Append Character To ALPHA By X. - XTA

Dependencies: None
Routines used: ERRAD @ 14E2 ERRDE @ 282D GOTINT @ ϕ2F8
APNDNW @ 2D14
Input: Character code in x, range ϕ≤|x|≤255. Non-integers are truncated
Output: Character with code |x| is appended to right hand of ALPHA register
Errors: ALPHA DATA - If x contains ALPHA string
DATA ERROR - If |x| > 255
Equivalents: Performs same operation as:
XTOA (Ext. Fun. Memory), XTOAR (Ext. I/O), TDC (Synthetic)
TBAB (W.M-J), etc...

```
x3ϕϕ   ϕ81    A:
       ϕ1E     ↗
       ϕ18     X
x3ϕ3           XTA:  READ  X
                     C=C-1  MS
                     C=C-1  MS
                     C  GO          ;ALPHA DATA error if x is
                        ERRAD       ; not numeric
                     A=C    S&X     ;Store exponent in A
                     C=ϕ    MS      ;Ignore sign of x's mantissa
                     C=ϕ    S&X     ;Zero exponent so C contains only mant.
                     ?A#ϕ   XS
                     JC+ØF  GZER    ;If |x| < 1 then append null
                     RCR    12      ;Put 1st digit of mantissa in C[ϕ]
                     A=A-1  S&X     ;Decrement exponent, if now zero set
                     JC+ϕ8  GINT    ;carry - only one digit so jump down
                     RCR    13      ;Rotate so 1st two digits of mant in C[1:ϕ]
                     A=A-1  S&X     ;Decrement exponent, if now zero set carry
                     JC+ϕ5  GINT    ;Two digits only so jump down
                     RCR    13      ;Rotate so 1st 3 digits of mant in C[2:ϕ]
                     A=A-1  S&X     ;Decrement exponent, if now zero set carry
               GERR: NC  GO         ;More than 3 digits so give DATA ERROR
                        ERRDE       ;Also if |x| > 255 from below
               GINT: NC     XQ      ;1,2 or 3 BCD digits in C[S&X] so
                        GOTINT      ;convert it to binary
                     ?C#ϕ   XS      ;If |x| > 255 then jump back and
                     JC-ϕ5  GERR    ; give DATA ERROR
               GZER: PT=ϕ           ;Position pointer ready to receive X binary
                     G=C            ;Put character into G
                     NC  GO         ;Append G (without warning) to the
x31E                    APNDNW      ; right hand edge of ALPHA and end.
```

21 words

Convert Alpha To Upper Case - UPR

Dependencies : None.
Routines Used : XAVIEW @ 0364.
Input : Alpha containing lower case and/or inverse video characters (i.e those with their top bit set which when displayed on the HP-IL Video Interface appear black on white).
Output : Alpha containing upper case (visible) characters.
Errors : None.

```
xx00   R:    092
   1   P     010
   2   U     015
xx03   130   UPR:   LDI                ;Function entry point
   4   004           HEX 004          ;Alpha begins at reg 005 (= 004 + 1)
   5   226   UPR1:   C=C+1   S&X       ;Select next register
   6   270           RAM   SLCT
   7   0E6           B<>C    S&X      ;Save registers address in B
   8   29C           PT=7             ;Use pointer as counter for 7 bytes in register
   9   038           READ  DATA       ;Get data register to be modified
   A   358   UPR2:   ST=C              ;Put character into ST
   B   2884          CLRF  7          ;Mask off the top bit (inverse video etc)
   C   398           C=ST             ;Get modified character
   D   0A6           A<>C    S&X      ;Put character & part of next into A
   E   130           LDI
   F   07B           HEX 07B          ;Upper limit, one above 'z'
xx10   216           C=A+C   XS       ;Get back part of next character
   1   306           ?A<C    S&X      ;If character is not less than 123 dec
   2   043           JNC+08  UPR3     ;  then don't modify it anymore
   3   130           LDI              ;Lower limit, one below 'a'
   4   060           060
   5   0A6           A<>C    S&X      ;Put it in A and character in C
   6   116           A=C     XS       ;Put with limit, the part of next character
   7   306           ?A<C    S&X      ;If character is not greater than 96 dec
   8   013           JNC+02  UPR3     ;  then don't modify it
   9   084           CLRF    5        ;Character is lower case, make it upper case
   A   398   UPR3:   C=ST             ;Put character back in register
   B   23C           RCR     2        ;Move round to next character
   C   3DC           PT+              ;Increment byte counter
   D   394           ?PT=0            ;If not the 7th byte then
   E   363           JNC-    UPR2     ;  go around the loop again
   F   2F0           WRITE   DATA     ;After 7 bytes, write the register back
xx20   0C6           C=B     S&X      ;Get register number
   1   358           ST=C             ;Put in flags for testing
   2   00C           ?FSET   3        ;If not just done register 008 (P)
   3   313           JNC-1E  UPR1     ;  then go and do next register
   4   2CC           ?FSET   13       ;If after all that, we are called
   5   360           C  RTN           ;  in a running program - exit immediately
   6   3B8           READ    14(d)    ;Get flags so we can check for SST
   7   358           ST=C             ;Put up SS0
   8   04C           ?FSET   4        ;If we are being called during
   9   360           C  RTN           ;  a SST then return immediately
   A   191           NC GO   XAVIEW   ;Must have been called from the keyboard
   B   00E                            ;  so AVIEW the modified contents & end
```

