Encoder-Check PLC Programs

Turbo PMAC controllers have several features to check for the integrity and presence of quadrature encoder feedback. These features include:

- Hardware count-error detection circuitry
- Hardware count capture capability with index pulse
- Hardware encoder-loss detection circuitry

Different users desire different actions on detection of these issues, so the action on detecting these conditions typically is left to the user's application program. This section shows some possibilities for reacting to these events. The examples are for a single axis only, but are easy to duplicate for multiple axes. Notice that the hardware features in the Servo ICs differ in their details between PMAC-style ICs and PMAC2-style ICs, so the setup for each differs slightly.

Count Errors

This first example checks for the hardware count error status bit and evaluates the number of counts elapsed between index-pulse captured positions. It does not start looking until the motor has been homed, and when it finds a problem, it simply clears the "home complete" status bit, indicating that there is no longer a valid position reference. No other automatic action is taken here. Typically, a new homing-search move is required.

This example assumes that the index pulse is used as at least part of the homing trigger. It uses the encoder position captured on the homing trigger as its reference for future captures with the index pulse.

Substitutions and Definitions – PMAC-style Servo IC Definitions

```
#define SIC0Ch1CaptPos  M103        ; Hardware-captured position
SIC0Ch1CaptCompPos->X:$078003,0,24,S
#define SIC0Ch1CaptFlag  M117       ; Hardware capture flag
SIC0Ch1CaptFlag->X:$078008,17,1
#define SIC0Ch1EncCtErr  M118       ; Hardware count error bit
SIC0Ch1EncCtErr->X:$078008,18,1
#define SIC0Ch1CaptCtrl  I7012      ; Defines trigger condition
```

PMAC2-style Servo IC Definitions

```
#define SIC0Ch1CaptPos  M103        ; Hardware-captured position
SIC0Ch1CaptCompPos->X:$078003,0,24,S
#define SIC0Ch1CaptFlag  M117       ; Hardware capture flag
SIC0Ch1CaptFlag->X:$078008,11,1
#define SIC0Ch1EncCtErr  M118       ; Hardware count error bit
SIC0Ch1EncCtErr->X:$078008,8,1
#define SIC0Ch1CaptCtrl  I7012      ; Defines trigger condition
#define SIC0Ch1GateIndEna  I7014    ; Limit index to one AB state
#define SIC0Ch1GateIndSta  I7015    ; Define AB state for index
```

Software Register Definitions

```
#define Mtr1Homed  M145             ; Indicates valid pos reference
Mtr1Homed->Y:$0000C0,10,1
#define Mtr1HomeCaptPos  M173       ; Stored on homing trigger
Mtr1HomeCaptPos->Y:$0000CE,0,24,S
#define Mtr1LastCaptPos  P174       ; From previous encoder cycle
#define Mtr1NewCaptPos  P175        ; Latest capture
#define Mtr1DeltaCaptPos  P176      ; Diff between New and Last
#define Mtr1ExtCaptPos  P177        ; Extended captured position
#define Mtr1CtsPerRev  P178         ; Expected counts per cycle
#define Mtr1ModOp  P179             ; Modulo operator
#define Diff  P180                  ; Difference from ideal
#define 2Power23  8388608           ; For rollover of 24-bit reg
```

Set System Constants (PMAC-style Servo IC)

```
Mtr1CtsPerRev=2000                  ; After decode
Mtr1IndexWidth=4                    ; Width in cts (full AB cycle)
```

