Application Note

Maintenance Override Programming

TÜV Rheinland publishes guidelines on the maintenance facilities of safety related programmable systems, currently available on the Internet at http://www.tuv-fs.com/modr_3_e.htm. Whilst this draft paper is not a compulsory part of safety system operation, it has been revised over many years and serves as a guide to sensible system design. The paper has been extended to cover maintenance facilities applied over communications networks.

Maintenance overrides are used to disable inputs or outputs from a safety related system for maintenance activities, to avoid trip actions caused by spurious signals or logic tests. The preferred method for applying overrides is by hardwired switches on a panel, either via inputs to the safety system application or directly affecting the circuit. Whilst this is the most secure method, it is also inflexible; there may not be room for all necessary switches and modifications to the switch panel are complex and expensive.

This document describes methods for implementing maintenance override facilities using programming and communications, within the TÜV guidelines.

---

Issue Record

<table>
<thead>
<tr>
<th>Issue Number</th>
<th>Date</th>
<th>Revised by</th>
<th>Technical Check</th>
<th>Authorised by</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>June 08</td>
<td>Nick Owens</td>
<td>Andy Holgate</td>
<td>Pete Stock</td>
<td>Initial Issue</td>
</tr>
</tbody>
</table>
Maintenance Override Programming

A maintenance override can be equally applied to inputs or outputs. This may also be described as an inhibit. Strictly speaking an ‘inhibit’ prevents something happening whereas an ‘override’ applies a forced state, but these two words tend to be used nearly interchangeably.

Toolset Debugger Tools

The Toolset allows input and output variables to be locked, which disconnects them from the real I/O. The downstream side of the disconnection may then be manipulated through the Toolset (for inputs, the application variable is manipulated, for outputs, the real output is manipulated).

Locking should only be used during offline testing and when the plant is not live; it should not be used as a maintenance override facility. The Processor will not hot-swap with locked variables present.

Do not use the Toolset debugger to apply overrides to a system connected to live plant. Pre-programmed maintenance override logic with security measures should be used, as described below.

Maintenance Override Programming TUV Requirements

This section lists the specific clauses from TUV’s paper which apply to maintenance override programming. It recommends methods for implementing each clause.

The following section gathers the recommended methods together into a programming specification for the safety system side of the maintenance overrides.

The TUV paper contains general details about security and the integrity of engineering tools; it is written for any possible implementation/permutation of tools for programming, debugging, commissioning, maintenance and operation. One sentence from these sections is worth applying to maintenance override programming.

1. Additional requirements must also be in place that guarantee security

This is best interpreted as ensuring that communications transactions are secure, since the paper also considers ‘authorisation’ and ‘enabling’ separately (clause 4 below).

There is no perfect way to guarantee communications security; opening a route to diverse transactions means that there is always a probability that one genuine transaction could be mutated into another genuine transaction. Packet checks and command validation will not help in this case.

The best method to improve data security therefore is to require human validation of the request. If the command is sent from DCS to system by one tag, and a specific acknowledgement is sent back to the DCS by another tag, then the operator can check that the right acknowledgement has been received. The chances of both messages being corrupted (so that the system sees one override request but the operator sees another) are much smaller than the chances of either one message being corrupted. This also fits well with the need for confirmation described in clause 5 below.

2. The maintenance strategy and procedures need to be established before or during application engineering

Overrides should not be an afterthought or applied using uncontrolled unplanned methods. The application should include override programming.

3. While the PLC application program is being created it should be determined whether later on it will be allowed to override a particular signal.

All safety related I/O should have a maintenance override. This allows any point to be manipulated without causing spurious trips. Maintenance overrides should also be considered for I/O that does not
have a part in ESD or F&G actions, but which may cause a significant nuisance if accidentally tripped (e.g. control room alerts).

4. **Maintenance overrides are enabled for the whole PLC or a subsystem (process unit) by the DCS or other applicable authorized procedures (e.g. key switch, or password authorization).**

   **Note:** “enabling” the overrides permits, but does not necessarily turn on the overrides.

The Maintenance Override Enable is a secure way of ensuring that overrides can only be applied under the site authorization procedures. Since it is only one or a limited number of points, it is usually implemented with a panel keyswitch; this is secure and easily accessible. The statement above also allows for a DCS-based Enable. This application note advises that this should be considered only if there cannot be a keyswitch (e.g. for remote servicing); note the comments on override removal in clause 10 below.

Beyond requiring an Enable, the statement above is quite open. The Enable must be active to allow overrides to be set, but it can be implemented equally as a pre-requisite (must be set before overrides may be set) or as a final authorisation. There is no indication whether removing the Enable will also remove all maintenance overrides; however this is the common interpretation when the Enable is a pre-requisite (see clause 10 below).

