Appendix C

Software listing for the single phase to two phase cycloconverter

The software is for the TMS320E14 digital signal processor and is written in assembler. For a guide to the pseudo code documentation, see Appendix G.
; MACRO LIBRARY

; CLEAR A SELECTED BIT
; A IS BIT NUMBER
; B IS VAR
;
SBIC $MACRO A,B
LAC ONE,:A:
XOR MINUS
AND :B:
SACL :B:,0
$ENDM
.END

; SET SELECTED BIT
; A IS BIT NUMBER
; B IS VAR
;
SBIS $MACRO A,B
LAC ONE,:A:
OR :B:
SACL :B:,0
$ENDM
.END

; TEST SELECTED BIT
; A IS BIT NUMBER
; B IS VAR TO TEST
;
SBIT $MACRO A,B
LAC ONE,:A:
AND :B:
$ENDM
.END

; NEGATE VAR A
;
NEG $MACRO A
ZAC
SUB :A:,0
SACL :A:,0
$ENDM
.END

; DECREMENT THE ACCUMULATOR, AN AUXILIARY REGISTER, OR MEMORY
; ACCUMULATOR: DEC
; MEMORY: DEC A
; AUXILIARY REGISTER: DEC ,AR (AR = 0 OR 1)
;
DEC $MACRO A,B
$IF A.L=0
$IF B.L=0
SUB ONE,0
$ELSE
LARP :B:
MAR *-
$ENDIF
$ELSE
LAC :A:,0
SUB ONE,0
SACL :A:,0
$ENDIF
$ENDM
.END

; INCREMENT AC, AR OR MEMORY
; ACCUMULATOR: INC
; MEMORY: INC A
; AUXILIARY REGISTER: INC ,AR
;
INC $MACRO A,B
$IF A.L=0
$IF B.L=0
ADD ONE,0
$ELSE
LARP :B:
MAR *
$ENDIF
$ELSE
LAC :A:,0
ADD ONE,0
SACL :A:,0
$ENDIF
$ENDM
.END

; make a variable global and reserve a memory location
;
data $MACRO A
.def :A:
.bss :A:,1
$ENDM
.END

; SELECT NEW I/O BANK
; FORMAT: BANK A where A is 0 to 7 or EXT
;
BANK $MACRO A
$IF A.L>1
OUT MINUS,7
$ELSE
$IF A.V=0
OUT ZERO,7
$ELSE
$IF A.V=1
OUT ONE,7
$ELSE
$IF A.V=2
OUT TWO,7
$ELSE
$IF A.V=3
OUT THREE,7
$ELSE
$IF A.V=4
OUT FOUR,7
$ELSE
$IF A.V=5
OUT FIVE,7
$ELSE
$IF A.V=6
OUT SIX,7
$ELSE
$IF A.V=7
OUT SEVEN,7
$ELSE
OUT MINUS,7
$ENDIF
$ENDIF
$ENDIF
$ENDIF
$ENDIF
$ENDIF
$ENDIF
$ENDIF
; make a constant global and give it a value
;
data $MACRO A,B
  .def :A:
  :A: .set :B:
$ENDM
.END

; READ CURRENT AND AVERAGE OVER PERIOD
; A IS WHERE RESULT IS STORED
;
READI $MACRO A
 ; load integral of aux. current from a/d
  BANK 0
  IN  TEMP1,0
  LAC TEMP1
  AND ADMASK
; subtract offset
  SUB ADZERO
  SACL TEMP1
; average over period: multiply by inverse of period
  LT TEMP1
  MPY PERINV
  PAC
  SACL :A:
$ENDM
.END
.title "DEFN"

; Defines all variables, flags and constants and allocates data memory
.mlib "mac.lib"

; PROGRAM:
.data variables
.data ZERO, ; contains 0
.data ONE, ; contains 1
.data TWO, ; contains 2
.data THREE, ; contains 3
.data FOUR, ; contains 4
.data FIVE, ; contains 5
.data SIX, ; contains 6
.data SEVEN, ; contains 7
.data MINUS, ; contains -1
.data WDRES1, ; contains 0ABCDh for watchdog reset
.data WDRES2, ; contains 02345h for watchdog reset
.data UMASK, ; P1 mask for turning off U triacs. contains 0011B
.data VMASK, ; P1 mask for turning off V triacs. contains 1100B
.data FLUXMAX, ; contains FLUXMAX
.data ACCEL, ; contains ACCEL * 2^8
.data PULSE, ; contains PULSE
.data FLUXREF, ; contains table address flxref
.data VX4, ; contains Vc * 4 (defined in TABLES.DOC)
.data TAL, ; storage of AL during timer interrupt routine
.data TAH, ; storage of AH during timer interrupt routine
.data TBSR, ; storage of BSR during timer interrupt routine
.data TSTAT, ; storage of status during timer interrupt routine
.data SAL, ; storage of AL during SAMPTIM interrupt routine
.data SAH, ; storage of AH during SAMPTIM interrupt routine
.data SBSR, ; storage of BSR during SAMPTIM interrupt routine
.data STREG, ; storage of T during SAMPTIM interrupt routine
.data SSTAT, ; storage of status during SAMPTIM interrupt routine
.data TEMPT, ; temporary register during timer interrupt routine
.data UFLUX, ; actual flux for U phase
.data VFLUX, ; actual flux for V phase
.data UBOOST, ; integral of boost voltage for U phase
.data VBOOST, ; integral of boost voltage for V phase
.data RES34, ; holds value of expressions 3+4+5 of algorithm
.data PHIN, ; input phase (0 - 45)
.data PHOUT, ; output phase (0 - 180)
.data UFLAG, ; indicates program next to run for U phase
.data PRECU, ; holds value of expressions 1+2 of algorithm for U phase
.data PRECV, ; holds value of expressions 1+2 of algorithm for V phase
.data INTEGU, ; integral of flux for U phase
.data INTEGV, ; integral of flux for V phase
.data FLAGS, ; various flags
.bit2 flag indicating last trigger order. 1: U phase first
.bit3 start- flag indicating start of half cycle
.bit5 end - end of half cycle flag
.bit6 endstpu- endstop disable. 1: disable endstop
.bit7 endstpv- endstop disable. 1: disable endstop
.bit8 mn - mains polarity. 1 = +'ve
.data BFLAGS, ; flags changed in background
.bit0 ur - U phase background calculations in progress
.bit1 vr - V phase background calculations in progress
.bit3 pll- set to 1 to run phase locked loop routine
.bit4 flag set by pll routine to indicate it has lost control
    and a restart is required.
.bit5 set by pll routine to indicate the phase is out of limits
.data VFLAG, ; indicates program next to run for V phase
data TRIAC, ;stores control for triac firing circuits
data UPULSE, ;time counter for U trigger pulse width
data VPULSE, ;time counter for V trigger pulse width
data TRIMASK, ;stores mask for switching banks
  (0 - positive, 1 - negative)
data VBANK, ;stores which bank is operating for V phase
  (0 - positive, 1 - negative)
data TSUFL, ;U phase flux reading at start of period
data TSVFL, ;V phase flux reading at start of period
data TEMP, ;temporary registers
data TEMP1,
data TEMP2,
data INTBU, ;stores integral of boost flux for U phase
data INTBV, ;stores integral of boost flux for V phase
data TMP, ;temporary register
data OUTSUM, ;stores remainder for PHOUT calcs. Q8
data SPDOLD, ;old speed selection, 256*2^8 = 25Hz
data SPDNEW, ;new speed selection, 256*2^8 = 25Hz
data SPORT, ;bit 9 contains last reading of port 9
data VERGU, ;stores remainder in boost integral calcs-U Phase
data VERGV, ;stores remainder in boost integral calcs-V Phase
data STIME, ;half cycle counter for speed v/f sampling time
data PHV, ;stores output phase for V phase
data UCOUNT, ;stores previous flux count for U phase
data VCOUNT, ;stores previous flux count for V phase
data SCOUNT, ;stores previous count for speed v/f counter
data TFT2, ;stores value of T2 -Tf in algorithm
data BPHASE, ;background: output phase
  flux integral
data BFINT, ;background: flux integral
data BREM, ;background: output phase remainder
data BTEMP, ;background: temporary register
  data BTEMP1, ;background: temporary register
data FUT1, ;reference flux at start of interval, U phase
  reference flux at start of interval, V phase
data COUNT, ;stores input phase for initialisation routine
  data SPIN, ;input speed read from v/f * 2^8
data SPDTMP, ;holds selected speed (temporary)
data TS, ;software counter for speed v/f converter
  data TSREAD, ;current TS sampled value
data PHUB, ;value of PHU at start of interval for use in BPU
  data PHVB, ;value of PHV at start of interval for use in BPV
data OUTSUB, ;value of OUTSUM at start of interval for use in BPU
  data OUTSVB, ;value of OUTSUM at start of interval for use in BPV
data WDBASE, ;nominal sample period in WDT (watchdog timer)

counts
data WDADJ, ;WDADJ/2^12 = signed adjustment to WDBASE
data WDBEM, ;remainder after WDADJ is added to WDBASE
data WPDER, ;current sample period in WDT counts
  data WDCOUNT, ;WDT counts from start of mains cycle to start of
  current sample period
  data MAINPER, ;WDT count in last mains cycle
  data MAINPH, ;WDT count from start of mains cycle to mains

negative zero cross
data UVFSET, ;V/F count per WDT count for U phase. 1 is 2^16
  data VVFSET, ;V/F count per WDT count for V phase. 1 is 2^16
data UVFREM, ;remainder of U V/F count after flux is adjusted for
  offset
  data VVFREM, ;remainder of V V/F count after flux is adjusted for
  offset
  data PLL_P, ;input phase error, pll routine
  data PLL_IH, ;upper byte, pll integrator output
  data PLL_IL, ;lower byte, pll integrator output
  data TEST1, ;USED FOR DEBUG
  data TEST2, ;USED FOR DEBUG

;CONSTANTS
cons ACCELc,25 ;acceleration and deceleration rate.
   ;5 gives 5Hz per second. < 256
cons T1T2c,45 ;length of trigger period. < 128
cons T1T2_2c,90 ;twice length of trigger period. < 255
cons OUTPERc,180 ;output period. < 256
cons OUTPER4c,45 ;output period/4
cons PULSEc,2 ;trigger pulse width in sample periods
cons FLUXMAXc,240 ;clip level for input flux. < 249 for no
overflow in CAL34 routines.
cons WDBASEc,166 ;nominal sample period to be loaded into
WDBASE. 166 for 50Hz, 24MHz xtal.
cons VFSETc,094Eh ;v/f counts per WDT count * 2^16 for v/f
offset adjust. 94Eh for 24MHz, B2Bh for 20MHz.
cons PLL_NFc,5 ;natural frequency of the phase locked loop * 0.2Hz. <=10.
cons PLL_NF1c,PLL_NFc*52
cons PLL_NF2c,29*PLL_NFc*PLL_NFc
cons PLL_ERRc,2780 ;maximum allowed phase error at input to phase
locked loop. < 2^12.
cons MCYC_LOc,14600 ;lower limit of mains period in WDT counts for
use in start-up tests
cons MCYC_HIc,15400 ;upper limit of mains period in WDT counts for
use in start-up tests
   ;the nominal count is 15000 for 50Hz, 24MHz
clock and 12500 for 50Hz, 20MHz clock
END
.title "MAIN"

; This routine is the first to run in the cycloconverter software.
; It initialises memory locations and hardware for the correct operation of the cycloconverter then jumps to SAMPTIM.