Set System Constants (PMAC2-style Servo IC)

```
Mtr1CtsPerRev=2000                  ; After decode
SIC0Ch1GateIndEna=1                 ; Selected "gated index"
SIC0Ch1GateIndSta=1                 ; Gate to low-low AB state
Mtr1IndexWidth=1                    ; Width in cts (after gating)
```

```
OPEN PLC 17 CLEAR
WHILE (Mtr1Homed=0)                 ; No checking before homed
ENDWHILE
Mtr1LastCaptPos=Mtr1HomeCaptPos     ; Store for comparison to next
Mtr1ExtCaptPos=Mtr1HomeCaptPos      ; Use as starting position
SIC0Ch1CaptCtrl=1                   ; Rising edge of index alone
Mtr1ModOp=-Mtr1CtsPerRev/2          ; Pre-compute for efficiency
WHILE (Mtr1Homed=1)
  WHILE (SIC0Ch1CaptFlag=0 AND Mtr1EncCtErr=0)
  ENDWHILE
  IF (Mtr1EncCtErr=1)               ; Count error found?
    Mtr1Homed=0                     ; Invalid position reference
  ELSE
    IF (SIC0Ch1CaptFlag=1)          ; New capture?
      Mtr1NewCaptPos=SIC0Ch1CaptPos ; Store position
      Mtr1DeltaCaptPos=(Mtr1NewCaptPos-Mtr1LastCaptPos)%-2Power23
      Mtr1ExtCaptPos=Mtr1ExtCaptPos+Mtr1DeltaCaptPos
      Mtr1LastCaptPos=Mtr1NewCaptPos      ; For next cycle
      Diff=(Mtr1ExtCaptPos-Mtr1HomeCaptPos)%Mtr1ModOp
      IF (ABS(Diff)>Mtr1IndexWidth) ; Wrong counts per rev?
        Mtr1Homed=0                 ; Invalid position reference
      ENDIF
    ENDIF
  ENDIF
ENDWHILE
CLOSE
```

The above program extends the 24-bit hardware captured position to a 48-bit value by taking the difference from the last captured value, rolling over the difference into a +/-2^{23} range (using the `%-2Power23` modulo operation), and adding this difference to the last extended position. This permits the algorithm to work even when the hardware counter rolls over. The modulo, or remainder, operation is used again to evaluate whether the capture has come in the right place in the encoder's cycle. Note that in Turbo PMAC, a modulo operation with a negative operator – "%-N" returns a value between –N and +N.

To clear the encoder-count error in a PMAC-style IC, you must set the encoder counter value. This is done by writing to the compare register with the count-write enable bit set. To avoid a jump, it is best to write the present counter value back into it. With the following additional definitions:

```
#define SIC0Ch1EncPos  M101         ; Present encoder counter value
SIC0Ch1EncPos->X:$078001,0,24,S
#define SIC0Ch1CtWriteEna  M110     ; Count-write enable control bit
SIC0Ch1CtWriteEna->X:$078000,10,1
```

You can then use the following commands to clear the error bit:

```
SIC0Ch1CtWriteEna=1
SIC0Ch1CaptCompPos=SIC0Ch1EncPos
SIC0Ch1CtWriteEna=0
```

To clear the encoder-count error in a PMAC2-style IC, simply write a zero to the count-error bit:

```
SIC0Ch1EncCtErr=0
```

Encoder-Loss Errors

This next example program reacts to a detection of encoder loss. This is a more serious condition than a count error, so a kill command is issued when the loss is detected.

Encoder-loss detection bits come in many different locations in different versions of the Turbo PMAC. The Turbo PMAC User's Manual shows these locations in tables for the different versions in the Making Your Application Safe section.

Substitutions and Definitions

```
#define Mtr1OpenLoop  M138          ; Motor status bit
Mtr1OpenLoop->Y:$0000B0,18,1        ; Standard definition
#define Enc1LossIn  M180            ; Input loss-detection bit
Enc1LossIn->Y:$078F08,5,1           ; UMAC SIC2 Ch1 loss bit
#define Mtr1EncLossStatus  P180     ; Internal latched status
#define Lost  0                     ; Low-true fault here
#define OK  1                       ; High is encoder present
```

Program to Check for and React to Encoder Loss

```
OPEN PLC 18 CLEAR
```

Logic to Disable and Set Fault Status

```
IF (Mtr1OpenLoop=0 AND Enc1LossIn=Lost)   ; Closed loop, no enc
CMD^K                               ; Kill all motors
Mtr1EncLossStatus=1
ENDIF
```

Logic to Clear Fault Status

```
IF (Mtr1OpenLoop=1 AND Enc1LossIn=OK AND Mtr1EncLossStatus=0)
      Mtr1EncLossStatus=0
ENDIF
CLOSE
```