5. **Because of organizational measures the operator should confirm the override condition.**

This prevents accidental application of overrides, providing a measure of security. This application note recommends that this is tied in with the suggested method for security in clause 1 above, where the command and acknowledgement must agree before the override can be applied.

This command – acknowledgement security has one disadvantage. If a command is sent, acknowledged and not confirmed, then the transaction remains permanently waiting and may be inadvertently completed in the future. This application note recommends that if an override is not acknowledged within a time window, the transaction is reset. The window is typically 30 seconds which allows for decision making time and communications/graphics delays.

6. **Direct overrides on inputs and outputs are not allowed (e.g., using clamps). Overrides have to be checked and implemented in relation to the application.**

This also does not allow variable locking. Overrides must be applied in a planned controlled manner.

7. **Multiple overrides in a PLC are allowed as long as only one override is used in a given safety related group.**

A safety related group is a set of I/O that has the same safety function. For example, do not allow the override of more than one of a set of three gas detectors in a group of three, or allow all high pressure safety alarms to be overridden for a vessel. Ensure that the process is still protected. The definition of ‘safety related group’ should be limited to all devices supplying the same or similar detection/protection to the same or associated safety parameter (level, pressure, fire, gas) in the same or in linked areas.

8. **The alarm shall not be overridden. It should always be clear that signals are in a maintenance condition.**

This statement is referring to the alarm raised at the DCS to indicate an active maintenance override and would make more sense if it followed clause 9. However, clauses 10 and 11 imply a similar requirement on the safety system that the operator must be still aware of the alarm state during the override of an input.

Keep the DCS alarm separate from the overridden safety point (do not use the safety point as the DCS alarm). The DCS alarm should still indicate when the input is in alarm. This will create nuisance alarms during maintenance, but the operators will be aware of the maintenance through the site permit system and can check if the alarm is genuine.

9. **The PLC alerts the operator (e.g., via the DCS) indicating the override condition. The operator will be warned until the override is removed.**

See clause 8. The presence of an override should be clear to the operator, e.g. by an alarm and an entry in the event log. The second clause is sometimes implemented as a returning alarm alerting the operator every new shift, so that new personnel do not have to hunt through hundreds of historical alarms to be aware of override status. Some sites demand that permits are suspended and overrides
are removed at the end of a shift; in this case the fresh re-application of an override will automatically generate a new alarm.

10. During the period of override proper operational measures have to be implemented to assure that the intervention can be removed again.

Removing the override should be guaranteed. A suggested solution is to ensure there is more than one way to remove overrides. In addition to individual override removal procedures, removing the Maintenance Override Enable could remove all overrides that it enables (see clause 4 above). This is a typical solution.

However, if the Maintenance Override Enable is based in the DCS, and the DCS fails with overrides still set, there is no remaining secure way to remove overrides. This is a good reason to make the Maintenance Override Enable a hardwired keyswitch instead of a DCS operation.

11. During the period of override proper operational measures have to be implemented to assure that the intervention into the process does not lead to unacceptable conditions.

This is linked to clause 7 above. If it is not possible to assure supervised safe operation within the safety system using multiple monitoring points for the same safety function, then manual monitoring (e.g. a firewatch) or a partial shutdown is needed; these ‘operational measures’ are outside of the system’s scope.

12. A program in the DCS checks regularly that no discrepancies exist between the override command signals from the DCS and the override activated signals received by the DCS from the PLC.

This is often overlooked. It is possible to cancel overrides using the Toolset online debugger; the DCS alarm and override indication will quietly disappear. Application changes could also cause the system to ‘forget’ applied overrides. In addition to the override control in the system, there must also be functions running in the DCS that highlight dropped or unauthorized overrides. This could either be via scripts that generate alarms or by graphical indications that make the discrepancy obvious, e.g. a partly coloured icon on an override summary page. A script is better because it doesn’t require the override summary page to be permanently visible.

13. The use of the maintenance override function should be documented on the DCS and on the programming environment if connected. The print-out should include:
   - the time stamp of start and end of maintenance override
   - the ID of the person who activated the maintenance override - maintenance engineer or operator
   - If the override information cannot be printed online (preferred), it should be entered in the work-permit
   - the tag name of the signal being overridden

This requires an event log; on newer systems this is more likely to be on disk than on paper. The DCS will record the currently logged-in username as the ‘ID’.
Summary of System Programming Recommendations

Do not use the Debugger ‘Locking’ facility as an override.

All safety related I/O should have a maintenance override. Maintenance overrides should also be considered for I/O that does not have a part in ESD or F&G actions, but will cause a nuisance if tripped.

The Maintenance Override Enable should be an easily accessible keyswitch. There may be more than one (for separate process units etc.). Overrides cannot be set if the Maintenance Override Enable is not set to Enable. Setting the Maintenance Override Enable to Disable will remove all associated overrides.

