

free42 Base-N Tools

Mitch Richling

2021-03-19

Author: Mitch Richling
Updated: 2021-05-13 19:55:18

Copyright 2021 Mitch Richling. All rights reserved.

Contents

1	Metadata	1
2	Introduction	1
3	BASE: BASE-N Application	1
3.1	Functionality & Menu	1
3.2	Menu Code	2
3.3	Application Local Subroutines	7
4	EOF	21

1 Metadata

The home for this HTML file is: <https://richmit.github.io/hp42/base.html>

A PDF version of this file may be found here: <https://richmit.github.io/hp42/base.pdf>

Files related to this document may be found on github: <https://github.com/richmit/hp42>

Directory contents:

src	-	The org-mode file that generated this HTML document
src_42s	-	Ready to convert source listings for 42s code in this document
docs	-	This html document
bin	-	Importable RAW program files

2 Introduction

When I'm doing embedded programming I use my HP-16c & DM16 calculators all the time. The programs here augment the 42's capabilities with every function on the 16c that I use. Functionality beyond the 16c includes comprehensive support for bit fields, better masks, integer square root & logarithms, and more. Finally this software provides a **SHOW**-like function called **BVIEW** is capable of displaying a full 64-bit binary number on a standard 42s display.

Everything is encapsulated into a single program with multiple global labels so that most of the functionality can be called directly from other programs.

The only thing that still bugs me is the annoying way the 42s requires one to enter hexadecimal digits. Not much I can do about that...

3 BASE: BASE-N Application

3.1 Functionality & Menu

Menu:SubMenu:Key	Program	Description
S&R:SLn		Shift Y Left X Bits
S&R:SRn		Shift Y Right X Bits
S&R:RLn		Rotate Y Left by X bits
S&R:RRn		Rotate Y Right by X bits
S&R:□□□□		
S&R:BVIEW		
S&R:LJ		Left Justify Bits.
S&R:RJ		Right Justify Bits
S&R:SHFXY		Shift X by Y Bits (left when Y negative)
S&R:ROTXY		
S&R:□□□□		
S&R:BVIEW		

Continued on next page

Continued from previous page		
Menu:SubMenu:Key	Program	Description
S&R:SL		Shift Left 1 Bit
S&R:SR		Shift Right 1 Bit
S&R:RL		Rotate Left 1 Bit
S&R:RR		Rotate Right 1 Bit
S&R:□□□□		
S&R:BVIEW		
BITS:GETB		Return the Xth bit of Y
BITS:SB	SETB	Set the Xth bit of Y
BITS:CB	CLRB	Clear the Xth bit in Y
BITS:AB	ASNB	Assign X to the Yth bit of Z
BITS:□□□□		
BITS:BVIEW		
BITS:GY@X		Return the Y bits of Z starting at bit X
BITS:SY@X		Set Y bits of Z starting at bit X
BITS:CY@X		Clear Y bits of Z starting at bit X
BITS:AZ@Y		Assign LS Z bits of X to Z bits of T starting at Y
BITS:□□□□		
BITS:BVIEW		
BITS:POPRB		Pop rightmost bits X bits off of Y
BITS:POPLB		Pop leftmost bits X bits off of Y
BITS:□□□□		
BITS:□□□□		
BITS:□□□□		
BITS:BVIEW		
FUN:B#		Count set bits
FUN:MSSB		Position of the Most Significant Set Bit
FUN:LSSB		Position of the Least Significant Set Bit
FUN:ILN2		Integer base 2 log
FUN:ISQRT		Integer square root
FUN:BVIEW		
FUN:REVN		Reverse rightmost X bits of Y
FUN:REVB		Reverse bits of X
FUN:□□□□		
FUN:□□□□		
FUN:□□□□		
FUN:BVIEW		
MASK:MSKL		Create integer with left most X bits set
MASK:MSKR		Create integer with right most X bits set
MASK:MSKn		Create integer with Y set bits located at bit X
MASK:□□□□		
MASK:□□□□		
MASK:BVIEW		
ARTH:AND		
ARTH:OR		
ARTH:XOR		
ARTH:NOT		
ARTH:NEG	BASE+/-	
ARTH:BVIEW		
ARTH:+	BASE+	
ARTH:-	BASE-	
ARTH:×	BASE×	
ARTH:÷	BASE÷	
ARTH:MOD		
ARTH:BVIEW		
BVIEW		
LBL 92	HEXM	Set current base to hexadecimal
LBL 93	DECM	Set current base to decimal
LBL 94	OCTM	Set current base to octal
LBL 95	BINM	Set current base to binary
□□□□	-	
BMNU	LBL 75	Switch to built in BASE menu
LBL 96	WSIZE	Set WSIZE
LBL 97	BSIGNED	Toggle signed/unsigned mode
LBL 98	BWRAP	Toggle wrapped mode
□□□□	-	
LBL 99	LBL 74	Toggle BVIEW padding
□□□□	-	

