Best practices for Concept Applications

Programming a PLC based system operates at two levels
Level 1: Familiarity with programming configuration and language syntax to create a logically correct program

Level 2: dynamics of system development:
- Being aware of the interaction of the programming package within the user's Windows operating system.
- Knowing the operating philosophy of the PLC platform in use.

The following best practices are intended to improve project design and operation. It is assumed that the systems engineer is familiar with level 1 requirements.

Schneider also GREATLY recommends that you read, understand, and implement the issues as described in the following files available in your Concept installation folder:
- README.DOC   General information concerning Concept 2.5
- WHATSNEW.DOC   SR1-Release specific information
- UPGRADE.DOC   Issues related to upgrading to Concept 2.5
- INFOSR2x.DOC   Information regarding installing, new features and known issues of the service release, Firmware/Executive Version Table

If you do not understand anything in this document please contact the suitable Schneider technical support staff:

        Seligenstadt, Germany
        Phone: +49 (0) 6182 - 81 29 00
                      01805 – 75 35 75 (for call within Germany)
        E-mail: techsupport.seligenstadt@modicon.com

        North Andover, USA
        Phone: +1 978 794 0800
        Or: (800) 468 5342 Option 1
        E-mail: customercentral@modicon.com

        France
        Phone: Your Schneider Agency

Contents:
1. Variables
2. Structured Data Types
3. Sections
4. Languages
5. Derived Function Blocks
6. Organizing The Project
7. Organizing Multi-User Work
8. Initial Values and State RAM Values
9. Memory Size and Memory Map
10. Concept Converter
11. EFB ('C') Tool Kit
12. Animation

Appendixes
A: The states EQUAL, MODIFIED, NOT EQUAL
B: Loop Control in ST.

Best practices for Concept Applications

1. Variables
- You do not need to store the same data in different variables (unless other program requirements exist). Variables occupy memory and moving data takes CPU time. *Hint:* The analyser is able to detect variable overlap for located variables.
- Use located variables for discrete (Boolean) variables only if they have to be interfaced to I/O, other PLCs or HMI devices. Otherwise use unlocated variables. Located variables are the traditional 0x, 1x, 3x and 4x Modbus addresses, thus must be used for I/O. Accesses by HMI can use the full features of Modbus to maximize data traffic. However, unlocated variables operate faster within the CPU than located variables.
- Concept allows multi-assigned output to variables. Data can thus update within a scan. Be aware that the last item executed this scan will update to on-screen animations. The analyser will report (sometimes many) warnings for multi-assignments.

2. Structured Data Types
- Organize the structured data types into local and global data types. This gives a clean structure, and avoids problems when combining subprojects.
- Global data types may use data types distributed along with the EFB-libraries, Local data types may use global data types.
- Please note that only one dty-file with structured data types is allowed in a global/local DFB folder.
- Optimize data structures; use only components that are really needed. This saves memory and improves scantime.
- All structures (Located or Unlocated) are BYTE (not WORD) aligned.
  Hint: Care is required when laying out structures so that they map on to the normal WORD boundaries of 4xxxx addresses. This is essential for correct access to data by HMI devices.

  Hint: It is not necessary to develop DFBs to move data structures or arrays. The IEC MOVE EFB accepts variables of any data type at its input or output.

3. Sections
- Do not partition your project into too many small sections. Section organization consumes CPU memory and time. Keep a balance between the number of sections and their size. Currently there is a limit of 1600 IEC-sections including transition sections.
- Do not make a section too large. A large section may prevent a 'download changes' if your program size is getting close to the memory limit of a CPU The Memory Prediction feature of Concept V2.5 can be used to show the logic size of a section.
  Hint: ST/IL sections can be particularly affected here. The graphical editors FBD/LD/SFC have boundary limits (but can still be large), while the text editors ST/IL do not. Use the Section enable/disable feature according to your application model. Disabled sections are not executed hence scan time will improve significantly.
  Hint: A 'download changes' with too many sections at once may not be possible. All new sections are held in one temporary area in the PLC until the download is finished; then they get activated together. Too many new sections together may exceed memory limits.
  Hint: The above problem can be avoided by downloading a few of your modified sections at a time. This is called 'Sequential Download'. You select a normal 'download changes' but de-select several of the modified sections. As downloaded sections match the CPU, they will disappear from the selection list. Note: sections that logically belong together should be downloaded together.

