FDECODE
Decoding Alarms (Faults)
with a Function Block Written for a Siemens S5 PLC
An Automation List PLC Archive Document

Author:	Michael Griffin
Document Date:	23-JUL-2000
Date Originally Written:	1997
Type:	PLC subroutine (function block).
PLC Model:	Siemens S5 series.
Application History:	More than 3 dozen systems in current use.
Known Problems:	None.
Current Revision:	Original version.
Algorithm Source:	The algorithm was created for this application.

This document describes a simple method of decoding alarms (faults). These alarms may be displayed on a text OP (operator panel), or used in some other fashion. The actual algorithm used here is trivially simple, but this document describes applying it in a consistent and systematic manner.

A fault is a condition which prevents a machine from operating normally. An alarm is a message intended to convey information on the fault to the operator.

The majority of machine operations can be described in terms of an actuator with two valid states which are detected by two sensors. A typical example is a pneumatic cylinder whose extended and retracted positions are detected by two proximity or MRS switches.

With two sensors, there is a possibility of a total of four fault conditions:

- Sensor 'A' is off when it should be on.
- Sensor 'A' is on when it should be off.
- Sensor 'B' is off when it should be on.
- Sensor 'B' is on when it should be off.

The required state of the sensors inputs can be predicted by the state of the actuator output. Unless all four states are decoded, not all possible fault conditions have been accounted for.

The function block shown here does not determine that a fault exists, it simply decodes it into an alarm. Also included is a variant which handles situations where only one sensor is used.

Application Example

In the example this was drawn from, program block PB012 was called only when a fault was detected. It was called on a one shot basis to capture a snap shot of the machine state at the time the fault was detected. Within this program block the FDECODE function blocks are being called unconditionally. These segments could of course be scanned continuously provided conditional jumps were used instead of the unconditional jumps shown here. Calling these blocks continuously when the machine is operating normally will of course produce spurious alarms.

```
PB011
SEGMENT 7
 The Fault Decode Logic will capture the faults present while the sequence
 advance conditions are not true. These faults will then be available for
 display on the OP.
|STATION
|FAULT
|ONS
|  F030.0                               +JUMP TO SUBROUTINE--+
[----] [--------------------------------] PB012              [-----------
|                                       | FAULT              |
|                                       | DECODE             |
|                                       | LOGIC              |
|                                       |                    |
|                                       |                    |
[                                       |                    [-----------
|                                       +-------------------+
```

Within PB012 the FDECODE function blocks are called as many times as necessary. The following two rungs illustrate typical examples. This function block can be called as many times as necessary, but of course different parameters must be used with each instance. This example shows the FDECODE function block being called to decode faults from two separate devices. The outputs Q80.3 and Q80.5 are monitored by I81.2 to I81.4. The decoded alarms are contained in flags F130.0 to F130.7. Thus a total of eight possible alarm conditions are handled here.

```
PB012
SEGMENT 15
  Decode faults for mechanism raise/lower.
      : JU  FB204
NAME: FDECODE
 ACT: Q80.3
SNSA: I81.1
SNSB: I81.2
FLT1: F130.0
FLT2: F130.1
FLT3: F130.2
FLT4: F130.3
      :

SEGMENT 16
  Decode faults for the mechanism advance/retract.
      : JU  FB204
NAME: FDECODE
 ACT: Q80.5
SNSA: I81.3
SNSB: I81.4
FLT1: F130.4
FLT2: F130.5
FLT3: F130.6
FLT4: F130.7
      :
      : BE
```

This function block is really a very simple one, as can be seen from the source code found at the end of this document. What the above illustrates though is that it can be worth while incorporating even very simple algorithms into reusable subroutines.

Turning the decoding logic into a function block had two advantages. Firstly, it reduced the amount of typing required. Four rungs of ladder logic were turned into one function call with seven parameters. Secondly, and more important, I found that I made far fewer typographical mistakes this way. Previous ad hoc creation of fault logic was often incomplete and contained errors.

Algorithm

For those not familiar with Siemens S5 statement list, the algorithm can be reduced to four simple rungs of ladder logic. It is obvious enough that it is not worth explaining it here.

```
SEGMENT 1
| Q080.3   I081.1                                                F130.0
[---] [------]/[---------------------------------------------------( )---
|
SEGMENT 2
| Q080.3   I081.1                                                F130.1
[---]/[------] [---------------------------------------------------( )---
|
SEGMENT 3
| Q080.3   I081.2                                                F130.2
[---]/[------]/[---------------------------------------------------( )---
|
SEGMENT 4
| Q080.3   I081.2                                                F130.3
[---] [------] [---------------------------------------------------( )---
|
```

FB Source Code

```
FB204
                        Michael Griffin       1997
  This block decodes fault conditions where an actuator is sensed
by two inputs. The fault output bits are turned "ON" according to the
following table.

         ACT       SNSA        SNSB
FLT1     ON        OFF         ---
FLT2     OFF       ON          ---
FLT3     OFF       ---         OFF
FLT4     ON        ---         ON
SEGMENT 1

NAME: FDECODE
DECL: ACT  I, BI     ;ACTUATOR STATE.
DECL: SNSA I, BI     ;TRUE WHEN ACTUATOR "ON".
DECL: SNSB I, BI     ;TRUE WHEN ACTUATOR "OFF".
DECL: FLT1 Q, BI     ;ACT "ON" AND SNSA "OFF".
DECL: FLT2 Q, BI     ;ACT "OFF" AND SNSA "ON".
DECL: FLT3 Q, BI     ;ACT "OFF" AND SNSB "OFF".
DECL: FLT4 Q, BI     ;ACT "ON" AND SNSB "ON"
      : A   =ACT     ;The actuator is on, but the "advanced"
      : AN  =SNSA    ;sensor is still off.
      : =   =FLT1
      :
      : AN  =ACT     ;The actuator is off, but the "advanced"
      : A   =SNSA    ;sensor is still on.
      : =   =FLT2
      :
      : AN  =ACT     ;The actuator is off, but the "retracted"
      : AN  =SNSB    ;sensor is still off.
      : =   =FLT3
      :
      : A   =ACT     ;The actuator is on, but the "retracted"
      : A   =SNSB    ;sensor is still on.
      : =   =FLT4
      : BE
```

Single Sensor: The same method can also be used when only one sensor is associated with the actuator.

```
FB205
                     Michael Griffin      1997
  This block decodes a fault condition where an output is sensed by one
input.
The fault bits are turned "ON" according to the following table.
         ACT      SENS
FLT1     ON       OFF
FLT2     OFF      ON
SEGMENT 1

NAME: FDECODE1
DECL: ACT  I, BI        ;ACTUATOR STATE.
DECL: SENS I, BI        ;TRUE WHEN ACTUATOR "ON".
DECL: FLT1 Q, BI        ;ACT "ON" AND SENS "OFF".
DECL: FLT2 Q, BI        ;ACT "OFF" AND SENS "ON".
      : A   =ACT        ;The actuator is on but the "advanced"
      : AN  =SENS       ;sensor is still off.
      : =   =FLT1
      :
      : AN  =ACT        ;The actuator is off but the "advanced"
      : A   =SENS       ;sensor is still on.
      : =   =FLT2
      : BE
```

Note: ***I have not experienced any problems with the program code in this document in my own applications so far. I have provided this example for illustrative purposes only. You***

are free to use the information in this document for your own purposes, but in doing so you must of course accept complete responsibility for any problems, bugs, unintended consequences, etc. you may encounter. In other words, use at your own risk.