3.2 Menu Code

The menu program is generated via the following bit of elisp. You must first define the `MJR-generate-42-menu-code` and `MJR-custom-x-gen` by evaluating the code blocks in the `hp42s-meta.org` file. The skeleton was generated by the following code. That skeleton has been fleshed out with some custom code.


```

GTO 04
LBL 16          @@@@ Page 2 of menu S&R
CLMENU
"LJ"
KEY 1 XEQ "LJ"
"RJ"
KEY 2 XEQ "RJ"
"SHFXY"
KEY 3 XEQ "SHFXY"
"ROTXY"
KEY 4 XEQ 18
"BVIEW"
KEY 6 XEQ "BVIEW"
KEY 7 GTO 04
KEY 8 GTO 17
KEY 9 GTO 01
MENU
STOP
GTO 16
LBL 17          @@@@ Page 3 of menu S&R
CLMENU
"SL"
KEY 1 XEQ "SL"
"SR"
KEY 2 XEQ "SR"
"RL"
KEY 3 XEQ "RL"
"RR"
KEY 4 XEQ "RR"
"BVIEW"
KEY 6 XEQ "BVIEW"
KEY 7 GTO 16
KEY 8 GTO 04
KEY 9 GTO 01
MENU
STOP
GTO 17
LBL 05          @@@@ Page 1 of menu BITS
CLMENU
"GETB"
KEY 1 XEQ "GETB"
"SB"
KEY 2 XEQ "SETB"
"CB"
KEY 3 XEQ "CLRB"
"AB"
KEY 4 XEQ "ASNB"
"BVIEW"
KEY 6 XEQ "BVIEW"
KEY 7 GTO 20
KEY 8 GTO 19
KEY 9 GTO 01
MENU
STOP
GTO 05
LBL 19          @@@@ Page 2 of menu BITS
CLMENU
"GY@X"
KEY 1 XEQ "GY@X"
"SY@X"
KEY 2 XEQ "SY@X"
"CY@X"
KEY 3 XEQ "CY@X"
"AZ@Y"
KEY 4 XEQ "AZ@Y"
"BVIEW"
KEY 6 XEQ "BVIEW"
KEY 7 GTO 05
KEY 8 GTO 20
KEY 9 GTO 01
MENU
STOP
GTO 19
LBL 20          @@@@ Page 3 of menu BITS
CLMENU
"POPRB"