Best practices for Concept Applications

3. Sections

- Pay attention to section execution order. The execution order determines in which sequence sections get executed during a scan. A wrong sequence will cause the control application to compute old data (new data will be delayed one scan).
  Hint: If any particular logic section is used for the overall application co-ordination, this section must be executed first.
  Hint: If a section contains a SFC CNTRL EFB, then this should be executed before the SFC section it controls.

---

4. Languages

Concept is a software toolbox. The control languages (SFC, FBD, LD, ST, IL, 984LL and 'C') represent powerful tools for different application purposes. As a rule of thumb the following guidelines apply:

SFC	Sequential control and batch recipe procedure
FBD	Continuous control and/or discrete control
LD	Discrete control
ST	Equations, table manipulation, complex algorithms
IL	Fastest possible logic execution
984LL	Traditional 984 logic with segments – sometimes necessary for interrupt based logic (for high speed machines)

- Note: Structured Text allows you to create loops. While executing loops, array boundaries are checked and there is an impact on scan time. This option can be chosen in the Code Generation option "Enable Loop Control"
  Example:
  ```
  FOR I:=1 TO 100 DO
  A[I]:=I;              (* boundary checking *)
  END_FOR;
  ```
  Or
  ```
  WHILE (I=I) DO
  A[I] := I;           (* infinite loop, loop control terminates loop *)
  END_WHILE;
  ```

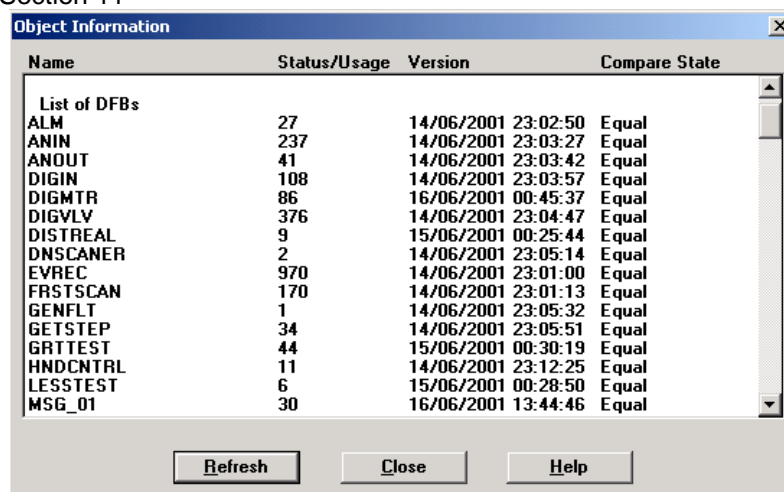 For more detail– see Concept file README.DOC or appendix B

  It is recommended to choose option "Enable Loop Control" only during the development phase.

- Use loops carefully.
  Hint: Structured Text execution can be significantly improved by choosing Fastest Code in the Code Generation option. This will generate inline code for all arithmetic and logic operations and overflow, underflow or division by zero will not be detected.

- Investigate design requirements versus maintenance requirements (e.g. Ladder Diagram is easy to maintain but it is not the most productive tools for all parts of the application). Try to combine the best of both worlds: create DFBs in FBD, ST or LD for high productivity and use those DFBs within Ladder Diagram for maintainability.

Best practices for Concept Applications

Schneider Electric

5. Derived Function Blocks

DFBs hide a big portion of your application code and as a result programs are much easier to understand.

- Configure only inputs and outputs that are really used by your application. Inputs and outputs consume memory and increase runtime.
- Do not use unnecessarily complex structures for inputs and outputs or variables. If you are only using a few components of a structured data variable then only pass those components as inputs.
- Often used DFBs (used more than 100 to 200 times) should be converted to EFBs (using the Concept EFB 'C' Toolkit). EFBs written in 'C' normally execute faster and consume less memory than DFBs. The following example shows the usage of DFBs in a real project. The larger usages here should be converted to EFBs. See Section 11

