

CALENDAR V17

Creating and Editing Calendars

Comments... Suggestions... Corrections...

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:

manuals@ts.fujitsu.com

Certified documentation according to DIN EN ISO 9001:2008

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2008.

cognitas. Gesellschaft für Technik-Dokumentation mbH

www.cognitas.de

Copyright and Trademarks

Copyright © Fujitsu Technology Solutions GmbH 2010.

All rights reserved.

Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers

i On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions. This document is a new edition of an earlier manual for a product version which was released a considerable time ago in which changes have been made to the subject matter. Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions. Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format ...@ts.fujitsu.com. The Internet pages of Fujitsu Technology Solutions are available at [http://ts.fujitsu.com/...](http://ts.fujitsu.com/)

Contents

1	Preface	7
1.1	Target group	7
1.2	Summary of contents	8
2	Structure and installation	11
2.1	The structure of a calendar	11
2.1.1	Basic information	11
2.1.2	Calendar days	13
2.1.3	Symbolic dates: SYMDATs	14
2.1.4	Holidays	17
2.2	Calendar files	19
2.2.1	Naming conventions for calendar files	19
2.2.2	Security mechanisms for calendar files	19
2.2.3	File attributes	20
2.3	Installation	20
2.3.1	Components	20
2.3.2	Installing the subsystem	21
3	Dialog interface: the calendar editor	23
3.1	Overview of functions	23
3.2	Operating the calendar editor	24
3.2.1	Starting and terminating	25
3.2.2	Overview of masks	27
3.2.3	General mask layout	28
3.2.4	Editor functions	29

3.2.5	Description of the masks	33
3.2.5.1	Mask C000: Main Main menu of the calendar editor	34
3.2.5.2	Mask C010: Edit Select functions	37
3.2.5.3	Mask B020: Basic Information Output and edit basic information	42
3.2.5.4	Mask D010: List of Days List calendar days and mark them for selection	46
3.2.5.5	Mask D020: Day Information Edit calendar days, i.e. day information	49
3.2.5.6	Mask S010: List of Symbolic Dates Output SYMDAT list / mark for selection	53
3.2.5.7	Mask S020: Symbolic Date Information Output and edit SYMDAT information	56
3.2.5.8	Mask H010: List of Holidays List holidays and mark for selection	61
3.2.5.9	Mask H020: Holiday Information Output and edit holiday information	64
4	Program interface	69
4.1	Format overview	71
4.2	Description of operands	74
4.3	Data areas for functions	81
4.4	Return codes	84
4.5	Example	87
4.6	Individual formats	94
4.6.1	*AKTHOL Activate holiday	95
4.6.2	*CLSECAL Close calendar	97
4.6.3	*CREACAL Create calendar	99
4.6.4	*CRECHOL Create cyclic holiday	103
4.6.5	*CRECSYM Create cyclic SYMDAT	105
4.6.6	*CRENHOL Create non-cyclic holiday	110
4.6.7	*CRENSYM Create non-cyclic SYMDAT	113
4.6.8	*DEAKHOL Deactivate holiday	117
4.6.9	*DELHOL Delete holiday	119
4.6.10	*DELSYM Delete SYMDAT	121
4.6.11	*MODBAS Modify basic information	123
4.6.12	*MODCHOL Modify cyclic holiday	127