```

```

KEY 1 XEQ "POPRB"
"POPLB"
KEY 2 XEQ "POPLB"
"BVIEW"
KEY 6 XEQ "BVIEW"
KEY 7 GTO 19
KEY 8 GTO 05
KEY 9 GTO 01
MENU
STOP
GTO 20
LBL 06          @@@@ Page 1 of menu FUN
CLMENU
"B#"
KEY 1 XEQ "B#"
"MSSB"
KEY 2 XEQ "MSSB"
"LSSB"
KEY 3 XEQ "LSSB"
"ILN2"
KEY 4 XEQ "ILN2"
"ISQRT"
KEY 5 XEQ "ISQRT"
"BVIEW"
KEY 6 XEQ "BVIEW"
KEY 7 GTO 21
KEY 8 GTO 21
KEY 9 GTO 01
MENU
STOP
GTO 06
LBL 21          @@@@ Page 2 of menu FUN
CLMENU
"REVB"
KEY 1 XEQ "REVB"
"REVB"
KEY 2 XEQ "REVB"
"BVIEW"
KEY 6 XEQ "BVIEW"
KEY 7 GTO 06
KEY 8 GTO 06
KEY 9 GTO 01
MENU
STOP
GTO 21
LBL 07          @@@@ Page 1 of menu MASK
CLMENU
"MSKL"
KEY 1 XEQ "MSKL"
"MSKR"
KEY 2 XEQ "MSKR"
"MSKn"
KEY 3 XEQ "MSKn"
"BVIEW"
KEY 6 XEQ "BVIEW"
KEY 9 GTO 01
MENU
STOP
GTO 07
LBL 08          @@@@ Page 1 of menu ARTH
CLMENU
"AND"
KEY 1 XEQ 23
"OR"
KEY 2 XEQ 24
"XOR"
KEY 3 XEQ 25
"NOT"
KEY 4 XEQ 26
"NEG"
KEY 5 XEQ 27
"BVIEW"
KEY 6 XEQ "BVIEW"
KEY 7 GTO 22
KEY 8 GTO 22
KEY 9 GTO 01

```

```

MENU
STOP
GTO 08
LBL 22          @@@@ Page 2 of menu ARTH
CLMENU
"+"
KEY 1 XEQ 28
"_"
KEY 2 XEQ 29
"x"
KEY 3 XEQ 30
"±"
KEY 4 XEQ 31
"MOD"
KEY 5 XEQ 32
"BVIEW"
KEY 6 XEQ "BVIEW"
KEY 7 GTO 08
KEY 8 GTO 08
KEY 9 GTO 01
MENU
STOP
GTO 22
LBL 00 @@@@ Application Exit
EXITALL
RTN
LBL 09          @@@@ Action for menu key LBL 92
HEXM
RTN
LBL 10          @@@@ Action for menu key LBL 93
DECM
RTN
LBL 11          @@@@ Action for menu key LBL 94
OCTM
RTN
LBL 12          @@@@ Action for menu key LBL 95
BINM
RTN
LBL 13          @@@@ Action for menu key LBL 96
WSIZE
RTN
LBL 14          @@@@ Action for menu key LBL 97
BSIGNED
RTN
LBL 15          @@@@ Action for menu key LBL 98
BWRAP
RTN
LBL 18          @@@@ Action for menu key ROTXY
ROTXY
RTN
LBL 23          @@@@ Action for menu key AND
AND
RTN
LBL 24          @@@@ Action for menu key OR
OR
RTN
LBL 25          @@@@ Action for menu key XOR
XOR
RTN
LBL 26          @@@@ Action for menu key NOT
NOT
RTN
LBL 27          @@@@ Action for menu key NEG
BASE+/-
RTN
LBL 28          @@@@ Action for menu key +
BASE+
RTN
LBL 29          @@@@ Action for menu key -
BASE-
RTN
LBL 30          @@@@ Action for menu key ×
BASE×
RTN
LBL 31          @@@@ Action for menu key ÷
BASE÷

```

```

RTN
LBL 32    @@@@ Action for menu key MOD
MOD
RTN
@@@@ Free labels start at: 33