Name	Status/Usage	Version	Compare State
List of DFBs			
ALM	27	14/06/2001 23:02:50	Equal
ANIN	237	14/06/2001 23:03:27	Equal
ANOUT	41	14/06/2001 23:03:42	Equal
DIGIN	108	14/06/2001 23:03:57	Equal
DIGMTR	86	16/06/2001 00:45:37	Equal
DIGVLV	376	14/06/2001 23:04:47	Equal
DISTREAL	9	15/06/2001 00:25:44	Equal
DNSCANER	2	14/06/2001 23:05:14	Equal
EVREC	970	14/06/2001 23:01:00	Equal
FRSTSCAN	170	14/06/2001 23:01:13	Equal
GENFLT	1	14/06/2001 23:05:32	Equal
GETSTEP	34	14/06/2001 23:05:51	Equal
GRTTEST	44	15/06/2001 00:30:19	Equal
HNDCNTRL	11	14/06/2001 23:12:25	Equal
LESSTEST	6	15/06/2001 00:28:50	Equal
MSG_01	30	16/06/2001 13:44:46	Equal

Object Information — Refresh | Close | Help

Hint: Prototype EFBs as DFBs and export them to ST to get the framework for the 'C' code body.

- It is better to design several specialized DFBs than one complex universal DFB. This saves scantime and memory.
- In order to avoid a full download when modifying an existing DFB.
  If you have a DFB which, is being used in a running process and you must make modifications to the DFB, modify per following steps:
  1. Using Concept make sure that the project is EQUAL with the controller.
  2. Disconnect and close Concept.
  3. Open the desired DFB using the DFB Editor.
  4. Make the desired modifications to the DFB's logic.
  5. Then use File Saves As and save the changes under a new DFB name.
  6. Start Concept with your project click on the DFB that you would like to change
  7. Click on Objects pull-down menu choose Replace FFBs… and choose the new DFB. You must perform step 7 for every instance of the DFB in your project.

- Note: The following are some of the implications to consider when modifying a DFB:
  - If you add a new input/output pin in between existing pins (with the DFB editor) then in a Concept program section, existing logic connections below the new pin may or may not be removed depending on the new datatype that gets presented at a given pin position.
  - If you change the datatype of an existing pin then logic will be removed
- The easiest way to add a pin to a DFB is at the bottom. This retains all existing logic connections while adding the new pin.
- If you want to preserve the possibility to convert a DFB to a EFB without much effort keep in mind that in EFBs in-out pins can be located only at the top.

Best practices for Concept Applications

6. Organizing the Project

- There are three different places where you can locate global DFBs (please refer to Help)
  - First, if the 'PreserveGlobalDFB' switch in the Concept.ini file is set, the directory <projectdir>.GLB is searched. If the folder exists the global DFBs are assumed to be here. Using this option every project can use its own 'global' DFB folder without affecting other projects.
  - Second, if the switch is reset or if there is no 'GLB' directory, the DFBs are assumed to be in the folder that the entry ''GlobalDFBPath'' refers to in the Concept.ini file. Using this option DFBs located on a network drive can be shared by several Concept applications.
  - Third, if the entry ''GlobalDFBPath'' is empty or refers to a non-existent folder the DFBs are assumed to be in the <Concept Install Folder>. This was known as the global folder in Concept 2.2.
- When importing projects from ASCII files via the Concept Converter or via the 'Archiving' feature, you should consider setting the 'PreserveGlobalDFB' switch and creating a 'GLB' folder manually so as to not override global DFBs and data types used by other projects. Be careful if you create more than one dty-file in a DFB folder – only one dty-file in a DFB folder is allowed so the other dty-file(s) will be ignored.
- Change used structured data types as little as possible because this means a lot of internal changes in DFBs, especially in text sections. Use the 'Update data types in nested DFBs' feature to make the project consistent again. Be aware that you might change DFBs used by other projects.
- Be aware that in contrast to Concept 2.2 declared but unused variables consume memory from the 'Global Data' area in Concept 2.5.
- From time to time execute (at least once) the 'optimize' feature to compress wasted data or code memory that is not re-usable.
- Sometimes a corrupted database can be repaired by deleting the key-files (*.q1 and *.q2) of the database. They will be re-generated the next time the database is opened.
- In most cases a corrupted database can be repaired by an export/import with the Concept Converter (do not use the mode 'Re-connect to Equal'). If you fail to export the project because of analysing errors try an export without exporting DFBs. This does not require a successful analysis.
- Deleting an item in a program section is a useful test of the database's consistency. Deleting an item should also correctly remove other connected logic items. You can then use 'undo delete' to return to your original logic.
  Hint: If you get a database error after the above delete test, use Concept Converter to export (not in re-connect to Equal-mode) and them re-import your project. The analyser will clean up your database to assure correct logic operation.