.mlib "mac.lib"

;DATA:
; .ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
; .ref FLUXMAX,ACCEL,VX4,VERGU,VERGV,VBCOST
; .ref TS,UFLUX,VFLUX,FLAGS,PHOUT,UBANK,VBANK,WDBASE
; .ref OUTSUM,SPDOLD,SPDNEW,UCOUNT,VCOUNT,SCOUNT,STIME
; .ref TEMP,TEMP1,FLUXREF
; .ref VMASK,UMASK,BFLAGS,SPORT,UVFSET,VVFSET
; .ref TAH,TAL,TBSR,TSTAT,TEMPT
; .ref WDRES1,WDRES2,WDPER,PULSE

;CONSTANTS
; .ref FLUXMAXc,ACCELc,Vc,WDBASEc,FPEAKc,VFSETc,PULSEc

;ADDRESSES:
; .ref samptim1,samptim2,imains0,ispeed0,flxref,cmains0,zerovf0
; .def imains1,ispeed1,cmains1,zerovf1

;VECTOR JUMP TABLE
;
;RESET vector
B main

;interrupt vector - polling routine
;store A and BSR
SACL TAL
SACH TAH
IN TBSR,7
;store status in TSTAT
SST TSTAT
;check for CAP0 interrupt (mains negative zero cross) and interrupt mask clear
;check for CAP0 interrupt
BANK 0
IN TEMPT,4
SBIT 8,TEMPT
BZ int1
;check for interrupt mask clear (0,1)
IN TEMPT,5
SBIT 8,TEMPT
BNZ int1
B imains0
;CAP0 interrupt and interrupt mask is clear (0,1)#
;branch to mains interrupt routine
;nil
;not CAP0 interrupt or interrupt mask is set #(0,1)
;check for CAP1 interrupt for speed input negative transition
int1 SBIT 9,TEMPT
BNZ ispeed0
;CAP1 interrupt (0,1)#
;branch to speed interrupt routine
;nil
;not CAP1 interrupt #(0,1)
;branch to sample routine
int2 B samptim1

imains1:
ispeed1:
;restore registers
LST  TSTAT
OUT  TBSR,7
ZALH  TAH
ADDS  TAL
;enable interrupts
EINT
;return from interrupt
RET

;START OF MAIN PROGRAM
main:
;clear overflow mode
ROVM
;set data memory page pointer to 1
LDPK 1
;disable interrupts
DINT
;clear interrupt flags
BANK 0
ZAC
SACL  TEMP
OUT  TEMP,4
;set auxiliary register pointer to 0
LARP 0
;clear data memory
LARK AR0,255
ZAC
dmem  SACL *
BANZ  dmem

;initialise data memory constants
;ZERO = 0
LACK  0
SACL  ZERO
;ONE = 1
LACK  1
SACL  ONE
;TWO = 2
LACK  2
SACL  TWO
;THREE = 3
LACK  3
SACL  THREE
;FOUR = 4
LACK  4
SACL  FOUR
;FIVE = 5
LACK  5
SACL  FIVE
;SIX = 6
LACK  6
SACL  SIX
;SEVEN = 7
LACK  7
SACL  SEVEN
;MINUS = -1
ZAC
DEC
SACL  MINUS
;FLUXMAX = FLUXMAXc
LT  ONE
MPYK  FLUXMAXc
PAC
SACL  FLUXMAX
;WDBASE = WDBASEc
LT  ONE
MPYK WDBASEc
PAC
SACL WDBASE

;UVFSET, VVFSET = VFSETc
LT ONE
MPYK VFSETc
PAC
SACL UVFSET
SACL VVFSET

;ACCEL = ACCELc * 2^8
LACK ACCELc
SACL ACCEL
LAC ACCEL, 8
SACL ACCEL

;VX4 = Vc * 4
LACK Vc
SACL VX4
LAC VX4, 2
SACL VX4

;FLUXREF = flxref
LT ONE
MPYK flxref
PAC
SACL FLUXREF

;UMASK = 0011B
LAC THREE
SACL UMASK

;VMASK = 1100B
LAC THREE, 2
SACL VMASK

;WDRES1 = 0ABCDh
LACK 0ABh
SACL TEMP
LACK 0CDh
ADD TEMP, 8
SACL WDRES1

;WDRES2 = 02345h
LACK 023h
SACL TEMP1
LACK 045h
ADD TEMP1, 8
SACL WDRES2

;PULSE = PULSEc
LT ONE
MPYK PULSEc
PAC
SACL PULSE

;initialise flux registers: U phase at 0 degrees, V phase at 90 degrees
ZAC
SACL UFLUX
LT ONE
MPYK FPEAKc
PAC
SACL VFLUX

;initialise sundry registers
LACK 10
SACL STIME
LAC ONE, 15
SACL VERGU
SACL VERGV

;start-up delay of 0.5 seconds to allow circuits to settle
;set up counter
LAC ONE, 5
SACL TEMP
;main loop (1,?)

sd1:
  ;delay of 16.4 msec.
  LAC ONE,15

sd2
SUB ONE
BNZ sd2
  ;test main loop counter
  ZALS TEMP
  SUB ONE
  SAACL TEMP
  BNZ sd1

;measure mains period over eight cycles then adjust WDCOUNT, WDBASE, WDADJ and PLL_IH

B cmains0

cmains1:
;calculate v/f offsets for zero inputs
B zerovf0

zerovf1:
;DEBUG switch to negative bank
  LAC ONE
  SAACL UBANK
  SAACL VBANK

;set all ports high
  BANK 0
  OUT MINUS,2

;set ports 0 to 3, 10 to 15 as outputs, 4 to 9 as inputs
;set DDR to 1111 1100 0000 1111
  LAC MINUS,10
  ADD THREE,2
  ADD THREE
  SAACL TEMP
  BANK 0
  OUT TEMP,1

;set periods of timers 1 & 2 to 256
;load period comparators with 255
  BANK 2
  LAC ONE,8
  SUB ONE
  SAACL TEMP
  OUT TEMP,1
  OUT TEMP,3

;set CAP0 and CAP1 interrupts for negative transition
;load CCON register with 0000 0000 0010 0010
  LAC ONE,1
  ADD ONE,5
  SAACL TEMP
  BANK 6
  OUT TEMP,4

;disable compare subsystems, select external clock and no prescaling for timers 1 & 2, enable capture interrupts
;load TCON with 0011 1000 1111 0111
  BANK 2
  LACK 11110111B
  ADD SEVEN,11
  SAACL TEMP
  OUT TEMP,4

;set watchdog timer period to WDBASE-1 and WDPER to WDBASE
  LAC WDBASE
  SAACL WDPER
  SUB ONE
  SAACL TEMP
  BANK 1
  OUT TEMP,1

;set interrupt mask for watchdog timer and CAP1 interrupts only
;load IM with 1111 1110 1111 1101
  BANK 0
LAC  ONE,1
ADD  ONE,9
XOR  MINUS
SACL  TEMP
OUT  TEMP,5
;wait for -'ve half cycle: do two tests separated by 1ms for noise rejection
;select bank 0
BANK  0
;wait loop (0,?)
;wait for negative mains (0,?)
ploop  IN              TEMP,0
SBIT    8,TEMP
BNZ             ploop
;delay 1.024ms
LAC             ONE,10
delay NOP
NOP
SUB    ONE
BNZ    delay
;retest for negative mains
IN    TEMP,0
SBIT    8,TEMP
BNZ    ploop
;wait for +'ve half cycle. Phase timers can then be sync'ed to mains
nloop  IN    TEMP,0
SBIT    8,TEMP
BZ    nloop
;set timers 1 & 2 to zero
BANK  2
ZAC
SACL    TEMP
OUT    TEMP,0
OUT    TEMP,2
;reset watchdog timer
BANK  1
OUT    WDRES1,0
OUT    WDRES2,0
;set interrupt flags
BANK  0
OUT    MINUS,6
;set mains polarity flag to 0 - indicates state of previous half cycle
;nil = already 0
;set FIFO-0 AND FIFO-1 to allow capture interrupts
BANK  6
IN    TEMP,0
IN    TEMP,0
IN    TEMP,0
IN    TEMP,0
IN    TEMP,1
IN    TEMP,1
IN    TEMP,1
IN    TEMP,1
;n clear fifo full flags for 0 & 1
LAC    ONE,3
ADD    ONE,7
SACL    TEMP
OUT    TEMP,5
;enable interrupts
EINT
;set samptim routine
B samptim2
.end
.title "BACK"

;Background calculations routine.
;This program controls the calculation of the pre-calculated integrals.
;This routine although called by the SAMPTIM interrupt routine may itself be interrupted by the SAMPTIM routine. This however is allowed to occur only to the first level.

.mlib "mac.lib"

;DATA:
.ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
.ref BFLAGS,BTEMP1

;ADDRESSES:
.def back,bpu1,bpv1,speed1,pll1
.ref bpu,bpv,speed,pll0

;PROGRAM:
back:
;call U phase background program
B bpu

bpu1:
;clear U phase background flags
SBIC 0,BFLAGS

;call V phase background program
B bpv

bpv1:
;clear V phase background flags
SBIC 1,BFLAGS

;call speed routine
B speed

speed1:
;check if phase locked loop routine is to run: check BFLAG 3
SBIT 3,BFLAGS
BZ x1

;pll routine is to run (0,1)
;clear b flag 3
SBIC 3,BFLAGS
;run pll routine
B pll10

pll1:

x1:
;wait for next half cycle
;no operation (0,.)

iloop: NOP
B iloop

.end
.title "BOOST"

;This routine determines the integral of boost voltage for the current
algorithm calculation period.

.mlib "mac.lib"

; CALLED FROM: SAMPTIM

; DATA:
-ref ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, MINUS
-ref PHOUT, VERGU, UBOOST, PHV, TEMP
-ref VERGV, VBOOST

; ADDRESSES:
-ref boost
-ref bstref

; PROGRAM (26)
boost:
; Calculate integral of boost flux for this algorithm time - this is
done by adding the boost voltage at this sample time to the integral
subregister VERGU.
; Allow VERGU to overflow into UBOOST, as in the upper byte of a double
precision variable where UBOOST is initially zero. UBOOST is to be
added to the flux then zeroed.

; calculate boost integral for U phase
; table lookup of boost voltage
LT ONE
MPLYK bstref
PAC
ADD PHOUT
TBLR TEMP
; add to the integral subregister
ZALS VERGU
ADD TEMP
SACL VERGU
; add upper word to UBOOST
ADDH UBOOST
SACH UBOOST

; calculate boost integral for V phase
; table lookup of boost voltage
LT ONE
MPLYK bstref
PAC
ADD PHV
TBLR TEMP
; add to the integral subregister
ZALS VERGV
ADD TEMP
SACL VERGV
; add upper word to VBOOST
ADDH VBOOST
SACH VBOOST

; return
RET

.end
.title "BPU"

; U PHASE BACKGROUND PROGRAM
; This routine is responsible for calculating integrals 1 and 2 of the control algorithm. It runs in the background of the sample time interrupts.