```

3.3 Application Local Subroutines

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ (ref:b#)

```

```

@@@@ DSC: Number of bits set
@@@@ IN:  X: an integer
@@@@ OUT: X: Number of 1 bits in IP(X)
@@@@ FAQ: Runtime complexity  $O(\log_2(X))$ 
@@@@ LBL: Used 70-72
@@@@ UPD: 2021-04-10
@@@@ LBL "B#"
LBL "B#"
FUNC 11    @@@# REQ:free42>=2.5.24
L4STK      @@@# REQ:free42>=3.0
IP
0
X<>Y      @@@@ NUM CNT
1
NOT
X<>Y      @@@@ NUM MASK CNT
LBL 70
RCL ST Y   @@@@ MASK NUM MASK CNT
X<>Y      @@@@ NUM MASK MASK CNT
AND        @@@@ NUM_N MASK CNT
LASTX      @@@@ NUM NUM_N MASK CNT
X=Y?
GTO 71
@@@@ Current bit was set: increment counter
R↓        @@@@ NUM_N MASK CNT
1
STO+ ST T  @@@@ 1 NUM_N MASK CNT
LBL 71
@@@@ Current bit was clear
R↓        @@@@ NUM_N MASK CNT
X=0?
GTO 72
@@@@ Still have bits to check
X<>Y      @@@@ MASK NUM_N CNT
-1
ROTX
X<>Y      @@@@ NUM_N MASK CNT
GTO 70
LBL 72
@@@@ No bits left to check
R↓        @@@@ MASK CNT
R↓        @@@@ CNT
RTN

```

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ (MSKn)

```

```

@@@@ DSC: Create integer with Y set bits located at bit X
@@@@ IN:  Y: An integer
@@@@ IN:  X: An integer (LSB=0)
@@@@ OUT: X: Integer with IP(X) least significant bits set
@@@@ FAQ: Returns int with all 0 bits when X<=0
@@@@ FAQ: Returns int with all 1 bits when X>=WSIZE?
@@@@ UPD: 2021-03-20
LBL "MSKn"
FUNC 21    @@@# REQ:free42>=2.5.24
L4STK      @@@# REQ:free42>=3.0
X<>Y
XEQ "MSKR"
X<>Y
XEQ "SLn"
RTN

```

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ (MSKL)

```

```

@@@@ DSC: Create integer with left most X bits set
@@@@ IN:  X: An integer
@@@@ OUT: X: Integer with IP(X) most significant bits set
@@@@ FAQ: Returns int with all 0 bits when X<=0
@@@@ FAQ: Returns int with all 1 bits when X>=WSIZE?

```


(CYaX)

(SYaX)

(AZaY)

10

```

#####(SL)
#### DSC: Shift Left 1 Bit
#### IN:  X: An integer
#### OUT: X: Integer shifted left 1 bit
#### UPD: 2021-03-20

LBL "SL"
FUNC 11          @### REQ:free42>=2.5.24
L4STK           @### REQ:free42>=3.0
-1
ROTXY
1
NOT
AND
RTN

```

```

##### (SR)
#### DSC: Shift Right 1 Bit
#### IN: X: An integer
#### OUT: X: Integer shifted right 1 bit
#### UPD: 2021-03-20
LBL "SR"
FUNC 11 @### REQ:free42>=2.5.24
L4STK @### REQ:free42>=3.0
1
NOT
AND
1
ROTXY
RTN

```

```

#####(RL)
#### DSC: Rotate Left 1 Bit
#### IN: X: An integer
#### OUT: X: Integer rotated left 1 bit
LBL "RL"
FUNC 11 @### REQ:free42>=2.5.24
L4STK @### REQ:free42>=3.0
-1
ROTXY
RTN
```

```

##### (RR)
#### DSC: Rotate Right 1 Bit
#### IN:  X: An integer
#### OUT: X: Integer rotated right 1 bit
LBL "RR"
FUNC 11          @### REQ:free42>=2.5.24
L4STK            @### REQ:free42>=3.0
1
ROTXY
RTN

```

[illegible]


```

+/-
ROTTY
RTN

##### (RRn)
#### DSC: Rotate Y Right by X bits
#### IN: Y: An integer
#### IN: X: An integer
#### OUT: X: X rotated right ABS(IP(Y)) bit(s)
#### FAQ: Y is returned unchanged when X is zero or negative
#### UPD: 2021-03-20
LBL "RRn"
FUNC 21 @### REQ:free42>=2.5.24
L4STK @### REQ:free42>=3.0
IP
X<>Y
#### Return Y if X was negative
O>= ST Y @### REQ:free42>=3.0.3
XEQ 44 @@@ O>=Y? @### REQ:free42<3.0.3 @### TODO: Delete when DM42 >= free42 3.0.3
RTN
X<>Y
#### X was positive
ROTTY
RTN