7. Organizing Multi-User Work	• Subproject variables can be merged by exporting them from the subprojects and importing them into the combined project. Logic can be merged by sequentially copying sections to the clipboard and saving them in the clipboard format to a file(s). This makes it unnecessary to constantly switch between the subprojects.
	• Another feature of Concept – Section Import/Export (creating xx.SEC files) can be used. The advantage with this method is that variables and DFBs come with the export process. A restriction exists though in that the source and target projects MUST have the same environment (i.e. same global/local DFBs and DTY files).
	• When working with several projects sharing the same set of DFBs and data types this common environment should be located on a server with access restrictions (use the ''GlobalDFBPath" entry in the concept.ini file).
	• When changing data types and/or nested DFBs first make the set of DFBs consistent by opening/closing the DFBs in a hierarchical bottom-up order.
	• Please see section 10 below about implications of transferring files between PCs.

8. Initial Values and State RAM Values	Located Variables
	• Initial values of located variables are downloaded to/uploaded from the PLC only if the corresponding check boxes in the download/upload dialog are checked (State RAM & Initial Values). Otherwise no State RAM is downloaded/uploaded. *Note* Only 4x variables can have initial values.
	State RAM
	• State RAM values are uploaded from the PLC by checking the State RAM check box in the upload dialog. Only 0x, 1x (along with their 'disable' state), 4x and 6x memory can be uploaded.
	• The uploaded State RAM values are stored in a database table and are persistent. The Concept Converter retains these values during an ASCII export/import.
	• The values can be visualized either by the RDE if Concept is offline or by looking into the ASCII export file. 4x State RAM values can be changed by the RDE for the next download if Concept is offline.
	• If the database table contains State RAM values (generated by a former upload of State RAM or by manual changes) State RAM values can be downloaded.
	Unlocated Variables
	• Initial values are downloaded generally by the download/download changes which follows the creation of the variable. Once the download is executed a change of the initial value or a new initial value will not affect the PLC until the next full download.

9. Memory Size and Memory Map

- The values for "x K logic" and "y K state" in the PLC selection dialog mean: "y K state" is roughly the memory available for registers, expressed in K ladder logic instructions (1K ladder logic instructions = 20Kbytes). "x K logic" is the memory available for the control application; this value is also expressed in K ladder logic. If the application does not contain 984-ladder logic, all program memory is available for the IEC program. The amount equals roughly the number of "available bytes" in the loadable instruction configuration dialog.
- Please note that the above is only true for the loadable base controllers. For the 32-bit controllers you would have to use the IEC Heap Size (Concept 2.2: IEC Usable Memory Size) to adjust the user logic area for IEC.
- Configure (in memory map) only as many registers as needed (plus 10% reserve because configuration changes are only possible while PLC is stopped) for the application. This frees up memory and increases scan time.
- The IEC Global Data area (configured in the PLC-selection dialog) should be adapted to the project needs. This sets the maximum data area available for unlocated variables. As a part of the IEC Heap (the totally available memory area for IEC-program and –data) it consumes IEC logic space.
  Note: Keep in mind that unused IEC Heap is temporarily used for online changes.
  Hint: Configure the IEC heap after project start-up so that 20% to 30% reserves are available for online changes. Note that when using the IEC simulator that the displayed values for memory and scan time merely represent values that could be expected using a real controller.

Best practices for Concept Applications

10. Concept Converter

The following applies to the Concept Converter when not working with the 'Re-connect to Equal' feature:

- Users may use the Concept Converter to move/copy Concept projects between PCs. Be aware that when importing Concept files with the Concept Converter that the internal version date/time stamps of the program are updated. This causes the following effects:
  - Your Concept project will become NOT-EQUAL.
  - Note: A particular xx.ASC file will import into different PCs differently. There are no constraints within Concept to guarantee that the Concept Converter program on 2 PCs will import variables and logic to exactly the same Concept memory locations. If programmers have used the Concept Converter to set up project files on 2 PCs, then files MUST NOT be directly copied between these PCs if you want to retain online EQUAL status.
  - *Hint:* Use the Concept Converter as needed to place files on one PC. At this point, users can directly copy files between PCs and EQUAL status will be preserved (if DFB files have the same internal version).
  - *Hint::* Use a zip program to transfer files between PCs
- Export and re-import your Concept project at the end of the commissioning phase. This minimizes the used memory by deleting code fragments and improves scan time.
  - *Hint:* Use a site Master PC as the single point of updating a set of site projects. Then distribute these files as the master set.
- Optimize your Concept project at the end of the commissioning phase (menu item 'optimize project'). Deleting objects and online changes create gaps in the symbol image. "Optimize project" eliminates those gaps.
- Close all other applications and disable your screen saver when using the Concept Converter. Problems have been reported occasionally. The converter will also execute significantly faster.