To set an override from the DCS,
- a command is sent from DCS to system by one tag,
- and a specific acknowledgement is sent back to the DCS by another tag.
- The operator must confirm that the acknowledgement matches the command, within a time window (30 seconds is recommended).
- The confirmation will set the override and clear the acknowledgement. It is reset after all override logic has run.
- The override is reported to the DCS on a separate point from the overridden safety point.

The confirmation may be one common point or individual points.
The above transaction and points are reset if a confirmation is not received within the time window.
Multiple overrides are allowed as long as only one override is used in a given safety related group.
Re-generate the DCS alarm if shift procedures do not remove and reinstate overrides on shift changes.
Override removal is effected by turning off the Maintenance Override Enable keyswitch or by clearing the command.
Override Programming Example

It is usual to make a function block which handles all conditioning, override logic and trip generation for one point, because it simplifies the generation of logic. An example has been written here to the list of recommendations above. This assumes that an input is being overridden. This is a lot simpler than most input conditioning blocks, because they also contain startup overrides etc. that add to complexity. Wherever possible, separate the logic requirement into different blocks to make each fully testable.

List of variables and parameters

- **DI1234**  Real digital input point  Active Low  Boolean input  From field device  DI
- **DT1234**  Trip signal to C&E Logic  Active Low  Boolean output  Safety signal  TRIP
- **DI1234_OS**  Override Status  Active High  Boolean output  Alarm to DCS  OS
- **DI1234_OR**  Override Request  Active High  Boolean input  DCS command  ORQ
- **DI1234_OA**  Override Acknowledge  Active High  Boolean output  Reply to DCS  OA
- **MO_CONF**  Common Confirm  Active High  Boolean input  From DCS  MC

The Common Confirm must be cleared when all override logic has completed.

- **MO_TIME**  Confirmation Timeout  30 seconds  Timer internal  Time limit for override request
- **G1234_OA**  Group Override Available  Active High  Boolean input  From safety group logic  GOA
- **MOENABLE**  Override Enable  Active High  Boolean input  From keyswitch  MOE
- **TRESET**  Trip Reset  Active High  Boolean input  Resets safety trip  RST

Function Block Parameters

It is recommended to explicitly declare each variable for each instance of the function block as shown here, rather than use corners to tidy a chained wire. This makes the program more reliable across online updates (see AN-T80009). Input conditioning function blocks are a common source of update problems because they contain internal variables for the override and trip states.

The function block may be written in Structured Text (ST) or Function Block Diagram (FBD). However, because it will involve program flow, ST is recommended. IF-THEN-ELSE-END_IF structures create easily readable and reliable code where different actions are required in different cases.
**Logic Chart**

This table presents the logic in a similar way to Cause and Effect charts. It is recommended to analyse the logic requirements of a function block before programming, because the resulting program will be more reliable and robust to unusual combinations of inputs and internal states. List all inputs, all previous values of inputs.

Blank boxes in the table indicate ‘don't care’ for that input.

<table>
<thead>
<tr>
<th>OR</th>
<th>MC</th>
<th>GOA</th>
<th>MOE</th>
<th>OS</th>
<th>OA</th>
<th>Timer</th>
<th>Implications</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Maintenance Override disable</td>
<td>Clear OS, zero/stop timer, clear OA</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Override requested and enabled</td>
<td>Set OA, start timer</td>
</tr>
<tr>
<td>T</td>
<td>F&gt;T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td></td>
<td>&lt;30s</td>
<td>Override confirmed</td>
<td>Clear OA, set OS, zero/stop timer</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Override cancelled</td>
<td>Clear OS, clear OA, zero/stop timer</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td>&gt;30s</td>
<td>Override timer ran out</td>
<td>Clear OA, zero/stop timer</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td>No group override available</td>
<td>Clear OA, zero/stop timer</td>
</tr>
</tbody>
</table>

The trip logic may be treated separately.

<table>
<thead>
<tr>
<th>DI</th>
<th>RS</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>Reset trip when input clear</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>Reset trip when Override set</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>Input tripped and no override</td>
</tr>
</tbody>
</table>
Program Code
(*
----------------------------------DETAILS----------------------------------
NAME           : DI_COND
REVISION       : 1
CREATED BY     : Nick Owens
DATE           : 18/03/08

LAST EDITED BY :
DATE           :
REVIEWED BY    :
DATE           :

TYPE           : Function Block
LANGUAGE       : Structured Text

----------------------------------DESCRIPTION----------------------------------
Digital input processing function block with Maintenance Override for AN-T80016.

This function block includes the following functions:
Detection of input trip, resettable when clear or overridden. Maintenance override: non-timed, with an acknowledge/confirm handshake to set, separate override enable, and limit to the maximum current overrides in a safety group.
Detailed descriptions of each function are given in the code comments below.

----------------------------------VARIABLES----------------------------------
Note: GLOBAL variables must be added to the project global dictionary as they are not owned by the block!