##### (SHFX Y)
#### DSC: Shift X by Y Bits (left when Y negative)
#### IN: Y: An integer
#### IN: X: An integer
#### OUT: X: Integer shifted by IP(Y) bit(s)
#### FAQ: Uses SLn & SLr.
#### LBL: Used 59
#### UPD: 2021-03-20
LBL "SHFX Y"
FUNC 21 @### REQ:free42>=2.5.24
L4STK @### REQ:free42>=3.0
IP
X<0?
GTO 59
#### X is non-negative -- shift right
XEQ "SRn"
RTN
LBL 59
#### X negative -- shift left
+/-
XEQ "SLn"
RTN

##### (RJ)
#### DSC: Right Justify Bits
#### IN: X: an integer
#### OUT: Y: Number of shifts required to justify
#### OUT: X: Justified number
#### FAQ: Like the HP-16c's LJ function, but justifies in the other direction
#### LBL: Used 56-58
#### UPD: 2021-03-20
LBL "RJ"
FUNC 12 @### REQ:free42>=2.5.24
L4STK @### REQ:free42>=3.0
IP
0
X<>Y
LBL 57
0
BIT?
GTO 56
GTO 58
LBL 56 @@@@ LSB is one -- DONE!
R↓
X<>Y
RTN
LBL 58 @@@@ LSB is zero
R↓
1
STO+ ST Z
ROTTY

```

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ (LJ)
@@@@ DSC: Left Justify Bits
@@@@ IN:  X: an integer
@@@@ OUT: Y: Number of shifts required to justify
@@@@ OUT: X: Justified number
@@@@ FAQ: Just like the HP-16c's LJ funciton
@@@@ LBL: Used 64-66
@@@@ UPD: 2021-03-20
LBL "LJ"
FUNC 12                @@@# REQ:free42>=2.5.24
L4STK                  @@@# REQ:free42>=3.0
IP
0
X<>Y
LBL 64
WSIZE?
1
-
BIT?
GTO 65
GTO 66
LBL 65    @@@@ MSB is one -- DONE!
R↓
X<>Y
RTN
LBL 66    @@@@ MSB is zero
R↓
1
STO+ ST Z
+/-
ROTXY
GTO 64

```

```

##### (ILN2)
#### DSC: Integer base 2 log
#### IN: X: a positive real number
#### OUT: X: IP(log_2(ABS(X)))
#### UPD: 2021-03-16
LBL "ILN2"
FUNC 11 ##### REQ:free42>=2.5.24
L4STK ##### REQ:free42>=3.0
ABS
LN
2
LN
÷
IP
RTN

```


RTN

@@

@@@@ Menu Label: BVIEW

LBL 99

FC? 02

"BVA●"

FS? 02

"BVA"

RTN

@@

@@@@ Menu Label: HEX

LBL 92

"HEX"

FS? 71

└─"●"

RTN

@@

@@@@ Menu Label: DEC

LBL 93

"DEC"

SF 81

FS? 71

CF 81

FS? 70

CF 81

FS? 68

CF 81

FS? 81

└─"●"

RTN

@@

@@@@ Menu Label: OCT

LBL 94

"OCT"

SF 81

FS? 71

CF 81

FC? 70

CF 81

FS? 81

└─"●"

RTN

@@

@@@@ Menu Label: BIN

LBL 95

"BIN"

SF 81

FC? 68

CF 81

FS? 69

CF 81

FS? 81

└─"●"

RTN

@@

@@@@ Toggle 02

LBL 74

FS?C 02

RTN

SF 02

RTN

@@

@@@@ Menu Action BINM

LBL 75

EXITALL

"Press R/S To"

└─ Return"