4.6.13	*MODDAY	Modify calendar day	129
4.6.14	*MODCSYM	Modify cyclic SYMDAT	133
4.6.15	*MODNHOL	Modify non-cyclic holiday	138
4.6.16	*MODNSYM	Modify non-cyclic SYMDAT	142
4.6.17	*OPENCAL	Open calendar	147
4.6.18	*SAVECAL	Save calendar	150
4.6.19	*SHBAS	Output basic information	152
4.6.20	*SHDAYHL	Output calendar day with assigned holidays	155
4.6.21	*SHDAYIN	Output calendar day with assigned SYMDATs	159
4.6.22	*SHHOLIN	Output holiday with assigned calendar days	164
4.6.23	*SHLODAY	Output list of calendar days	169
4.6.24	*SHLOHOL	Output list of holidays	174
4.6.25	*SHLOSYM	Output list of SYMDATs	178
4.6.26	*SHSYMIN	Output SYMDAT with assigned calendar days	183
4.6.27	*SHNESTM	Output next time for a set of SYMDATs	188
4.7	Parameter list		193
4.8	Macro syntax		198
4.8.1	Macro call format		198
4.8.2	Elements of the metasyntax		199
4.8.3	Data types of the operand values		200
4.8.4	Suffixes for data types		200
4.8.5	Data types of variables		201
4.8.6	MF operand		201
4.9	C header file (calendr.h)		204
5	Command interface		217
<hr/>			
	SHOW-CALENDAR		
	Request information from calendar file		217
	Function		217
	Format		218
	Operands		219
	Output formats		225
	References		231
	Index		233

1 Preface

The CALENDAR subsystem offers a central calendar function for BS2000.

The calendar system allows the user to create a new calendar according to the data to be processed, or to read and edit existing calendars.

A calendar is a file (calendar file) that contains information on workdays, working hours, free days, holidays, additional symbolic dates (SYMDATs), etc. for a defined period of time.

CALENDAR provides basic structures such as a standard working week and the ongoing creation of days within the defined period. Similarly, the holidays for the calendar file can be taken from a file called the holiday file.

A user who creates or modifies a calendar file essentially defines calendar data by

- assigning information to calendar days
- defining holidays
- defining symbolic dates (SYMDATs).

The user can access the data in a calendar file using three different interfaces:

- The **dialog interface** is used to create, modify, or output the data of a calendar file using the calendar editor. FHS masks are used for editing.
- The **program interface** is used to create, edit and output the data of a calendar file on the program level (Assembler, C).
- The **command interface** is used to request calendar data for output. The data is output to SYSOUT / SYSLST and, if required by the user, in S variables. This means that the data from calendar files can also be used in SDF-P procedures.

1.1 Target group

This manual is aimed at BS2000 systems support staff and all users who want to implement a central calendar function.

1.2 Summary of contents

This manual comprises five chapters with the following contents:

Chapter 1, *Preface*

Contains a brief description of the CALENDAR subsystem, names the target readership, and explains the structure of the manual.

Chapter 2, *Structure and installation*

Provides information on the structure and elements of a calendar and describes the naming conventions, security mechanisms and file attributes of a calendar file. It also offers information on installation and on the files required to implement CALENDAR.

Chapter 3, *Dialog interface: the calendar editor*

Describes the “calendar editor” utility which is used (via masks) to read and modify the calendar data from existing calendars and to create new calendars.

Chapter 4, *Program interface*

Contains a detailed description of the CALENDR macro as well as the parameter list and a C header file with comments. All individual formats, sorted alphabetically according to function calls, are described following the format overview of the macro with an operand description and a list of return codes. This chapter also provides a description of the macro syntax and an example of the CALENDR macro.

Chapter 5, *Command interface:*

Describes the SHOW-CALENDAR command, which is used to request information from the specified calendar for output. The most important output formats are illustrated.

A section on related publications and an index can be found at the back of the manual.

README file

Information on any functional changes and additions to the current product version can be found in the product-specific README file.

You will find the README file on your BS2000 computer under the file name SYSRME.CALENDAR.<version>.E. The user ID under which the README file is cataloged can be obtained from your system administrator. You can view the README file using the SHOW-FILE command or any editor, and print it out on a standard printer using the following command:

```
/PRINT-DOCUMENT SYSRME.CALENDAR.<version>.E, LINE-SPACING=*BY-EBCDIC-CONTROL
```

2 Structure and installation

2.1 The structure of a calendar

The information structure in a calendar is made up of four main elements:

- basic information (calendar limits and standard working week)
- calendar days
- symbolic dates (SYMDATs)
- holidays

2.1.1 Basic information

The definition of the basic information “calendar limits” and “standard working week” is an essential prerequisite for all subsequent steps in creating a calendar.

The calendar limits

The time span of a calendar is defined by the first calendar day as the lower limit and the last calendar day as the upper limit.