.mlil "mac.lib"

; CALLED FROM: BACK

; DATA:
.data
  .ref ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, MINUS
  .ref PRECU, T1T2c, OUTSUM, SPDOLD
  .ref BPHASE, BFINT, BREM, FUT1, BTEMP
  .ref TSUFL, FLUXREF
  .ref PHUB, OUTSUB, VX4, OUTPER4c

; ADDRESSES:
  .def bpu
  .ref bpu1

; (754)
; PROGRAM:
.bpu:
; find 1(a), integral of reference flux, by numerical integration using trapezoidal rule
; initialise BPHASE to output phase at start of period
  LAC PHUB
  SACL BPHASE
; load half initial flux value into BFINT
  ADD  FLUXREF
  TBLR  FUT1
; divide by two and put in BFINT
  LAC  FUT1, 15
  SACH BFINT
; initialise registers for numerical integration {5}
; BREM = OUTSUB, remainder for output phase
  LAC  OUTSUB
  SACL BREM
; AR0 = T1T2c-2, loop counter
  LARK 0, T1T2c-2
  MAR *, 0
; integration loop (T1T2c-1 times) {16}
.loop:
; adjust output phase, BPHASE
; add speed value to output phase remainder
  ZALS BREM
  ADDS SPDOLD
  SACL BREM
; add carry to output phase reference, BPHASE
  ADDH BPHASE
  SACH BPHASE
; table lookup of flux value at current output phase
  LAC  FLUXREF
  TBLR BTEMP
; add to flux integral register, BFINT
  LAC  BFINT
  ADD  BTEMP
  SACL BFINT
; check loop count
  BANZ loop
; add half of last flux value {13}
;adjust output phase, BPHASE
;add speed value to output phase remainder
ZALS BREM
ADDS SPDOLD
SACL BREM

;add carry to output phase reference, BPHASE
ADDH BPHASE
SACH BPHASE

;table lookup of flux value at current output phase
LAC FLUXREF
ADD BPHASE
TBLR BTEMP
;divide by 2
LAC BTEMP,15

;add to flux integral register
ADDH BFINT
SACH BFINT

;scale for correct voltage - multiply by volts X 4 then divide by 2^12
LT VX4
MPY BFINT
PAC
SACH BFINT,4

;load integral of flux into precalc sum {2}
LAC BFINT
SACL PRECU

;calculate 1b
;table lookup of flux value at current output phase
LAC FLUXREF
ADD BPHASE
TBLR BTEMP
LAC BTEMP

;subtract FUT1
SUB FUT1
SACL BTEMP

;multiply by T1T2c then divide by 2
LT BTEMP
MPYK T1T2c
PAC
SACL BTEMP
LAC BTEMP,15
SACH BTEMP

;scale - multiply by volts X 4 then divide by 2^12
LT VX4
MPY BTEMP
PAC
SACH BTEMP,4

;add to precalc sum
LAC BTEMP
ADDH PRECU
SACH PRECU

;calculate 2
;multiply TSUFL by T1T2c then divide by 2
LT TSUFL
MPYK T1T2c
PAC
SACL BTEMP
LAC BTEMP,15

;add to precalc sum {2}
ADDH PRECU
SACH PRECU

;negate above
ZAC
SUB PRECU
SACL PRECU

;scale FUT1 for later use - multiply by volts X 4 then divide by 2^12
LT VX4
MPY FUT1
PAC
SACH FUT1,4

;return
B bpul

.end
.title "BPV"

;V PHASE BACKGROUND PROGRAM
;This routine is responsible for calculating integrals 1 and 2 of the control algorithm. It runs in the background of the sample time interrupts.

.mlib "mac.lib"

;CALLED FROM: BACK

;DATA:
..ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
..ref PRECV,T1T2c,OUTSUM,SPDOLD
..ref BPHASE,BFIN,TREM,FVT1,BTEMP
..ref TSVFL,FLUXREF
..ref PHVB,OUTSVB,VX4,OUTPER4c

;ADDRESSES:
..def bpv
..ref bpv1

;(754)
;PROGRAM:
bpv:
;find 1(a), integral of reference flux, by numerical integration using trapezoidal rule
;initialise BPHASE to output phase at start of period
LAC PHVB
SACL BPHASE
;load half initial flux value into BFINT
;table lookup of flux
ADD FLUXREF
TBLR FVT1
;divide by two and put in BFINT
LAC FVT1,15
SACH BFINT
;initialise registers for numerical integration {5}
;BREM = OUTSVB, remainder for output phase
LAC OUTSVB
SACL BREM
;AR0 = T1T2c-2, loop counter
LARK 0,T1T2c-2
MAR *,0
;integration loop (T1T2c-1 times) {16}
loop:
;adjust output phase, BPHASE
;add speed value to output phase remainder
ZALS BREM
ADDH SPDOLD
SACL BREM
;add carry to output phase reference, BPHASE
ADDH BPHASE
SACH BPHASE
;table lookup of flux value at current output phase
LAC FLUXREF
ADD BPHASE
TBLR BTEMP
;add to flux integral register, BFINT
LAC BFINT
ADD BTEMP
SACL BFINT
;check loop count
BANZ loop
;add half of last flux value {13}
; adjust output phase, BPHASE
; add speed value to output phase remainder
ZALS BREM
ADDS SPDOLD
SACL BREM
; add carry to output phase reference, BPHASE
ADDH BPHASE
SACH BPHASE
; table lookup of flux value at current output phase
LAC FLUXREF
ADD BPHASE
TBLR BTEMP
; divide by 2
LAC BTEMP,15
; add to flux integral register
ADDH BFINT
SACH BFINT
; scale for correct voltage - multiply by volts X 4 then divide by 2^12
LT VX4
MPY BFINT
PAC
SACH BFINT,4
; load integral of flux into precalc sum {2}
LAC BFINT
SACL PRECV

; calculate 1b
; table lookup of flux value at current output phase
LAC FLUXREF
ADD BPHASE
TBLR BTEMP
LAC BTEMP
; subtract FVT1
SUB FVT1
SACL BTEMP
; multiply by T1T2c then divide by 2
LT BTEMP
MPYK T1T2c
PAC
SACL BTEMP
LAC BTEMP,15
SACH BTEMP
; scale - multiply by volts X 4 then divide by 2^12
LT VX4
MPY BTEMP
PAC
SACH BTEMP,4
; add to precalc sum
LAC BTEMP
ADDH PRECV
SACH PRECV

; calculate 2
; multiply TSVFL by T1T2c then divide by 2
LT TSVFL
MPYK T1T2c
PAC
SACL BTEMP
LAC BTEMP,15
; add to precalc sum {2}
ADDH PRECV
SACH PRECV

; negate above
ZAC
SUB     PRECV
SACL    PRECV

; scale FVT1 for later use - multiply by volts X 4 then divide by 2^12
LT     VX4
MPY     FVT1
PAC
SACH    FVT1,4

; return
B     bpvl

.end
.title "CAL34U"

; This routine performs the sample time calculations for U phase.
; ie.it calculates equations 3 and 4 of the firing algorithm.

.mlib "mac.lib"

; CALLED FROM: PU2

; DATA:
    .ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
    .ref PHIN,RES34
    .ref INTEGU,UFLUX,UBANK,TEMP,T1T2c

; ADDRESSES:
    .def cal34u
    .ref caltab,cal34u1

; (33)
; PROGAM:
    cal34u:
    ; calculate 4c + 4d
    ; look up value in table caltab for PHIN and put in RES34
    LT ONE
    MPYK caltab
    PAC
    ADD PHIN
    TBLR RES34

    ; negate for negative bank
    ; test for negative bank
    LAC UBANK
    BZ intcon
    ; negative bank (0,1)
    ; negate RES34
    ZAC
    SUB RES34
    SACL RES34

    intcon:

    ; calculate 3a and add to RES34
    ; add INTEGU to RES34
    LAC RES34
    ADD INTEGU
    SACL RES34

    ; calculate 4a + 4b = ((T2 - Tf)*2-1)*UFLUX + (T2-T1)*UFLUX)/2
    = (T1T2c*3 -1 - PHIN*2)*UFLUX/2
    LT THREE
    MPYK T1T2c
    PAC
    SUB ONE
    SUB PHIN,1
    SACL TEMP
    LT TEMP
    MPY UFLUX
    PAC
    SACL TEMP
    LAC TEMP,15
    SACH TEMP

    ; add to RES34
    LAC RES34
    ADD TEMP
    SACL RES34

    ; return
    B cal34u1
.end
.title "CAL34V"

; This routine performs the sample time calculations for V phase.
; i.e. it calculates equations 3 and 4 of the firing algorithm.

.mlib "mac.lib"

; CALLED FROM: PV2

; DATA:
   .ref ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, MINUS
   .ref PHIN, RES34
   .ref INTEGV, VFLUX, VBANK, TEMP, T1T2c

; ADDRESSES:
   .def cal34v
   .ref caltab, cal34v1

;(33)
; PROGRAM:
   cal34v:
   ; calculate 4c + 4d
   ; look up value in table caltab for PHIN and put in RES34
   LT ONE
   MPYK caltab
   PAC
   ADD PHIN
   TBLR RES34
   ; negate for negative bank
   ; test for negative bank
   LAC VBANK
   BZ intcon
   ; negative bank (0,1)
   ; negate RES34
   ZAC
   SUB RES34
   SACL RES34
   intcon:
   ; calculate 3a and add to RES34
   ; add INTEGV to RES34
   LAC RES34
   ADD INTEGV
   SACL RES34

   ; calculate 4a + 4b = ((T2 - Tf)*2-1)*VFLUX + (T2-T1)*VFLUX)/2 =
   (T1T2c*3 -1 - PHIN*2)*VFLUX/2
   LT THREE
   MPYK T1T2c
   PAC
   SUB ONE
   SUB PHIN, 1
   SACL TEMP
   LT TEMP
   MPY VFLUX
   PAC
   SACL TEMP
   LAC TEMP, 15
   SACH TEMP

   ; add to RES34
   LAC RES34
   ADD TEMP
   SACL RES34

   ; return
   B cal34v1
.end
.title "CMAINS"

;Measure mains period over eight cycles then adjust WDBASE and WDCOUNT
.mlib "mc.lib"

;CALLED FROM: MAIN

;UNCHANGED DATA:
.ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
.ref WDRES1,WDRES2