GLOBAL:

None.
(MC, MOE and RST could be made globals to save on input connections, since all instances will use the same variables.)

LOCAL: defined in function block’s local dictionary
(Signals are active in the TRUE state unless stated otherwise)

Last_ORQ  Boolean, Internal Override Request at previous scan for edge detection
Last_MC   Boolean, Internal Override Confirm at previous scan for edge detection
OR_Timer  Timer, Internal Time limit on DCS Override Request
Int_OS    Boolean, Internal Internal copy of Override Status
Int_OA    Boolean, Internal Internal copy of Override Acknowledge

Int_OS is used as an internal copy of OS (Override Status) since function block outputs can't be read from. Int_OA is an internal copy of OA for the same reason.
**Parameters**

- **DI** (Boolean): Input signal from field device (Active False)
- **ORQ** (Boolean): DCS Override Request
- **MC** (Boolean): DCS common Override Confirm
- **GOA** (Boolean): Group Override Available from safety logic
- **MOE** (Boolean): Maintenance Override Enable from keyswitch
- **RST** (Boolean): Trip Reset

**Return**

- **TRIP** (Boolean): Trip output to Cause and Effect logic (Active False)
- **OS** (Boolean): Override Status to DCS
- **OA** (Boolean): Override Acknowledge to DCS

---

*Maintenance override*

This is a non-timed override that prevents the alarm trip from going false.

To set an override from the DCS:
- A command is sent from DCS to system on tag ORQ.
- And a specific acknowledgement is sent back to the DCS by tag OA.
- The operator must confirm that the acknowledgement matches the command, within a time window, counted by OR_Timer (here set to 30 seconds). This sends a Confirm signal on tag MC which is common to all instances of this function block.
- MC is reset after all override logic has run, after the last instance of this function block.
- The confirmation will set the override (effectively Int_OS) and clear the acknowledgement OA, as long as there are not too many overrides already set in the safety group (GOA) and maintenance overrides are enabled (MOE).
- The override is reported to the DCS on tag OS.

The above transaction and points are reset if a confirmation is not received within the time window.

Override removal is effected by turning off the Maintenance Override Enable keyswitch or by clearing the command.

(*)

(* If an override request has arrived and a group override is available and maintenance overrides are enabled, acknowledge the request and start the timeout *)

```plaintext
IF (ORQ AND NOT Last_ORQ) AND GOA AND MOE THEN
    Int_OA := True;
    OR_Timer := t#0s;
    TSTART(OR_Timer);
END_IF;
```

(* If an override confirm has arrived and an override request is in progress and
the time limit has not been reached and a group override is available and
maintenance overrides are enabled, set the override and clear down the request.
This logic checks that the override request went True on a previous scan, to
avoid setting overrides without acknowledgement *)

IF (ORQ AND Last_ORQ) AND Int_OA AND (MC AND NOT Last_MC) AND GOA AND MOE AND
    NOT Int_OS AND OR_Timer < t#30s
THEN

    Int_OS := True;
    TSTOP(OR_Timer);
    OR_Timer := t#0s;
    Int_OA := False;

END_IF;

(* If there is no group override available, abort any override request *)
(* If an override request is in progress and the timer has run out, clear down the
override request *)

IF NOT GOA OR (Int_OA AND OR_Timer >= t#30s) THEN

    TSTOP(OR_Timer);
    OR_Timer := t#0s;
    Int_OA := False;

END_IF;

(* If the Maintenance Override Enable keyswitch is set to Disable, or the
override request has been removed, remove the override and abort any override
request *)

IF NOT MOE OR NOT ORQ THEN

    Int_OS := False;
    TSTOP(OR_Timer);
    OR_Timer := t#0s;
    Int_OA := False;

END_IF;

(* Report override status and acknowledge to DCS *)

OS := Int_OS;
OA := Int_OA;

(* Record state of Override Request OR and Override Confirm MC for next scan
to detect rising/falling edges *)

Last_ORQ := ORQ;
Last_MC := MC;

(*
Safety Trip

The safety signal is set active (False) when the real input is active (False) and the maintenance override is not active. The safety signal is cleared (True) when Reset is active (True) and either the input is not active or the Maintenance Override is active.

*)

IF RST AND (DI OR Int_OS) THEN
    TRIP := True; (* Trip can be reset *)
END_IF;

IF NOT DI AND NOT Int_OS THEN
    TRIP := False; (* Input tripped and not overridden *)
END_IF;
ICS Triplex technologies and services are available worldwide.

Hall Road
Maldon Essex
CM9 4LA
UK
Tel: +44 1621 854444
Fax: +44 1621 851531

www.icstriplex.com
For technical support email: support@icstriplex.com
Sales enquiries: sales@icstriplex.com

Technology Driven Customer Led