER↑
49 CLX
50 NRCL
51 4
52 BCDBIN
53 X+Y
54 STO 00
55 RDN
56 SXL
57 SXL
58 RXR
59 DECODE
60 CLX
61 STO \
62 "⊢   "
63 STO \
64 ACA
65 RDN
66 BCDBIN
67 DECODE
68 CLX
69 STO \
70 "⊢   "
71 STO \
72 ACA
73 PRBUF
74 GTO 00
75 END
```

Figure 2 PPC-T N3 updated "BHEX" listing (TOULROM 1C)