;CHANGED DATA:
.ref BTEMP1,BTEMP,TMP,TEMP,TEMP1,WDBASE,TEMP2
.ref WDCOUNT,WDADJ,PLL_IH

;CONSTANTS:
.ref MCYC_LOc,MCYC_HIc,T1T2c,T1T2_2c

;ADDRESSES:
.def cmains0
.def cmains1

;CONSTANTS IN MEMORY:
dlo: .word MCYC_LOc
dhi: .word MCYC_HIc

;PROGRAM:
cmains0:
;measure mains period over eight cycles (1,?)
;load MCYC_LOc into BTEMP and MCYC_HIc into BTEMP1 for later use
   c2: LT ONE
          MPYK dlo
          PAC
          TBLR BTEMP
          LT ONE
          MPYK dhi
          PAC
          TBLR BTEMP1
;clear cycle counter: TMP
          ZAC
          SAACL TMP
;clear time register: 32 bit: TEMP,TEMP1
          SAACL TEMP
          SAACL TEMP1
;clear error register WDBASE
          SAACL WDBASE
;wait for negative zero cross
;wait for '+'ve half cycle: do two tests separated by 1ms for noise rejection
;select bank 0
          BANK 0
;wait loop (1,?)
;wait for positive mains (1,?)
          pl IN TEMP2,0
          SBIT 8,TEMP2
          BZ pl
          ;delay 1.024ms
          LAC ONE,10
          de NOP
          NOP
          SUB ONE
          BNZ de
;retest for positive mains
          IN TEMP2,0
          SBIT 8,TEMP2
BZ pl
;wait for -'ve half cycle
nl IN TEMP2,0
SBIT 8,TEMP2
BNZ nl

;reset WD counter
BANK 1
OUT WDRES1,0
OUT WDRES2,0

;cycle loop (1,?)
;wait 14.3msec
c1: LAC ONE,15
SUB ONE,12
sd12 SUB ONE
BNZ sd12

;wait for negative zero cross
BANK 0
nl1 IN TEMP2,0
SBIT 8,TEMP2
BNZ nl1

;read WDT count. Previous reading is TEMP1, lower word of timer register
BANK 1
IN TEMP2,0
;test if counter has overflowed
ZALS TEMP2
SUBS TEMP1
BGZ ov1

;counter has overflowed (0,1)
;add 2^16 to timer register
INC TEMP

ov1:
;update previous reading
LAC TEMP2
SACL TEMP1
;test if WDT count is between MCYC_LOc and MCYC_HIc * cycle count
;load MCYC_LOc * cycle count
ZAC
LAR AR0,TMP
f1: ADDS BTEMP
BANZ f1

;subtract timer register
SUBS TEMP1
SUBH TEMP
BGZ f3

;result > 0 (0,1)#
;WDT count outside limits
;negative result #(0,1)
;load MCYC_HIc * cycle count
ZAC
LAR AR0,TMP
f2: ADDS BTEMP1
BANZ f2

;subtract timer register
SUBS TEMP1
SUBH TEMP
BGEZ f4

;result < 0 (0,1)
;WDT count outside limits
;WDT count outside limits (0,1)
;indicate error and set cycle count to 7 to end loop
f3: LAC SEVEN
SACL WDBASE
SACL TMP
f4:
  ;increment cycle counter, TMP
  LAC TMP
  ADD ONE
  SACL TMP
  ;end loop if cycle count = 8
  SUB ONE,3
  BLZ c1
  ;end loop if no error
  LAC WDBASE
  BNZ c2
  ;adjust WDCOUNT
  ;divide total time in time register by 8 with round-off and put in WDCOUNT
  ;round off
  ZALH TEMP
  ADDS TEMP1
  ADD ONE,2
  SACL TEMP1
  SACH TEMP
  ;extract lower 3 bits of upper word, TEMP, left justified
  LAC TEMP,13
  SACL TEMP
  ;extract upper 13 bits of lower word, TEMP1, right justified
  LAC TEMP1,13
  SACH TEMP1
  LAC MINUS,13
  XOR MINUS
  AND TEMP1
  ;add together and put into WDCOUNT
  ADD TEMP
  SACL WDCOUNT
  ;adjust WDBASE, WDADJ and PLL_IH
  ;divide WDCOUNT by # samples per mains cycle, T1T2c * 2
  LACK T1T2_2c
  SACL TEMP
  ZALS WDCOUNT
  LARK 0,15
  div SUBC TEMP
  BANZ div
  ;store quotient in WDBASE, remainder in TEMP1
  SACH TEMP1
  SACL WDBASE
  ;divide remainder by T1T2c*2 to get fractional part
  ZALH TEMP1
  LARK 0,15
  div1 SUBC TEMP
  BANZ div1
  ;store fractional part, shifted right by 4, into WDADJ and PLL_IH
  SACL TEMP1
  LAC TEMP1,12
  SACH TEMP1
  LAC MINUS,12
  XOR MINUS
  AND TEMP1
  SACL WDADJ
  SACL PLL_IH

;return
  B cmains1
.end
.title "COMMU"

; This module is responsible for firing the U phase TRIACS.

.mlib "mac.lib"

; CALLED FROM: PU2,SAMPTIM

; DATA:
; .ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
; .ref TRIAC,TEMP,PULSE,UPULSE

; ADDRESSES:
; .def commu
; .ref commul

;(20)
; PROGRAM:

commu:
commu:
; send bits 0,1 of triac control word to ports P0,P1
  LAC TRIAC
  XOR MINUS
  SA CL TEMP
  LAC THREE
  AND TEMP
  SA CL TEMP
  BANK 0
  OUT TEMP,3
; set pulsewidth timer, UPULSE, to PULSE
  LAC PULSE
  SA CL UPULSE
; output test data
;  LACK 086H
;  SA CL TEMP
;  LAC TEMP,8
;  SA CL TEMP
;  ZAC
;  TBLW TEMP
; return
  B commul

.end
.title "COMMV"

; This module is responsible for firing the V phase TRIACS.

.mlib "mac.lib"

; CALLED FROM: PV2, SAMPTIM

; DATA:
  .ref ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, MINUS
  .ref TRIAC, TEMP, PULSE, VPULSE

; ADDRESSES:
  .def commv
  .ref commv1

;(20)
; PROGRAM:
commv:
; send bits 2,3 of triac control word to ports P2,P3
LAC TRIAC
XOR MINUS
SACL TEMP
LAC THREE,2
AND TEMP
SACL TEMP
BANK 0
OUT TEMP,3
; set pulsewidth timer, VPULSE, to PULSE
LAC PULSE
SACL VPULSE
; return
B commv1

.end
.title "IMAINS"

; Interrupt routine for handling mains negative zero cross interrupt.
; The number of watchdog timer counts since the start of the mains cycle is
; stored in MAINPH register for use of the phase locked loop.
; The interrupt mask is enabled to prevent further interrupts
.mlib "mac.lib"

; CALLED FROM: MAIN

; UNCHANGED DATA:
   .ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
   .ref WDCOUNT

; CHANGED DATA:
   .ref TEMPT,MAINPH

; ADDRESSES:
   .def imains0
   .ref imains1

; PROGRAM:
imains0:
; test if mains is negative
; test if port 8 is low
   BANK 0
   IN TEMPT,0
   SBIT 8,TEMPT
   BNZ p1
; mains is negative (0,1)
; read watchdog count and put into TEMPT
   BANK 1
   IN TEMPT,0
; MAINPH = TEMPT + WDCOUNT
   ZALS WDCOUNT
   ADD TEMPT
   SACL MAINPH
; set interrupt mask
   IN TEMPT,5
   SBIS 8,TEMPT
   OUT TEMPT,5
p1:
; clear interrupt flag
   BANK 0
   LAC ONE,8
   SACL TEMPT
   OUT TEMPT,6
; clear FIFO-0 to allow capture interrupts
   BANK 6
   IN TEMPT,0
   IN TEMPT,0
; return to main routine
   B imains1

.end
.title "ISPEED"
; This is an interrupt routine that runs every count of the speed v/f converter
; output. It increments the speed counter, TS, by 2.

.mlib "mac.lib"

; CALLED FROM: MAIN

; UNCHANGED DATA:
 .ref ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, MINUS

; CHANGED DATA:
 .ref TEMPT, TS

; ADDRESSES:
 .def ispeed0
 .ref ispeed0

; PROGRAM:
 ispeed0:
 ; update speed counter, TS, by 2
  ZALS TS
  ADD TWO
  SACL TS
 ; clear FIFO-1 to allow capture interrupts
  BANK 6
  IN TEMPT, 1
  IN TEMPT, 1
  IN TEMPT, 1
  IN TEMPT, 1
 ; clear fifo full flag for 1
  LAC ONE, 7
  SACL TEMPT
  OUT TEMPT, 5
 ; clear interrupt flag
  BANK 0
  LAC ONE, 9
  SACL TEMPT
  OUT TEMPT, 6
 ; return to main routine
  B ispeed1

.end
.title "PACE"

Limits acceleration to value in ACCEL and deceleration to value in DECEL

.mlib "mac.lib"

;CALLED FROM: SAMPTIM

;INPUT DATA:
; TSREAD : TS counter reading at start of interval
; SCOUNT : previous TSREAD value
; SPIN  : input speed
; SPDNEW : actual speed
; ACCEL : acceleration rate and deceleration rate

;OUTPUT DATA:
; SPDNEW

;LOCAL DATA:
; STIME  : half cycle counter for v/f sampling

;DATA:
 .ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
 .ref SPIN,SPDNEW,ACCEL,BTEMP
 .ref TSREAD,SCOUNT,STIME

;ADDRESSES:
 .def speed
 .ref speed1

;PROGRAM:
speed:
timing and speed v/f sampling
decrement counter
LAC STIME
SUB ONE
SACL STIME
test counter = 0
LAC STIME
BNZ A2
;counter = 0 (0,1)
;set counter to 10 for next count down
LACK 10
SACL STIME
subtract previous count from count & store in SPIN
LAC TSREAD
SUB SCOUNT
SACL SPIN
clip spin to 255
ZALS SPIN
SUB ONE,8
BLZ A6
LACK 255
SACL SPIN
A6:
;update previous count
LAC TSREAD
SACL SCOUNT
;update SPDNEW below:
test SPDNEW < SPIN*2^8
ZALS SPDNEW
SUB SPIN,8
BGEZ A5
;SPDNEW < SPIN*2^8 (0,1)#
;accelerate routine
B Celer

; SPDNEW >= SPIN*2^8 #(0,1)
; test SPDNEW = SPIN
A5:  BZ A1
    ; SPDNEW > SPIN (0,1)
    ; decelerate routine
    B DC
A1:  A2:
    ; return
    B speed1
    ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
CELER:
    ; accelerate routine
    ; SPDNEW = SPDNEW + #ACCEL
    ZALS SPDNEW
    ADDS ACCEL
    SACL SPDNEW
    ; test for carry
    SUBS SPDNEW
    BZ B1
    ; carry set (0,1)
    ; SPDNEW = 255*2^8
    LAC MINUS,8
    SACL SPDNEW
    ; test for SPDNEW >= SPIN*2^8
B1  ZALS SPDNEW
    SUB SPIN,8
    BLZ B2
    ; SPDNEW >= SPIN*2^8 (0,1) #
    ; set SPDNEW = SPIN*2^8
    LAC SPIN,8
    SACL SPDNEW
    ; end
B2:  B A1
    ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
DC:
    ; decelerate routine
    ; SPDNEW = SPDNEW - #ACCEL
    ZALS SPDNEW
    SUBS ACCEL
    SACL SPDNEW
    ; test for carry
    SUBS SPDNEW
    BZ B4
    ; carry set (0,1)
    ; SPDNEW = 0
    ZAC
    SACL SPDNEW
    ; test for SPDNEW < SPIN*2^8
B4  ZALS SPDNEW
    SUB SPIN,8
    BGEZ B3
    ; SPDNEW < SPIN*2^8 (0,1)
    ; set SPDNEW to SPIN*2^8
    LAC SPIN,8
    SACL SPDNEW
    ; end
B3:  B A1
    .end
This program implements the actual phase lock loop. It calculates a new inverter period from the input phase error. The input phase, PLL_P, is calculated from MAINPH and MAINPER then set to zero if outside $\pm$ PLL_ERRc. If it is outside the limits for two cycles, a flag is set to restart the PLL loop. The output period is clamped to the allowed limits by putting the accumulator into saturation mode. The loop natural frequency (it is a second order loop) is PLL_NFc in units of (output frequency)/250. PLL_NFc can only have values from 1 to 10. The loop damping factor is set to 0.707. For more details, see the author's research notes. For PLL_NFc set to 1, the natural loop frequency will be 0.2Hz for an output frequency of 50Hz. The output is WDADJ.