AVIEW

HEXM

STOP
RTN

@@
@@@@ DSC: Test if Y<WSIZE
@@@@ NAM: Y≥WSIZE? 67
@@@@ I/O: No stack change. Uses Y
@@@@ RET: YES if Y≥WSIZE, NO otherwise
@@@@ UPD: 2021-04-21
LBL 67
FUNC 00
L4STK
X<>Y
WSIZE?
X>Y?
RTNNO
RTNYES

@@ (REVBIT)
@@@@ DSC: Reverse Bits rightmost X bits of Y
@@@@ NAM: REVBIT
@@@@ IN: Y: An integer
@@@@ X: An integer
@@@@ OUT: X: An integer
@@@@ UPD: 2021-04-21
LBL "REVB"
FUNC 11
L4STK
ENTER @@@@ X X Y
XEQ "MSKR" @@@@ M X Y
X<>Y @@@@ X M Y
1
-
LSTO "_MBC"
1000
÷
LSTO "_CTR"
R↓ @@@@ M Y
RCL ST Y @@@@ Y M Y
OR @@@@ N Y
X<>Y @@@@ OLD NEW
LBL 51
RCL "_CTR" @@@@ CTR OLD NEW
IP
BIT?
GTO 52
R↓
X<>Y
RCL "_CTR"
IP
RCL- "_MBC"
1
X<>Y
ROTXY
NOT
AND
X<>Y
ENTER
LBL 52
R↓
ISG "_CTR"
GTO 51
X<>Y
RTN

@@ (REVBIT)
@@@@ DSC: Reverse all Bits of X
@@@@ NAM: REVBIT
@@@@ IN: X: An integer
@@@@ OUT: X: An integer
@@@@ UPD: 2021-04-21
LBL "REVB"
FUNC 11
L4STK
WSIZE?
XEQ "REVB"

```

(POPRB)
0000 DSC: Shift rihtmost X bits off Y.
0000 IN:   Y: An integer
0000      X: An integer (number of bits)
0000 OUT:  Y: Part of Y left
0000      X: Part of Y shifted off

LBL "POPRB"
FUNC 22
L4STK
RCL ST Y    @@@@ Y  X  Y  ?
RCL ST Y    @@@@ X  Y  X  Y
XEQ "SRn"   @@@@ LFT X  Y  Y
RCL ST Z    @@@@ Y  LFT X  Y
RCL ST Z    @@@@ X  Y  LFT X
XEQ "MSKR"   @@@@ MSK Y  LFT X
AND         @@@@ POP LFT X  Y
RTN
```

[illegible]

18

```

R↓
CF 81
SF 25
RCL "GrMod"
FS?C 25
SF 81
FC? 81
GTO 88
@@@ We are on a DM42
0
STO "GrMod"      @@@ Set graphics mode to 42 classic on DM42
R↓
LBL 88
CLLCD
EXITALL
63
1000
÷
LSTO "_CTR"
LBL 91
LBL 76 @@@ TODO. Not used?
@@@ Figure out X&Y coordinates for digit
RCL "_CTR"      @@@ IF
IP
32
X>Y?
GTO 73
GTO 83
LBL 73          @@@ IF-THEN FIRST ROW
1
RCL "_CTR"
4
×
1
+
GTO 90
LBL 83          @@@ IF-ELSE SECOND ROW
9
RCL "_CTR"
32
-
4
×
1
+
LBL 90          @@@ IF-END
@@@ Figure out current bit value
63
RCL "_CTR"      @@@ IF-BEGIN
IP
-
WSIZE?
X>Y?
GTO 80
GTO 81
LBL 80          @@@ IF-THEN
R↓
+/-            @@@ IF-BEGIN
1
X<>Y
ROTX
RCL "_X"
AND
X=0?
GTO 85
GTO 86
LBL 85          @@@ IF-THEN bit is 0
R↓
0
GTO 87
LBL 86          @@@ IF-ELSE bit is 1
R↓
1
LBL 87
LSTO "_CB"
GTO 82

```


