Fault Handling for Software Faults — Model 5100 and 5200 Controllers

How to use the fault handler registers to handle software fault events
December, 2004

Control Technology Corporation, Hopkinton, MA • 800.282.5008 • www.ctc-control.com

Did you know you the Blue Fusion 5100/5200 controllers have Fault Handling capabilities for software faults?

This allows you to decide what to do when certain types of faults occur. The Fault Handler is enabled using the FAULT_MASK_REGISTER and the FAULT_TASK_REGISTER (see the list of Fault Registers in Table 2). When a Handler is executing it will ignore further soft faults and continue executing. The fault state must be cleared for normal operation to continue. This is controlled by register 13041, the **TASK_FAULT_CLEAR_REGISTER** (WriteOnly). This register controls the state of program execution. Below are the possible settings for this register:

VALUE	STATE
1	RESET – Reset the controller only and then stop.
5	RESTART – Reset the controller and begin running again at step 1.
6	STOPPED – Stop the controller but do not reset.
8	RUNNING – Ignore the fault and continue running.
9	FAULT – Continue to fault as usual.
10	SHUTDOWN – Reset the controller and shutdown, requires a power cycle to exit.

Here are the register definitions and 2 simple examples. See the Fault Task Handler section in the communications guide for more info (pg 65).
http://www.ctc-control.com/customer/techinfo/docs/5100_951/951-510002-0002_Communications_Protocol_Guide.pdf

Table 1
Fault Registers
13032 Fault Code – (R) Contains the fault code for what caused the fault.
13033 Fault Step – (R) Step in which fault occurred.
13034 Fault Task – (R) Task number, starting at 1, which caused the fault.
13035 Fault Data – (R) Any relevant error data.
13038 Fault Step Register – (R/W) Step to branch to when fault occurs. Write a 0 to disable.
13039 Fault Task Register – (R) Task number that is the active Fault Handler, 0 means none.
13040 Fault Mask Register – (R/W) Bit OR of types of fault that will invoke the handler, by default all enabled (-1) when the Fault Step Register is written
13041 Fault Clear Register – (W) Used to write the recovery state when done processing the Fault.

Please Note: These examples reset the controller when certain types of software faults occur. Resetting the controller can cause adverse effects in many systems. Setting a register with an error code and stopping the controller may be more appropriate in many cases (as shown in example 2).

Example 1:
[1] init
;;; This program example does not do anything except handle a communication failure.

;;; Note that a Fault Handler is initialized in the first step to monitor for communications failure. The FaultMaskRegister ;;; must be set after the FaultStepRegister, otherwise the handler will be invoked for all faults (default).

<NO CHANGE IN DIGITAL OUTPUTS>
----
store 10 to FaultStepRegister_reg13038
store 32 to FaultMaskRegister_reg13040
goto Step_Two
.
.[Main Program]
.
[10] Fault_Handler
;;; This step is invoked should a fault occur, such as a network disconnect. The FaultMaskRegister controls under what
;;; circumstances the handler is invoked. This example is very simple. It basically shuts all the local outputs off and sets a ;;; flag called FaultFlag that has no purpose. Note that no other tasks will be running in the system nor can this task fault ;;; when the handler is invoked.

<TURN OFF ALL DIGITAL OUTPUTS>
store 1 to FaultFlag
delay 3 sec goto ClearFault

[11] CearFault
;;; Now attempt to recover from the fault by issuing a RESTART command.

<NO CHANGE IN DIGITAL OUTPUTS>
----
store 5 to FaultClearRegister_reg13041
goto FaultHandler

Example 2:
[1] init
;;; This program example does not do anything except handle any type of software fault. Note that a Fault Handler is
;;; initialized in the first step to monitor for any type fault. This example has the Fault Hander jump to this task for all
;;; Software faults.

<NO CHANGE IN DIGITAL OUTPUTS>
----
store 10 to FaultStepRegister_reg13038
store 0 to FaultMaskRegister_reg13040
goto Step_Two
.
.[Main Program]
.
[10] Fault_Handler
;;; This step is invoked should any fault occur. This example is very simple. If the fault is a Watch Dog Timer Error
;;; (fault code 28), the controller will reset.   All other faults will report the code to a register for display on an HMI
;;; and shutdown the controller. Note that no other tasks will be running in the system nor can this task fault
;;; when the handler is invoked.

<NO CHANGE IN DIGITAL OUTPUTS>
----
if FaultCode_reg13032 = 28 goto HandleWatchDogError

store FaultCode_reg13032 to HMI_FaultCodeRegister
delay 3 sec goto Shutdown

[11] Shutdown
;;; Shutdown the controller. The controller will need require a powercycle to continue.

<NO CHANGE IN DIGITAL OUTPUTS>
----
store 10 to FaultClearRegister_reg13041
goto FaultHandler

[12] HandleWatchDogError
;;; Since a Watch Dog Timer Error occurred we will reset the controller and begin running at step 1.

<NO CHANGE IN DIGITAL OUTPUTS>
----
store 5 to FaultClearRegister_reg13041
goto FaultHandler