Internal Registers
- PLL_IH : upper 16 bits, 32 bit integrator output
- PLL_IL : lower 16 bits, 32 bit integrator output

.mlib "mac.lib"

; CALLED FROM: BACK

;UNCHANGED DATA:
.ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
.ref MAINPER

;CHANGED DATA:
.ref BTEMP,PLL_P,BTEMP1,PLL_IL,PLL_IH,WDADJ,BFLAGS,MAINPH

;CONSTANTS:
.ref PLL_ERRc,PLL_NFc,PLL_NF1c,PLL_NF2c

;ADDRESSES:
.def pll0
.def pll1

;PROGRAM:
pll0:
; save status
SST BTEMP
; set overflow mode to saturating
SOVM
; test if phase was out of limits last cycle - test if BFLAGS.5 set
SBIT 5,BFLAGS
BZ b1
; BFLAGS.5 set (0,1)
; set BFLAGS.4 to indicate the PLL has lost control
SBIS 4,BFLAGS
b1:
; calculate input phase: PLL_P = MAINPH - MAINPER/2
ZALH MAINPH
SUB MAINPER,15
SACH PLL_P
; clip input phase to $\pm$ PLL_ERRc
; test if PLL_P > PLL_ERRc
LT ONE
MPYK PLL_ERRc
PAC
SUB PLL_P
BGEZ c1
;PLL_P > PLL_ERRc (0,1)
B c3
;PLL_P <= PLL_ERRc #(0,1)
ADD PLL_P,1
BGEZ c2
;PLL_P < -PLL_ERRc (0,1)
SACL PLL_P
SBIS 5,BFLAGS
ZAC

ADD PLL_P,1
BGEZ c2
;PLL_P < -PLL_ERRc (0,1)
set PLL_P to zero and set BFLAGS.5

LAC ONE,5
SACL BTEMP1
LT BTEMP1
MPYK PLL_NF1c
PAC
SACL BTEMP1
;multiply PLL_P by 2^6 for later use
LAC PLL_P,6
SACL PLL_P

;integral part. The integrator output is stored in the 32 bit variable
PLL_I [PLL_IH, PLL_IL]. PLL_I,[Q24] = PLL_NF2c^2 * 29 * PLL_P * 2^6 + PLL_I. PLL_NF2c = PLL_NF2c^2 * 29
LT PLL_P
ZALS PLL_IL
ADDH PLL_IH
MPYK PLL_NF2c
APAC
SACL PLL_IL
SACH PLL_IH

;proportional part & output. WDADJ,[Q8] = (52*2^5*PLL_NFc * PLL_P*2^6 + PLL_I)/2^16
MPY BTEMP1
APAC
SACH WDADJ

;set MAINPH to zero so that an error occurs if no zero cross interrupt occurs
ZAC
SACL MAINPH
;clear FIFO-0 to allow capture interrupts
BANK 6
IN BTEMP1,0
IN BTEMP1,0
IN BTEMP1,0
IN BTEMP1,0
;clear fifo full flag for 0
LAC ONE,3
SACL BTEMP1
OUT BTEMP1,5
;clear mains interrupt flag and clear mains interrupt mask
BANK 0
LAC ONE,8
SACL BTEMP1
OUT BTEMP1,6
IN BTEMP1,5
SBIC 8,BTEMP1
OUT   BTEMP1,5
;restore status
LST   BTEMP
;return
B     pll1
.end
.title "PU1"

; This routine initialises variables for the following integration period.
; Also, UBOOST is subtracted from UFLUX and UBOOST is zeroed.

.mlib "mac.lib"

; CALLED FROM: SAMPTIM

; DATA:
. ref ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, MINUS
. ref INTEGU, UFLAG
. ref PHIN, TSUFL, UFLUX
. ref PHOUT, UBOOST, PHUB, OUTSUB
. ref OUTSUM, FLAGS, FLUXMAX

; ADDRESSES:
. def pu1

;(32)
; PROGRAM:
pu1:
; enable endstops
SBIC 7, FLAGS
; start integration of UFLUX - load half UFLUX into INTEGU for trapezoidal integration
LAC UFLUX, 15
SACH INTEGU
; store values of PHOUT and OUTSUM for use in background calculations
LAC PHOUT
SACL PHUB
LAC OUTSUM
SACL OUTSUB
; flag stage 2 for U phase
LACK 2
SACL UFLAG
; correct UFLUX for boost - subtract UBOOST from UFLUX
LAC UFLUX
SUB UBOOST
SACL UFLUX
; test & correct for overflow (10)
; test for negative overflow - test if UFLUX < -FLUXMAX
LAC UFLUX
ADD FLUXMAX
BGEZ pu1h
; UFLUX < -FLUXMAX (0, 1)
; set UFLUX to -FLUXMAX
ZAC
SUB FLUXMAX
SACL UFLUX
B pula
; UFLUX >= -FLUXMAX # (0, 1)
; test for positive overflow - test if UFLUX > FLUXMAX
pu1h: LAC UFLUX
SUB FLUXMAX
BLEZ pula
; UFLUX > FLUXMAX (0, 1)
; set UFLUX to FLUXMAX
LAC FLUXMAX
SACL UFLUX
; reset UBOOST to 0
pu1a: ZAC
SACL UBOOST
; store starting flux value for use in calculations
LAC UFLUX
SACL TSUFL
;return to sample interrupt routine
pulg: RET
.end
This routine adjusts the calculations for a bank change.

```
.title "PU1B"

;CALLED FROM: PU2

;DATA:
  .ref PRECU,T1T2c,TEMP,TSUFL,FUT1,UBANK,SIX

;ADDRESSES:
  .def pulb

;(16)
;PROGRAM:
pu1b:
  ;test if bank positive - test if UBANK = 0
  LAC UBANK
  BNZ ab1
  ;bank positive (0,1)
  ;output test data
  LACK 084H
  SACL TEMP
  LAC TEMP,8
  SACL TEMP
  LAC SIX
  TBLW TEMP
  B ab2
ab1:
  ;bank not positive #(0,1)
  ;output test data
  LACK 083H
  SACL TEMP
  LAC TEMP,8
  SACL TEMP
  LAC SIX
  TBLW TEMP
ab2:
  ;add 2K(t2-t1)(int. Vo at t1 - flux ref. at t1) to PRECU to correct stabilising term
  ;add 2K(t2-t1)(int. Vo at t1)
  LAC PRECU
  LT TSUFL
  MPYK T1T2c
  APAC
  ;subtract 2K(t2-t1)(flux ref. at t1)
  LT FUT1
  MPYK T1T2c
  SPAC
  SACL PRECU
  ;negate TSUFL and FUT1 in case there are further bank changes this sample time
  ZAC
  SUB TSUFL
  SACL TSUFL
  ZAC
  SUB FUT1
  SACL FUT1
  ;return
  RET

.end
```
.title "PU2"
;U PHASE STAGE 2 ROUTINE
;This routine controls the firing of the triacs at the correct time.
;It also calculates the current ripple for determining the bank
switching time.
.mlilb "mac.lib"

;CALLED FROM: SAMPTIM

;DATA:
    .ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
    .ref PHOUT,TEMP,UFLUX,UBANK,T1T2c,OUTPER4c
    .ref FLAGS,BFLAGS,INTEGU,UMASK,RES34
    .ref PHIN,UFLAG,PRECU,TMP,FLUXREF
    .ref TRIAC,UBOOST,SPDOLD,VFLAG,VX4

;ADDRESSES:
    .def pu2,cal34u1,commu1
    .ref endref,commu,pulb,cal34u,pu21

;(147)
;PROGRAM:
pu2:
;update integral of flux
    LAC UFLUX
    ADD INTEGU
    SACL INTEGU
;check if background calcs finished
    SBIT 0,BFLAGS
    BNZ endpu2
;background finished (0,1)
;check for current flow: check if one of TSD1,TSD2 inputs is high
    BANK 0
    IN TEMP,0
    LAC THREE,4
    AND TEMP
    BZ noflow
;current flow (0,1)#
;check if endstop disabled
    SBIT 6,FLAGS
    BNZ trig
;endstop disabled [false current flow reading] (0,1)#
;triac trigger routine
;enstop enabled #(0,1)
;check if endstop reached
    LACK 40
    SUB PHIN
    BGEZ trig1
;endstop reached (0,1)#
;output test data
    LACK 082H
    SACL TEMP
    LAC TEMP,8
    ADD PHIN
    SACL TEMP
    LAC ONE,3
    ADD SIX
    TBLW TEMP
    TBLW PRECU
    TBLW RES34
    ;flag stage 3 program
    LAC THREE
    SACL UFLAG
    ;indicate V phase fired first - clear FLAGS.2
    SBIC 2,FLAGS
; commutate triacs
B commu
B endpu2
; endstop not reached #(0,1)
; secondary triac trigger routine, trig1
; no current flow #(0,1)
; set endstop disable flag
noflow: SBIS 6,FLAGS
; check for bank change
; calculate flux ripple
; find flux reference in look-up table
LAC FLUXREF
ADD PHOUT
TBLR TEMP
; scale for correct voltage - multiply by volts X 4 then divide by 2^12
LT VX4
MPY TEMP
PAC
SACH TEMP,4
; subtract flux reference and boost from flux to get flux ripple and store in TEMP
LAC UFLUX
SUB TEMP
SUB UBOOST
SACL TEMP
; check sign of present bank
LAC UBANK
BNZ negbnk
; positive bank (0,1)
; test for bank change - test if TEMP > 1
LAC TEMP
SUB ONE
BLEZ trig
B bnkchg
; negative bank (0,1)
; test for bank change - test if TEMP < -1
negbnk: LAC TEMP
ADD ONE
BGEZ trig
B bnkchg
; bank change (0,1)
; turn off U triacs
bnkchg: BANK 0
OUT UMASK,2
; change triac to be fired
LAC UMASK
XOR TRIAC
SACL TRIAC
; change bank indicator
LAC ONE
XOR UBANK
AND ONE
SACL UBANK
; adjust pre-calculations for bank change
CALL pu1b
; end here to give triac time to recover
B endpu2 ; end stage 2 program