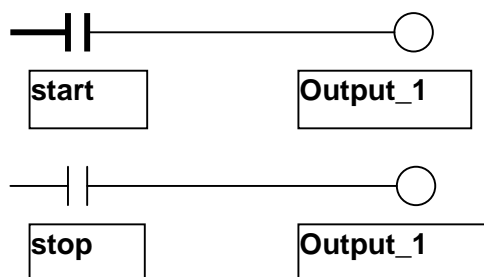Note: Please note that project Export/Import or Project Optimization will force a full download.

Converting with the 'Re-connect to Equal' feature:
- In this mode the Concept Converter does not optimize the database.
- The ASCII file can be used for archiving purposes.
- When importing the ASCII file it is recommended to locate the global DFBs in the 'GLB' folder.

---

11. EFB ('C') Tool Kit

- The EFB-Toolkit Concept 2.1 and higher works with Borland V5.0 Compiler & Borland Turbo 5.0 Assembler (TASM.EXE AND TASM32.EXE).
- Concept comes with many built in EFBs. The user may design more EFBs to meet local requirements. A good source for EFB designs is the set of site DFBs. See section 5. Above for more details on this.
- If EFBs (Elementary Function Blocks) have to be installed on a different PC, they can be copied from their folder under Concept\lib on the source PC, into the same folder name in Concept\lib on the destination PC. The following files have to be deleted to activate the new EFB library:
  - Concept\LIB\~AITMP.*

Best practices for Concept Applications

12. Animation
- Animating within Concept takes a lot of PC resources (with regard to the background real-time tasks involved). Problems have been known to occur during and up to some time after data animation. After animating, and from time to time, you should save your work, exit Concept, and restart to ensure Window's resources are maintained properly.

- Concept allows writing to coils and numerical variables more than once within IEC sections. This might lead to "unusual" animation (power flow) behaviour, because states of variables are sent to the programmer (PC) always at the end of a PLC scan. The following example illustrates what can happen:
  Input "Start" is true and as a result "Output_1" is true. "Output_1" is set to false in the next rung ("Stop" is false). Because the animation update is performed at the end of scan, "Output_1" is shown as false, even though "Start" is true.
  Please keep in mind that the RDE would always indicate Output_1 as OFF.

```
    ─┤ ├──────────────( )
    ┌──────┐         ┌──────────┐
    │start │         │Output_1  │
    └──────┘         └──────────┘

    ──┤ ├─────────────( )
    ┌──────┐         ┌──────────┐
    │stop  │         │Output_1  │
    └──────┘         └──────────┘
```

- A similar animation effect can be observed with DFBs. If an output from a DFB is connected to a multi-assigned variable, then animated data that the user sees may not match data from the same pin internally within the DFB (as seen when the DFB is 'Refined' and animated). Data as seen by the user when a program section is animated is updated to the PC screen at the end of each program scan – not the data that might exist on a given DFB pin in its order of execution. Data as seen by the user when a DFB is 'Refined' and animated is read from the DFB (even though it is updated on the PC screen at the end of the program scan).
  Hint: This data display effect can be used as a feature to display intermediate data along a calculation path.

Best practices for Concept Applications

Appendix A: The states DIRECT, EQUAL, MODIFIED, NOT EQUAL

The conditions DIRECT-, EQUAL, MODIFIED, NOT EQUAL, are all on-line states. A status line in the lower right hand corner of the display will indicate which state you are in.

These states are controlled by 3 sets of version time stamps. Two of these versions can be seen under Project, Properties – the program timestamp, and the assumed PLC timestamp. The 3rd version is the actual PLC timestamp in the CPU – available when Connected under 'Online', 'Object Information'.

The DFB editor can also show version time stamp information under the DFB properties.

DIRECT
This represents going on-line without an open Project. This will allow the user the same on-line capabilities as being NOT EQUAL except Download will not be allowed.

EQUAL
This means your project files have the same version time stamps as the one in the PLC. The following conditions are all satisfied:
- The PLC configuration image is identical between PLC and program files.
- The checksum for LL984 is identical.
- The version time stamps of the DFBs and EFBs used in your program are identical.
- The version time stamp of the program is identical.
- User defined datatypes are identical to the last time you opened the Project.