44 words (inc. title) Execution time 150ms approx

Code ALPHA To Non-Normalised Number - CODEA

Dependencies : None.
Routines used : User function CODEA uses RCL@122E.
Input : 14 Hex characters in ALPHA register. Non hex characters are accepted though results are possibly obscure. Uses right most digits. If any digits are blank then ϕ is used.
Output : XCODE leaves NNN in CPU registers B and A. Source-ZENROM. CODEA function places NNN in X register observing Stack Lift.
Errors : No errors are indicated.

```
x12C    XCODE: SETDEC                    x253
               A=ϕ  ALL
               A=A-1 ALL
               SETHEX
               READ 6(N)
               R=13
               C=C+C @R
               C=C+C @R
               JC+ϕ2
               A=ϕ  @R
               RCR 13
               A=A+C @R
               ?R=7
               JNC+ϕ3
               READ 5(M)
               RCR 7
               R=R-1
               ?R=13
               JNC-ϕC
               B=A  ALL
x14ϕ           RTN                      x267
                                        x2F6
        ϕ81  A:
        ϕϕ5  E
        ϕϕ4  D
        ϕϕF  O
        ϕϕ3  C
x146    CODEA: NC XQ                    x2FB
                 GOSUB
                 XCODE
               NC GO
x14A             RCL
```

31 words

Extended Beeper Control Functions

DEPENDENCIES: NONE
ROUTINES USED: NONE
INPUT: NONE
ERRORS: NONE
OUTPUT: Sound functions only operate if flag 26 is set (Adio Enable)

CLAXON: Sounds a distorted tone.

RASP: Under normal circumstances (after a SOFF, or tone which leaves no value in the tone register) sounds a rasp. If following a SONS or SONL gives a twitter. If following a TONE 57 or similar gives a click.

SOFF: Turns off the affect of SONS/SONL or tones like TONE 57. This is also achieved by BEEP, TONE, CLAXON and RASP but not merely by turning the machine off and on again.

SONS: Sounds a soft tone whenever the processor is running.

SONL: Sounds a loud tone whenever the processor is running.

```
x0FC  09E  N:
      00F  O
      018  X
      001  A
      00C  L
      003  C
      3B8  READ 14(d)      ; CLAXON entry point
       17C  RCR 6
      3D8  C<>ST
      08C  ?FSET 5
      3A0  NC RTN          ; Return if audio disabled
      130  LDI
      140  HEX 140         ; Set up counter
      2D8  ST<>T           ; Sound buzzer
      358  ST=C
      2D8  ST<>T           ; Sound buzzer
      266  C=C-1 S&X
      3E3  JNC -04         ; Loop until count goes negative
      3E0  RTN
      090  P:
      013  S
      001  A
      012  R
      3B8  READ 14(d)      ; RASP entry point
      17C  RCR 6
      3D8  C<>ST
      08C  ?FSET 5
      3A0  NC RTN          ; Return if audio disabled
      130  LDI
      0FF  HEX 0FF
      358  ST=C            ; Put FF into tone register
      2D8  ST<>T
      130  LDI
      048  HEX 048         ; Duration counter into C
```

```
2D8   ST<>T          ; Sound click
1Ø6   A<>C  S&X      ; Put temporary duration in A
1A6   A=A-1 S&X      ; Count out duration
3FB   JNC-Ø1
266   C=C-1 S&X      ; Next duration
3DB   JNC-Ø5         ; Jump back until duration <Ø
Ø2B   JNC+Ø5         ; Jump down and reset tone register
Ø86   F:
ØØ6   F
ØØF   O
Ø13   S
3C4   ST=Ø           ; SOFF entry point (Rasp exits through this)
3B8   READ 14(d)     ; For soff request tone state = ØØ
17C   RCR 6
3D8   C<>ST          ; Save tone state (requested) in c whilst
Ø8C   ?FSET 5        ;    audio flag is checked
3AØ   NC RTN         ; Return if audio disabled
3D8   C<>ST          ; Get back tone state requested
2D8   ST<>T          ; Put into tone register
3EØ   RTN
Ø93   S:
ØØE   N
ØØF   O
Ø13   S
3C4   ST=Ø           ; SONS entry point
Ø48   SETF 4         ; Request tone state = 1Ø
393   JNC-ØE         ; Use common code in soff above
Ø8C   L:
ØØE   N
ØØF   O
Ø13   S
3C4   ST=Ø           ; SONL entry point
388   SETF Ø         ; Request tone state = Ø1
x13F  35B   JNC-15   ; Use common code in SOFF above
```