; no bank change #(0,1)
; triac trigger routine

trig:
; triac trigger routine
    ; find maximum allowed value of PHIN [TEMP] for safe triac triggering
;look up flux table at PHOUT + OUTPER4c, to calculate back
e.m.f.
LACK OUTPER4c
ADD PHOUT
ADD FLUXREF
TBLR TEMP
;test bank direction
LAC UBANK
BZ negtst
;negative bank (0,1)
;negate table reading
ZAC
SUB TEMP
SACL TEMP
;test if negative
negtst: LAC TEMP
BGEZ noneg
;negative (0,1)
;set max PHIN to 40 and put in TEMP
LACK 40
SACL TEMP
B pht
noneg:
;not negative (0,1)
;scale for correct voltage - multiply by volts X 4 then divide by \(2^{12}\)
LT VX4
MPY TEMP
PAC
SACH TEMP,4
;multiply by SPDOLD & discard lower byte to get back emf
LAC SPDOLD,15
SACH TMP
SBIC 15,TMP
LT TEMP
MPY TMP
PAC
SACH TEMP,1
;table lookup at ENDREF+TEMP, put result in TEMP
LT ONE
MPYK endref
PAC
ADD TEMP
TBLR TEMP
;test PHIN > TEMP
pht: LAC PHIN
SUB TEMP
BLEZ pu2c
;PHIN > TEMP (0,1)
;flag stage 3
LACK 3
SACL UFLAG
;check if V phase triggered
LAC VFLAG
SUB THREE
BNZ pnt
;V phase triggered (0,1)
;clear FLAGS.2
SBIC 2,FLAGS
pnt:
B endpu2
;PHIN <= TEMP (0,1)
trig1:
;calculate equations 3 & 4. Result is placed in RES34.
pu2c: B cal34u
cal34u1:
;add to precalc and store in TMP
LAC RES34
ADD PRECU
SACL TMP
;test for bank direction
LAC UBANK
BNZ pu2h
;positive bank (0,1)
;test sign of TMP
LAC TMP
BGZ endpu2
;sign negative (0,1)
;output test data
LACK 082H
SACL TEMP
LAC TEMP,8
ADD PHIN
SACL TEMP
LAC ONE,3
ADD SIX
TBLW TEMP
TBLW PRECU
TBLW RES34
;flag stage 3
LACK 3
SACL UFLAG
;check if V phase triggered
LAC VFLAG
SUB THREE
BNZ pn2
;V phase triggered (0,1)
;clear FLAGS.2
SBIC 2,FLAGS
pn2:
;commutate triac
B commu
B endpu2
;negative bank #(0,1)
;test sign
pu2h: LAC TMP
BLZ endpu2
;sign positive (0,1)
;output test data
LACK 082H
SACL TEMP
LAC TEMP,8
ADD PHIN
SACL TEMP
LAC ONE,3
ADD SIX
TBLW TEMP
TBLW PRECU
TBLW RES34
;flag stage 3
LACK 3
SACL UFLAG
;check if V phase triggered
LAC VFLAG
SUB THREE
BNZ pn3
;V phase triggered (0,1)
;clear FLAGS.2
SBIC 2,FLAGS
pn3:
;commutate triac
B commu
;return
commul:
endpu2: B pu21
.end
.title "PV1"

; This routine initialises variables for the following integration period.
; Also, VBOOST is subtracted from VFLUX and VBOOST is zeroed.

.mlib "mac.lib"

; CALLED FROM: SAMPTIM

; DATA:
;  .ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
;  .ref INTEGV,VFLAG
;  .ref PHIN,TSVFL,VFLUX
;  .ref PHV,VBOOST,PHVB,OUTSVB
;  .ref OUTSUM,FLAGS,FLUXMAX

; ADDRESSES:
; .def pv1

;(32)
; PROGRAM:
pv1:
; enable endstops
SBIC 7,FLAGS
; start integration of VFLUX - load half VFLUX into INTEGV for trapezoidal integration
LAC VFLUX,15
SACH INTEGV
; store values of PHV and OUTSUM for use in background calculations
LAC PHV
SACL PHVB
LAC OUTSUM
SACL OUTSVB
; flag stage 2 for V phase
LACK 2
SACL VFLAG
; correct VFLUX for boost - subtract VBOOST from VFLUX
LAC VFLUX
SUB VBOOST
SACL VFLUX
; test & correct for overflow (10)
; test for negative overflow - test if VFLUX < -FLUXMAX
LAC VFLUX
ADD FLUXMAX
BGEZ pv1h
; VFLUX < -FLUXMAX (0,1)
; set VFLUX to -FLUXMAX
ZAC
SUB FLUXMAX
SACL VFLUX
B pv1a
; VFLUX >= -FLUXMAX (0,1)
; test for positive overflow - test if VFLUX > FLUXMAX
pv1h: LAC VFLUX
SUB FLUXMAX
BLEZ pv1a
; VFLUX > FLUXMAX (0,1)
; set VFLUX to FLUXMAX
LAC FLUXMAX
SACL VFLUX
; reset VBOOST to 0
pv1a: ZAC
SACL VBOOST
; store starting flux value for use in calculations
LAC VFLUX
SACL TSVFL
;return to sample interrupt routine
pvlg: RET
.end
.title "PV1B"

;This routine adjusts the calculations for a bank change

.mlib "mac.lib"

;CALLED FROM: PV2

;DATA:
 .ref PRECV,T1T2c,TEMP,TSVFL,FVT1,VBANK,FIVE

;ADDRESSES:
 .def pv1b

;(16)
;PROGRAM:
pv1b:
 ;test if bank positive - test if VBANK = 0
 LAC VBANK
 BNZ ab1
 ;bank positive (0,1)#
 ;output test data
 LACK 084H
 SACL TEMP
 LAC TEMP,8
 SACL TEMP
 LAC FIVE
 TBLW TEMP
 B ab2
 ab1:
 ;bank not positive #(0,1)
 ;output test data
 LACK 083H
 SACL TEMP
 LAC TEMP,8
 SACL TEMP
 LAC FIVE
 TBLW TEMP

ab2:
 ;add 2K(t2-t1)(int. Vo at t1 - flux ref. at t1) to PRECV to correct
 ;stabilising term
 ;add 2K(t2-t1)(int. Vo at t1)
 LAC PRECV
 LT TSVFL
 MPYK T1T2c
 APAC
 ;subtract 2K(t2-t1)(flux ref. at t1)
 LT FVT1
 MPYK T1T2c
 SPAC
 SACL PRECV
 ;negate TSVFL and FVT1 in case there are further bank changes this
 ;sample time
 ZAC
 SUB TSVFL
 SACL TSVFL
 ZAC
 SUB FVT1
 SACL FVT1
 ;return
 RET

.end
.title "PV2"
;V PHASE STAGE 2 ROUTINE
;This routine controls the firing of the triacs at the correct time.
;It also calculates the current ripple for determining the bank
switching time.

.mlib "mac.lib"

;CALLED FROM: SAMPTIM

;DATA:
.ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS
.ref PHV,TEMP,VFLUX,VBANK,T1T2c,OUTPER4c
.ref FLAGS,BFLAGS,INTEGV,VMASK,RES34
.ref PHIN,VFLAG,PRECV,TMP,FLUXREF
.ref TRIAC,VBOOST,SPDOLD,UFLAG,VX4

;ADDRESSES:
.def pv2,cal34v1,commv1
.ref endref,commv,pv1b,cal34v,pv21

;(147)
;PROGRAM:

pv2:
;update integral of flux
LAC VFLUX
ADD INTEGV
SACL INTEGV

;check if background calcs finished
SBIT 1,BFLAGS
BNZ endpv2

;background finished (0,1)
;check for current flow : check if one of TSD3,TSD4 inputs is high
BANK 0
IN TEMP,0
LAC THREE,6
AND TEMP
BZ noflow

;current flow (0,1)#
;check if endstop disabled
SBIT 7,FLAGS
BNZ trig

;endstop disabled [false current flow reading] (0,1)#
;triac trigger routine
;endstop enabled #(0,1)
;check if endstop reached
LACK 40
SUB PHIN
BGEZ trig1

;endstop reached (0,1)#
;output test data
LACK 082H
SACL TEMP
LAC TEMP,8
ADD PHIN
SACL TEMP
LAC ONE,3
ADD FIVE
TBLW TEMP
TBLW PRECV
TBLW RES34

;flag stage 3 program
LAC THREE
SACL VFLAG
;indicate U phase fired first - set FLAGS.2
SBIS 2,FLAGS
; commutate triacs
B commv
B endpv2
; endstop not reached #(0,1)
; secondary triac trigger routine, trig1
; no current flow #(0,1)
; set endstop disable flag
noflow: SBIS 7,FLAGS
; check for bank change
; calculate flux ripple
; find flux reference in look-up table
LAC FLUXREF
ADD PHV
TBLR TEMP
; scale for correct voltage - multiply by volts X 4 then
divide by 2^12
LT VX4
MPY TEMP
PAC
SACH TEMP,4
; subtract flux reference and boost from flux to get flux
ripple and store in TEMP
LAC VFLUX
SUB TEMP
SUB VBOOST
SACL TEMP
; check sign of present bank
LAC VBANK
BNZ negbnk
; positive bank (0,1)
; test for bank change - test if TEMP > 1
LAC TEMP
SUB ONE
BLEZ trig
B bnkchg
; negative bank (0,1)
; test for bank change - test if TEMP < -1
negbnk: LAC TEMP
ADD ONE
BGEZ trig
B bnkchg
; bank change (0,1)
; turn off V triacs
bnkchg: BANK 0
OUT VMASK,2
; change triac to be fired
LAC VMASK
XOR TRIAC
SACL TRIAC
; change bank indicator
LAC ONE
XOR VBANK
AND ONE
SACL VBANK
; adjust pre-calculations for bank change
CALL pvib
; end here to give triac time to recover
B endpv2 ; end stage 2 program

