Customer Support Engineering Tech Note #21

Displaying Error Codes Using Flashing Lights

Abstract

Many applications do not use any kind of display capable of showing error codes. If you are not able to connect MotionPRO™ to monitor the system, trouble shooting is extremely difficult. This tech note shows you how to write your program so that it indicates any MotionBASIC® error code using the LED indicators on the front of the controller. It also indicates the status of the program using a DIO@ output.

Coding System

The DIO@ output, the example uses number 16, indicates whether the program is running or not.

- Flashing ten times per second means the program running but still initializing.
- Flashing once per second means it is done initializing and is running.
- Not flashing (on or off) means your program is not running.

The User 1 LED indicator on the front panel of the controller tells you the MotionBASIC® error code (ERR value). This is the single most important piece of information you need to trouble shoot any problems. The code is a four digit number which is displayed one digit at a time.

First the LED turns on for 1.5 seconds, this tells you it is about to start the sequence. It then flashes the first digit. If the first digit is one, it will flash once. If it is two, it will flash twice and so on. After indicating the first digit, the LED will pause for one second and then flash on for a hundredth of a second. This very quick flash tells you it has finished with the first digit and is about to start on the second digit.

Once all four digits have been displayed, it will turn on for 1.5 seconds again to tell you it is about to start the sequence over again.

Here is an example:

One 1.5 second flash about to start
One 0.25 second flash First digit is 1
One quick flash
Nine 0.25 second flashes Second digit is 9
One quick flash
One 0.25 second flash Third digit is 1
One quick flash
No 0.25 second flash Fourth digit is 0
One quick flash

The error code is 1910 which indicates the E-STOP input opened.

Example Program

The following example program shows you how it works. You will need an Output module installed in DIO position 16.
Customer Support Engineering Tech Note #21

When you run the program, Output 16 will start flashing rapidly, this tells you the program has started. A few seconds later it will slow to about once per second telling you it has successfully finished the initialization and is now in the main part of the program.

Once the ESTOP.OK input is closed, the program will let you enter an error code number. When you press <ENTER> it will simulate the error so you can observe how the LED flashes out the code. Clear the error by turning the ESTOP.OK input off then on again.

You can then stop the program by pressing <Ctrl-C>, this stops the program to simulate what happens if your program crashes and the system drops into direct mode. You will see output 16 stop flashing (it will stay on or off).

Program Listing

```
!AUTO 0

Routine name: POWERUP
Abstract: Program entry point
Routines called: CLEAR.FAULTS, ERROR.HDLR, INIT.DIO, INIT.EVENTS
              INIT.FUNCTIONS, INIT.VARS, RESTART
Variables used: ESTOP.OK@, PERIOD

POWERUP:
  MP.CONFIG 'this is where you would call the MP.CONFIG sub-routine in a real application
  INIT.DIO 'initialize the discrete I/O
  INIT.VARS 'initialize your program variables
  INIT.EVENTS 'initialize any ON EVENT routines
  INIT.FUNCTIONS 'initialize any user defined functions
  CLEAR.FAULTS
  ON ERROR GOTO ERROR.HDLR
  WAIT 5000 'delay 5 seconds so you can see the change in flash frequency
  PERIOD=-1000 'change the run.light frequency
  IF ESTOP.OK@ THEN RESTART ELSE ERROR 1910
END

Routine name: INIT.EVENTS
Abstract: Initialize any EVENT subroutines.
Routines called: INT.ROUTINE
Variables used: DIO@(), PERIOD, RUN.LIGHT

INIT.EVENTS:
  ON EVENT DIO@(RUN.LIGHT) GOSUB INT.ROUTINE
  EVENT DIO@(RUN.LIGHT) ON
  EVENT ON
  DIO@(RUN.LIGHT)=OFF
  PERIOD=-100
  DIO@(RUN.LIGHT)=PERIOD
RETURN
```
Program: TN021.BAS
Routine name: INIT.FUNCTIONS
Abstract: Initialize any user defined functions
Routines called: None
Variables used: None

INIT.FUNCTIONS:
'Any DEF FN ... statements go here.
RETURN

Program: TN021.BAS
Routine name: INIT.VARS
Abstract: Initialize any variables your program uses
Routines called: None
Variables used: ERR.DIGIT(), INITIALIZED

INIT.VARS:
IF NOT INITIALIZED THEN
  'Things that only need to be initialized once
  'when the program is first run, go here.
  ERASE ERR.DIGIT :DIM ERR.DIGIT(4)
ENDIF
  'String variables and other variables you need to
  'initialize each time the program is run, go here.
INITIALIZED=TRUE
RETURN

Program: TN021.BAS
Routine name: INIT.DIO
Abstract: Initialize any DIO points your program uses
Routines called: None
Variables used: FAULT.LIGHT, IO.MODE@(), RUN.LIGHT

INIT.DIO:
  FAULT.LIGHT=1 :IO.MODE@ (FAULT.LIGHT)="O"
  RUN.LIGHT=16 :IO.MODE@ (RUN.LIGHT)="O"
RETURN

Program: TN021.BAS
Routine name: INT.ROUTINE
Abstract: Program running interrupt routine
Routines called: None
Variables used: DIO@(), PERIOD, RUN.LIGHT

INT.ROUTINE:
  DIO@ (RUN.LIGHT)=PERIOD
RETURN
Routine name: ESTOP.STATE

Abstract: Wait here for transition of ESTOP.OK@ from false to true to clear faults and restart