MODIFIED
This means that your project assumed PLC timestamp is identical to the actual PLC time stamp, while the Program time stamp is different.
- The other conditions above remain unchanged, while the version time stamp of the program is different.

To finish the modification and make the Project and PLC EQUAL, do a 'Download Changes'.

Concept V2.1 will give the following limitation: If the user attempts to exit while in a MODIFIED state, Concept will give a warning that the next time the user goes on-line with this Project, it will be in a NOT EQUAL state.

Beginning with V2.2 Concept will allow the user to be modified, save his changes, exit Concept, re-open Concept, open the project, and be Modified.

NOT EQUAL
This occurs if the actual PLC time stamp is different from the assumed PLC timestamp. The assumed PLC timestamp is changed inside Concept upon any of the following:
- The PLC configuration image is different between PLC and program files.
- The checksum for LL984 is different.
- The version time stamps of any of the DFBs and EFBs used in your program are different.
- User defined datatypes have changed from the last time you opened the Project.
- Note: If a Download Changes is aborted, then NOT EQUAL is displayed. The timestamps are actually still in the MODIFIED state.

Note: If you are MODIFIED, download changes to the PLC, then exit Concept without saving, you will then return to Concept NOT EQUAL. You will have downloaded a new version time stamp to the PLC without saving this to your project files.

Note: Only certain on-line functions will be available when in a NOT EQUAL status.

Ways of getting out of NOT EQUAL status.
1. If Upload information is available in the PLC, then perform an Upload.

2. Stop the PLC and perform a complete Download of the user project. At this point, all three time stamps are the same.
   Note: If the user is running multiple PCs with the Project files to the same PLC on all of them, he or she should follow a strict version control program to assure that no PC contains a NOT EQUAL Project. The moment any changes to the PLC program are made and downloaded with one Project the PLC actual time stamp will then be as per that PC's project. The other PC projects will then have different time stamps, i.e. be NOT EQUAL.

3. Note above that one mode of NOT EQUAL occurs if a Download Changes is aborted. The assumed PLC time stamp is still the same as the actual PLC time stamp. There are other internal Concept flags involved that are reporting the (fictitious) NOT EQUAL state. The user can cause a return to the correct MODIFIED state by the following means. In both cases Concept will examine the time stamps and update the flags:
   - Immediately after the aborted Download Changes go into 'Online', Object Information'.
   - Disconnect, and then Reconnect.

4. For users of 984LL only, an upload of the ladder code can be performed to correct the NOT EQUAL status.

Appendix B: Loop Control in ST

Loop control consists of a software watchdog for infinite loops in ST sections. The watchdog supervises whether loops are exited within a certain time-span. The supervised time depends on the manually defined hardware watchdog time. It is recommended that you enable the loop control only during the commissioning phase as it will increase the normal execution time and code.

The loop control typically affects the execution of ST sections if ST loops contain time-consuming calculations or infinite loops. The maximum execution time is limited to approximately 80% of the hardware watchdog time and can be adapted by adjusting the hardware watchdog time in the configuration. The measurement of the execution time starts at the beginning of the program scan and comprises all sections executed before the concerned section.
- If the time limit is exceeded while executing a ST loop, the execution of the loop is stopped - the remaining code of that section is skipped and an online event is generated.
- If the time limit is exceeded before entering a ST loop, that loop is executed only once and the remaining section code is skipped.

ST sections in DFBs are also supervised by the loop control. In this case the ST section of the DFB is terminated.

All other sections (even ST sections) which follow according to the execution order are processed as usual (except loops and the ST code following the loop) if the remaining 20% execution time is sufficient. If the scan cannot be completed within the remaining 20% of the hardware watchdog time, the PLC will be stopped by the hardware watchdog.

Thus the loop control is able to terminate a critical ST section before the controlled process is jeopardised, but triggering the hardware watchdog depends finally on the succeeding application code and the position of critical ST sections in the execution order. While the stoppage of the PLC through the hardware watchdog can be avoided in certain circumstances by sophisticated arrangement of critical ST sections (at the end of the scan) it is usually not possible to do the same concerning critical DFBs.

If loop control in ST sections is switched off control code is left out. Instead the PLC is stopped directly by the hardware watchdog if the scan time exceeds the limit.