; no bank change #(0,1)
; triac trigger routine

trig:
; triac trigger routine
; find maximum allowed value of PHIN [TEMP] for safe triac
triggering
; look up flux table at PHV + OUTPER4c, to calculate back e.m.f.
LACK OUTPER4c
ADD PHV
ADD FLUXREF
TBLR TEMP
; test bank direction
LAC VBANK
BZ negtst
; negative bank (0,1)
; negate table reading
ZAC
SUB TEMP
SACL TEMP
; test if negative
negtst: LAC TEMP
BGEZ noneg
; negative (0,1)
; set max PHIN to 40 and put in TEMP
LACK 40
SACL TEMP
B pht
noneg:
; not negative (0,1)
; scale for correct voltage - multiply by volts X 4 then divide by 2^12
LT VX4
MPY TEMP
PAC
SACH TEMP,4
; multiply by SPDOLD & discard lower byte to get back emf
LAC SPDOLD,15
SACH TMP
SBIC 15, TMP
LT TEMP
MPY TMP
PAC
SACH TEMP,1
; table lookup at ENDREF+TEMP, put result in TEMP
LT ONE
MPYK endref
PAC
ADD TEMP
TBLR TEMP
; test PHIN > TEMP
pht: LAC PHIN
SUB TEMP
BLEZ pv2c
; PHIN > TEMP (0,1)
; flag stage 3
LACK 3
SACL VFLAG
; check if U phase triggered
LAC UFLAG
SUB THREE
BNZ pnt
; U phase triggered (0,1)
; set FLAGS.2
SBIS 2(FLAGS
pnt: B endpv2
; PHIN <= TEMP # (0,1)
trig1:
; calculate equations 3 & 4. Result is placed in RES34.
pv2c: B cal34v
cal34v1:
; add to precalc and store in TMP
LAC RES34
ADD PRECV
SACL TMP
;test for bank direction
LAC VBANK
BNZ pv2h
;positive bank (0,1)#
;test sign of TMP
LAC TMP
BGZ endpv2
;sign negative (0,1)
;output test data
LACK 082H
SACL TEMP
LAC TEMP,8
ADD PHIN
SACL TEMP
LAC ONE,3
ADD FIVE
TBLW TEMP
TBLW PRECV
TBLW RES34
;flag stage 3
LACK 3
SACL VFLAG
;check if U phase triggered
LAC UFLAG
SUB THREE
BNZ pn2
;U phase triggered (0,1)
;set FLAGS.2
SBIS 2,FLAGS
pn2:
;commutate triac
B commv
B endpv2
;negative bank #(0,1)
;test sign
pv2h: LAC TMP
BLZ endpv2
;sign positive (0,1)
;output test data
LACK 082H
SACL TEMP
LAC TEMP,8
ADD PHIN
SACL TEMP
LAC ONE,3
ADD FIVE
TBLW TEMP
TBLW PRECV
TBLW RES34
;flag stage 3
LACK 3
SACL VFLAG
;check if U phase triggered
LAC UFLAG
SUB THREE
BNZ pn3
;U phase triggered (0,1)
;set FLAGS.2
SBIS 2,FLAGS
pn3:
;commutate triac
B commv
;return
commv1:
endpv2: B pv21
.end
.title "READ"

; This routine inputs the values in the motor flux counters, transforms them into flux changes over the previous sample time and adds these to the flux registers, correcting for overflows.

.mlib "mac.lib"

; CALLED FROM: SAMPTIM

; DATA:
.ref ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, MINUS
.ref UCOUNT, VCOUNT, UFLUX, VFLUX, TEMP, TEMP1
.ref FLUXMAX

; ADDRESSES:
.def read

;(42)

; PROGRAM:
read:

; extract flux value for U phase
; find current flux counter reading: read timer 1
    BANK 2
    IN TEMP, 0
; subtract previous count, UCOUNT
    LAC TEMP, 8
    SUB UCOUNT, 8
    SA CL TEMP1
; update previous count, UCOUNT, with new count
    LAC TEMP
    SAC L UCOUNT

; calculate new motor flux for U phase
; add flux change to previous flux value
    ZALH UFLUX
    ADD TEMP1, 8
    SACH UFLUX

; check for negative flux overflow: check if < -FLUXMAX
    LAC UFLUX
    ADD FLUXMAX
    BGEZ bst5

; negative overflow (0,1)
; limit to -FLUXMAX
    ZAC
    SUB FLUXMAX
    SACL UFLUX
    B bst4

; not negative overflow #(0,1)
; check for positive overflow: check if > FLUXMAX
bst5: SUB FLUXMAX, 1
    BLEZ bst4
; positive overflow (0,1)
; limit to FLUXMAX
    LAC FLUXMAX
    SACL UFLUX

bst4: BANK 2
    IN TEMP, 2
; subtract previous count, VCOUNT
    LAC TEMP, 8
    SUB VCOUNT, 8
    SAC L TEMP1
; update previous count, VCOUNT, with new count
    LAC TEMP
SACL VCOUNT
;calculate new motor flux for V phase
;add flux change to previous flux value
ZALH VFLUX
ADD TEMP1,8
SACH VFLUX
;check for negative flux overflow: check if < -FLUXMAX
LAC VFLUX
ADD FLUXMAX
BGEZ rdv5
;negative overflow (0,1)#
;limit to -FLUXMAX
ZAC
SUB FLUXMAX
SACL VFLUX
B rdv4
;not negative overflow #(0,1)
;check for positive overflow: check if > FLUXMAX
rdv5: SUB FLUXMAX,1
BLEZ rdv4
;positive overflow (0,1)
;limit to FLUXMAX
LAC FLUXMAX
SACL VFLUX
;return
rdv4: RET
.end
This routine is the most important routine of the cycloconverter software and is called every sample time interrupt. The routine controls the sample time interrupts, directing program flow to the correct routine. It also takes care of house-keeping required for the software to operate correctly i.e. increment input and output phase, control the source of the interrupt etc.

.mlib "mac.lib"

;CALLED FROM: MAIN

;DATA:
  .ref ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, MINUS
  .ref UCOUNT, VCOUNT, TEMP, UPULSE, VPULSE
  .ref PHIN, SPDOLD, SPDNEW, OUTSUM, PHOUT, PHV, UFLAG, VFLAG
  .ref FLAGS, COUNT, BFLAGS, TS
  .ref SPDTMP, UBANK, VBANK, TRIAC, UFLUX, VFLUX, FLUXREF, VX4, MAINPER
  .ref WDREM, WDADJ, WDBASE, WDCOUNT, WDPER, UVFREM, UVFSET, VVFREM, VVFSET
  .ref TSREAD, TAL, TAH, TBSR, SAH, SAL, SBSR, STREG, TSTAT, SSTAT
  .ref OUTPERc, OUTPER4c, T1T2c

;ADDRESSES:
  .def samptim1, samptim2, pu21, pv21
  .ref pv1, pv2, read, boost
  .ref back, pu1, pu2, flxref

;VARIOUS:
EXT .set 0

;PROGRAM:
samptim1:
  ;save registers
  LAC TSTAT
  SACL SSTAT
  LAC TBSR
  SACL SBSR
  LAC TAL
  SACL SAL
  LAC TAH
  SACL SAH
  MPYK 1
  PAC
  SACL STREG
  ;set overflow mode to unsaturating
  ROVM
  ;set next sample period
  ;add WDADJ * 2^4 to WDREM
  ZALS WDREM
  ADD WDADJ, 4
  SACL WDREM
  ;add WDBASE to upper word of ACC to get new period
  ADDH WDBASE
  ;put new period - 1 into WPER register of WD timer
  SUBH ONE
  SACH TEMP
  BANK 1
  OUT TEMP, 1
  ;update count for mains cycle to start of present sample period,
  WDCOUNT
  ;add count for last sample period, WDPER, to WDCOUNT
  LAC WDCOUNT
  ADD WDPER
  SACL WDCOUNT
;update WDPER - load in WTPL register + 1
BANK 1
IN   WDPER,2
INC WDPER
;clobber WDT interrupt flag
LAC ONE,1
SACL TEMP
BANK 0
OUT   TEMP,6
;enable interrupts
EINT
;correct offset in U v/f counter: UCOUNT = UCOUNT + WDPER * UVFSET + UVFREM
ZALS UVFREM
ADDH UCOUNT
LT  UVFSET
MPY WDPER
APAC
SACH UCOUNT
SACL UVFREM
;correct offset in V v/f counter: VCOUNT = VCOUNT + WDPER * VVFSET + VVFREM
ZALS VVFREM
ADDH VCOUNT
LT  VVFSET
MPY WDPER
APAC
SACH VCOUNT
SACL VVFREM
;check for mains half cycle: end of half cycle flag is set
SBIT 5,FLAGS
BZ x2
;set: mains half cycle (0,1)#
;execute samptim2 routine
B samptim2
x2:
;not mains half cycle #0,1)
;jobs required every non start of half cycle sample time interval
;clear start flag
SBIC 3,FLAGS
;increment input phase
LAC PHIN
ADD ONE
SACL PHIN
;check if it is near end of half cycle
LACK T1T2c-1
SUB PHIN
BGZ a1
;near end of half cycle (0,1)
;set end of half cycle flag
SBIS 5,FLAGS
a1:
;execute main sample time routine
CALL mainsam
;test if U fired first - FLAGS.2 set
B abl
;U fired first (0,1)#
;U phase routine
CALL uphase
;V phase routine
CALL vphase
B endsmp
abl:
;U not fired first #(0,1)
;V phase routine
CALL vphase
;U phase routine
CALL uphase
endsmp:
;restore registers for return to current background or deep
background operations
LST SSTAT
OUT SBSR,7
ZALH SAH
ADDS SAL
LT STREG
;return from interrupt
rin: RET

;SUBROUTINE UPHASE
uphase:

;check if stage 2 flag is set for U phase
LAC UFLAG
SUB TWO
BNZ ustg3
;stage 2 flag set (0,1)#
;call PU2 routine
B pu2
pu21:
B ustg3

;stage 2 flag not set #(0,1)
;check if stage 3 flag is set for U phase -

ustg3: SUB ONE
BNZ uabort
;stage 3 flag set (0,1)#
;no operations in stage 3
B ustg3
;stage 3 flag not set #(0,1)
;wrong UFLAG - attempt recovery
uabort: LAC TWO
SACL UFLAG
ustg4:
;return
RET

;SUBROUTINE VPHASE
vphase:

;check if stage 2 flag is set for V phase
LAC VFLAG
SUB TWO
BNZ vstg3
;stage 2 flag set (0,1)#
;call PV2 routine
B pv2
pv21:
B vstg3
;stage 2 flag not set #(0,1)
;check if stage 3 flag is not set for V phase -
vstg3: SUB ONE
BNZ vabort
B vstg3
;stage 3 flag not set (0,1)
;wrong VFLAG - attempt recovery
vabort: LAC TWO
SACL VFLAG
vstg4:
;return
RET

; SAMPTIM2 ROUTINE.
This routine resets as many variables as possible including the stack to maximise the chance of recovery from a noise disruption.