Routines called: CLEAR.FAULTS, ERR.LED, GET.FAULTS, RESTART

Variables used: DELAY, DIO@(), ESTOP.FLAG, ESTOP.OK@, EXIT

FAULT.LIGHT, STORED.ERR, USER1.LED@

ESTOP.STATE:
DIO@(FAULT.LIGHT)=ON
ESTOP.FLAG=NOT ESTOP.OK@
GET.FAULTS
USER1.LED@=OFF :DELAY=100 :WAIT.DELAY
EXIT=FALSE
WHILE NOT EXIT
   IF STORED.ERR THEN ERR.LED ELSE DELAY=50 :WAIT.DELAY
WEND
CLEAR.FAULTS
STACK CLEAR
RESTART
   'restart your main program
END

Routine name: ERR.LED

Abstract: Make the User 1 LED flash the error code number you could change this to flash a DIO@ output if you wanted to announce the code on the operator control panel rather than on the controller.

Routines called: WAIT.DELAY

Variables used: DELAY, DIGIT, ERR.DIGIT(), EXIT, STORED.ERR$

USER1.LED@, VALUE

ERR.LED:
USER1.LED@=ON :DELAY=2000 :WAIT.DELAY
USER1.LED@=OFF :DELAY=1000 :WAIT.DELAY
DIGIT=1
WHILE DIGIT<=LEN(STORED.ERR$) AND NOT EXIT
   VALUE=1
   WHILE VALUE<ERR.DIGIT(DIGIT) AND NOT EXIT
      USER1.LED@=ON :DELAY=250 :WAIT.DELAY
      USER1.LED@=OFF :DELAY=250 :WAIT.DELAY
      VALUE=VALUE+1
   WEND
   DELAY=1000 :WAIT.DELAY
   USER1.LED@=ON
   WAIT 10
   USER1.LED@=OFF
   DELAY=1000 :WAIT.DELAY
   DIGIT=DIGIT+ 1
WEND
RETURN
**Routine name:** WAIT.DELAY

**Abstract:** Waits for a time set in the DELAY variable (ms) checking to see if the ESTOP input has been while it waits.

**Routines called:** None

**Variables used:** DELAY, ESTOP.FLAG, ESTOP.OK@, EXIT, TMP

```
WAIT.DELAY:
    TMP=0
    WHILE TMP<DELAY AND NOT EXIT
        WAIT 50
        IF ESTOP.OK@ AND ESTOP.FLAG THEN
            EXIT=TRUE
        ELSEIF NOT ESTOP.OK@ THEN
            ESTOP.FLAG=TRUE
        ENDIF
        TMP=TMP+50
    WEND
RETURN
```

**Routine name:** GET.FAULTS

**Abstract:** Break the error code number into digits

**Routines called:** None

**Variables used:** ERR.DIGIT(), STORED.ERR, STORED.ERR$, TMP

```
GET.FAULTS:
    STORED.ERR$=STR$(STORED.ERR)
    FOR TMP=1 TO LEN(STORED.ERR$)
        ERR.DIGIT(TMP)=VAL(MID$(STORED.ERR$,TMP,1))
    NEXT TMP
RETURN
```

**Routine name:** RESTART

**Abstract:** Application restart location. This is where your application program takes over

**Routines called:** None

**Variables used:** AXIS.LIST@, DIO@(), MODE@(), RUN.LIGHT, X

```
RESTART:
    COLOR 7,0
    CLS
    MODE@(AXIS.LIST@)=5 'This is optional. If you are testing the example and don’t want to enable any servos, comment this line out.
    EVENT DIO@(RUN.LIGHT) ON
    EVENT ON
    WHILE TRUE
        'This is where the application program really starts. For this example we have a simple routine that lets you enter a MotionBASIC(R

```
'error code to see how the system responds to various errors.

INPUT @ 1,1;" Error code ";USING " #### ";X
PRINT
PRINT " Forcing error code";X
WAIT 100
ERROR X
WEND

Program: TN021.BAS
Routine name: ERROR.HDLR
Abstract: React to a fault or error
Routines called: ESTOP, ESTOP.STATE
Variables used: FAULT@, STORED.ERR

ERROR.HDLR:
STORED.ERR=ERR 'remember the error code number
'A CTRL-C stops the program for debugging purposes. Normally you
should comment this line out after completing your development.
IF STORED.ERR=1805 THEN CLOSE: ON ERROR GOTO 0
ESTOP
IF FAULT@={} THEN FAULT@=TRUE
RESUME ESTOP.STATE
END

Program: TN021.BAS
Routine name: ESTOP
Abstract: Emergency Stop
Routines called: None
Variables used: AXIS.LIST@, DSP.DONE@(), FAULT@, MODE@()

ESTOP:
HALT AXIS.LIST@
WAIT UNTIL DSP.DONE@(AXIS.LIST@) OR FAULT@<>{}
MODE@(AXIS.LIST@)=0
RETURN

Program: TN021.BAS
Routine name: CLEAR.FAULTS
Abstract: Attempt to clear faults
Routines called: None
Variables used: AFAULT@, AXIS.LIST@, AXIS.SET@, FAULT@, OTL.FWD@
               OTL.REV@, STORED.ERR, STORED.ERR$, USER1.LED@

CLEAR.FAULTS:
AXIS.SET@=AXIS.LIST@
OTL.FWD@=0 :OTL.REV@=0 :AFAULT@=0 :FAULT@=0 :WAIT 300
USER1.LED@=OFF
STORED.ERR=0
STORED.ERR$=""
RETURN