It runs once at the start of each half cycle. Jobs to be done once per half cycle are included.

```assembly
samptim2:
; output test data
LACK 008H
SACL TEMP
LAC TEMP,8
ADD PHOUT
SACL TEMP
LAC THREE
TBLW TEMP
; read speed v/f counter for use in PACE routine
LAC TS
SACL TSREAD
; complement mains polarity flag
LAC ONE,8
XOR FLAGS
SACL FLAGS
; test for start of mains cycle: mains polarity flag is 1
SBIT 8,FLAGS
BZ cyc1
; start of mains cycle (0,1)
; load MAINPER with the total count for the last cycle
LAC WDCOUNT
SACL MAINPER
; clear WDCOUNT
ZAC
SACL WDCOUNT
; set background flag 3 to run pll routine
SBIS 3,BFLAGS

cyc1:
; indicate background calculations by setting ur & vr flags
LAC THREE
OR BFLAGS
SACL BFLAGS
; clear end of half cycle flag
SBIC 5,FLAGS
; initialise PHIN to 0
ZAC
SACL PHIN
; update speed to new speed input
LAC SPDNEW
SACL SPDOLD
; set TRIAC control word: a 0 indicates the port connected to a triac to be turned on this half cycle
; U phase
; test mains polarity same as bank polarity - UBANK.0 = FLAGS.8
LAC ONE,8
AND FLAGS
SACL TEMP
LAC UBANK,8
XOR TEMP
BNZ xr1
; mains polarity same as bank polarity (0,1)
; load TRIAC with 11111101B
LAC MINUS
SUB ONE,1
SACL TRIAC
B xr4
; mains polarity opposite to bank polarity #(0,1)
; load TRIAC with 11111110B
xr1: LAC MINUS,1
```
SACL TRIAC
;V phase
;test mains polarity same as bank polarity - VBANK.0 = FLAGS.8
xr4: LAC VBANK,8
XOR TEMP
BNZ xr2
; mains polarity same as bank polarity (0,1)#
;clear bit 3 of TRIAC
SBIC 3,TRIAC
B xr3
; mains polarity opposite to bank polarity #(0,1)
;clear bit 2 of TRIAC
xr2: SBIC 2,TRIAC
xr3:
;execute main sample time routine
CALL mainsam
; call PU1 routine
CALL pu1
; call PV1 routine
CALL pv1
; execute back routine
B back

; MAIN SAMPLE TIME ROUTINE
; Runs every sample time interval.
mainsam:
;set bank to IO ports: BANK 0
BANK 0
;remove gate signals, U phase
;test if UPULSE <= 0
LAC UPULSE
BGZ up0
; UPULSE <= 0 (0,1)#
; remove U phase gate signals: set ports 0,1
OUT THREE,2
B up1
up0:
; UPULSE > 0 #(0,1)
; decrement UPULSE
SUB ONE
SACL UPULSE
up1:
; remove gate signals, V phase
; test if VPULSE <= 0
LAC VPULSE
BGZ vp0
; VPULSE <= 0 (0,1)#
; remove V phase gate signals: set ports 2,3
LAC THREE,2
SACL TEMP
OUT TEMP,2
B vp1
vp0:
; VPULSE > 0 #(0,1)
; decrement VPULSE
SUB ONE
SACL VPULSE
vp1:
; read motor flux counter
CALL read
; calculate new output phase
; add speed value to OUTSUM
ZALS OUTSUM
ADDS SPDOLD
SACL OUTSUM
; test for carry
SUBS OUTSUM
BZ sti2
; carry (0,1)
; increment PHOUT
LAC PHOUT
ADD ONE
SACL PHOUT
; check if PHOUT >= OUTPERc [output period]
LACK OUTPERc
SUB PHOUT
BGZ sti2
; PHOUT >= OUTPERc (0,1)
ZAC
SACL PHOUT
sti2:
; calculate V phase, PHV = PHOUT + OUTPERc/4 [OUTPER4c]
LACK OUTPER4c
ADD PHOUT
SACL PHV
; update boost integrals UBOOST, VBOOST
CALL boost
; output test data
; V phase
LAC FLUXREF
ADD PHV
TBLR TEMP
LT VX4
MPY TEMP
PAC
SACH TEMP,4
LAC TEMP
ADD VFLUX,8
SACL TEMP
LAC ONE
TBLW TEMP
; U phase
LAC FLUXREF
ADD PHOUT
TBLR TEMP
LT VX4
MPY TEMP
PAC
SACH TEMP,4
LAC TEMP
ADD UFLUX,8
SACL TEMP
LAC TWO
TBLW TEMP
; return
RET
.end
.title "ZEROVF"
;Calculates UVFSET and VVFSET, V/F count per WDT count for U and V
;phases
; for zero input voltage to the v/f converters. It does this by
; finding the
; total v/f count for 2^16 WDT counts.

.mlib "mac.lib"

;CALLED FROM: MAIN

;UNCHANGED DATA:
.ref ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,MINUS

;CHANGED DATA:
.ref TEMP,TEMP1,TEMP2,TMP,UVFSET,VVFSET

;ADDRESSES:
.def zerovf0
.ref zerovf1

;PROGRAM:
zerovf0:
;set periods of timers 1 & 2 to 2^16
;load period comparators with FFFFh
BANK 2
LAC MINUS
SACL TEMP
OUT TEMP,1
OUT TEMP,3
;set timers for external clock, no prescaling
;load TCON with 0111 1000 1111 0111 b
LACK 11110111B
ADD SEVEN,11
ADD ONE,14
SACL TEMP
OUT TEMP,4
;read timer values into TEMP, TEMP1
IN TEMP,0
IN TEMP1,2
;wait 2^16 * 8 - 4 cycles
;delay of 1 + (2^16 - 1) * 8 cycles.
ZALS MINUS
sd2 SUBS ONE
NOP
NOP
NOP
NOP
NOP
BNZ sd2
;delay of 3 cycles
NOP
NOP
NOP
;read timer values into TEMP2, TMP
IN TEMP2,0
IN TMP,2
;calculate UVFSET = TEMP2 - TEMP
LAC TEMP2
SUB TEMP
SACL UVFSET
;calculate VVFSET = TMP - TEMP1
LAC TMP
SUB TEMP1
SACL VVFSET
;return
  B   zero
.end
The cycloconverter software.

The lookup tables include flux table, boost table, integral of supply voltage, double integral of the supply voltage and endstop table.

.def bstref, flxref, caltab, endref, FPEAKc, Vc

Constants determining output voltage and boost voltage

Vc EQU 140 ;rms output voltage at 25Hz. Must be even integer less than 256.

M EQU 27 ;five times the peak boost voltage at 0Hz. Must be integer.

Make peak value of flux table available for external use

FPEAKc EQU Vc / 2

Flux table. This is a sine table with a peak value of 512

flxref:
.int 0, 18, 36, 54, 71, 89, 106, 124, 141, 158, 175, 192, 208, 224, 240
.int 256, 271, 286, 301, 315, 329, 343, 356, 368, 380, 392, 403, 414, 424, 434
.int 443, 452, 460, 468, 475, 487, 492, 497, 501, 504, 507, 509, 511, 512
.int 512, 512, 511, 509, 507, 504, 501, 497, 492, 487, 481, 475, 468, 460, 452
.int 443, 434, 424, 414, 403, 392, 380, 368, 356, 343, 332, 315, 301, 286, 271
.int 256, 240, 224, 208, 192, 175, 158, 141, 124, 106, 89, 71, 54, 36, 18

.int 0, -18, -36, -54, -71, -89, -106, -124, -141, -158, -175, -192, -208, -224, -240
.int -443, -452, -460, -468, -475, -487, -492, -497, -501, -504, -507, -509, -511, -512
.int -512, -512, -511, -509, -507, -504, -501, -497, -492, -487, -481, -475, -468, -460, -452
.int -256, -240, -224, -208, -192, -175, -158, -141, -124, -106, -89, -71, -54, -36, -18

.int 0, 18, 36, 54, 71, 89, 106, 124, 141, 158, 175, 192, 208, 224, 240
.int 256, 271, 286, 301, 315, 329, 343, 356, 368, 380, 392, 403, 414, 424, 434
.int 443, 452, 460, 468, 475, 487, 492, 497, 501, 504, 507, 509, 511, 512
.int 512, 512, 511, 509, 507, 504, 501, 497, 492, 487, 481, 475, 468, 460, 452
.int 443, 434, 424, 414, 403, 392, 380, 368, 356, 343, 332, 315, 301, 286, 271
.int 256, 240, 224, 208, 192, 175, 158, 141, 124, 106, 89, 71, 54, 36, 18

.int 0, -18, -36, -54, -71, -89, -106, -124, -141, -158, -175, -192, -208, -224, -240
.int -443, -452, -460, -468, -475, -487, -492, -497, -501, -504, -507, -509, -511, -512
.int -512, -512, -511, -509, -507, -504, -501, -497, -492, -487, -481, -475, -468, -460, -452
.int -256, -240, -224, -208, -192, -175, -158, -141, -124, -106, -89, -71, -54, -36, -18

Boost table. A sine table with no offset

bstref:
.int M*0,M*8,M*16,M*24,M*32,M*40,M*48,M*55,M*63,M*71,M*78,M*86,M*93,M*100,M*108
.int M*115,M*121,M*128,M*135,M*141,M*147,M*153,M*159,M*165,M*170,M*175,M*180,M*185,M*190,M*194
.int M*198,M*202,M*206,M*209,M*212,M*215,M*218,M*220,M*222,M*224,M*226,M*227,M*228,M*229
.int M*229,M*228,M*227,M*226,M*224,M*222,M*220,M*218,M*215,M*212,M*209,M*206,M*202
.int M*198,M*194,M*190,M*185,M*180,M*175,M*170,M*165,M*159,M*153,M*147,M*141,M*135,M*128,M*121
.int M*114,M*108,M*100,M*93,M*86,M*78,M*71,M*63,M*55,M*48,M*40,M*32,M*24,M*16,M*8
.int M*0,M*8,M*16,M*24,M*32,M*40,M*48,M*55,M*63,M*71,M*78,M*86,M*93,M*100,M*108
.int M*115,M*121,M*128,M*135,M*141,M*147,M*153,M*159,M*165,M*170,M*175,M*180,M*185,M*190,M*194
.int M*198,M*202,M*206,M*209,M*212,M*215,M*218,M*220,M*222,M*224,M*226,M*227,M*228,M*229
.int M*229,M*228,M*227,M*226,M*224,M*222,M*220,M*218,M*215,M*212,M*209,M*206,M*202
.int M*198,M*194,M*190,M*185,M*180,M*175,M*170,M*165,M*159,M*153,M*147,M*141,M*135,M*128,M*121
.int M*114,M*108,M*100,M*93,M*86,M*78,M*71,M*63,M*55,M*48,M*40,M*32,M*24,M*16,M*8

;itab & caltab are for PHIN = 0.5, 1.5, ....... 45.5

;Integral of supply voltage over one half cycle - reference only, not used.
; ; .int 84,80,76,73,69,65,61,57,53,49,45,41,38,34,30
; ; .int 27,23,20,17,15,12,10,8,6,4,3,2,1,0,0,0

; 4(d)+4(c) over one half cycle.
caltab: .int 5128,5109,5073,5020,4950,4865,4766,4653,4527,4390
.int 4243,4087,3923,3752,3576,3395,3211,3024,2837,2650
.int 2464,2280,2099,1922,1750,1583,1423,1270,1124,987
.int 857,737,626,524,431,348,274,210,155,109,71,42,21,7,1,1

C-64
Table determining the endstop in each half cycle for different motor voltages.

<table>
<thead>
<tr>
<th>Voltage (volts)</th>
<th>Endstop Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>40,40,40,40,40,40</td>
<td>40,40,40,40,40,40</td>
</tr>
<tr>
<td>38,37,37,37,37,37</td>
<td>38,37,37,37,37,37</td>
</tr>
<tr>
<td>35,35,35,35,35,35</td>
<td>34,34,34,34,34,34</td>
</tr>
<tr>
<td>32,32,32,32,32,32</td>
<td>31,31,31,31,31,31</td>
</tr>
<tr>
<td>29,29,29,29,29,29</td>
<td>28,28,28,28,28,28</td>
</tr>
<tr>
<td>26,26,26,26,26,26</td>
<td>25,25,25,25,25,25</td>
</tr>
<tr>
<td>23,23,23,23,23,23</td>
<td>22,22,22,22,22,22</td>
</tr>
<tr>
<td>0,0,0,0,0,0,0</td>
<td>0,0,0,0,0,0,0</td>
</tr>
</tbody>
</table>

.end