There can be up to five years between the first day and the last day of a calendar, i.e. calendars that cover a longer period than five years are not permitted. Furthermore, the calendar limits must lie between January 1, 1970 (1970-01-01) and December 31, 2030 (2030-12-31).

The calendar limits can be modified at any time, taking due account of these restrictions. The following must be noted in this case:

There must be an overlap if the calendar limits are moved, i.e. the new date for the calendar start must be \leq the old date for the calendar end, and the new date for the calendar end must be \geq the old date for the calendar start. Otherwise it cannot be ensured that there is a defined calculation basis for determining the cyclic, symbolic dates (SYMDATs, see description on [page 14ff](#)). If the calendar limits are moved by the user, the information that now lies outside the new calendar limits is lost. This applies even if the modifications are cancelled again, i.e. if the calendar limits are reset to the original values.

The standard working week

Within the calendar limits defined by the user, the individual days are created by “CALENDAR” and are assigned initial values. These values are defined globally in the standard working week and apply to all calendar days that were not explicitly defined otherwise. Any modification to the values in the standard working week is likewise valid for all calendar days, apart from those explicitly defined otherwise.

The standard working week contains the following specifications:

- the weekdays
- the attribute of a weekday
- the working hours of a weekday

Weekdays are the days from Monday through Sunday.

It is not possible to change the names of the weekdays nor the order in which they appear: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

The **attribute** of a weekday specifies whether the respective weekday is a workday or a free day by default.

Default settings:

Monday, Tuesday, Wednesday, Thursday, Friday: workdays
Saturday, Sunday: free days

The **working hours** of a weekday, defined by the two values Start and End, are specified in hours and minutes.

Example

08:30 - 17:30

The following are the permitted combinations for specifying the start time and end time:

- start time = end time
- start time < end time
- start time > end time (e.g. for the night shift)
- “CALENDAR” merely checks whether the values given are acceptable time specifications, i.e. whether the hour specification is in the range from 00 to 23 and the minutes specification is in the range from 00 to 59.

Default setting:

00:00 - 23:59

2.1.2 Calendar days

Within the defined calendar limits, “CALENDAR” creates the calendar days in a contiguous sequence. The number of calendar days created for a calendar file thus depends solely on the specified calendar limits. The user cannot create or delete calendar days.

A calendar day is a weekday that lies within the calendar limits of a calendar. The initial values for the attribute (workday / free day) and the working hours of a calendar day are defined in the standard working week. Both values can be modified explicitly by the user.

The definition of holidays must also be taken into account for the final definition of a calendar day as a workday or a free day.

If the specifications for the attribute value of a calendar day from the standard working week, the definition of holidays, and the explicit specification of an attribute value do not coincide, the final value is determined by the following **attribute rule**:

1. If the attribute (workday / free day) was specified explicitly for a calendar day, this definition always takes priority.
Example:
A calendar day that was defined as a workday remains a workday if it is simultaneously defined as a holiday (for information on holidays, see [page 17ff](#)).
2. If the following applies to a calendar day:
 - no attribute value was explicitly specified or the value was reset (STD)
 - this calendar day was defined as a holidaythen the holiday definition comes into effect. The calendar day is thus a free day.
3. If the following applies to a calendar day:
 - no attribute value was explicitly specified or the value was reset
 - no holiday definition was enteredthen the attribute value for the calendar day is taken from the standard working week.

Note

If the attribute value (workday / free day) is modified for a calendar day, this affects the calculation of the cyclic, symbolic dates (SYMDATs) in a calendar (see description in the following section).

2.1.3 Symbolic dates: SYMDATs

Calendar days of a calendar are assigned to a symbolic date, known as a SYMDAT for short.

A SYMDAT allows you to combine logically related days under one name and use them under this name.

Example

ULTIMO.

With the SYMDAT “ULTIMO”, the set comprising the last calendar day of each month within the defined calendar limits can be combined to form a logical unit and addressed under this name.

A **time specification** can be assigned to each SYMDAT.

This time specification then applies to all calendar days assigned to the SYMDAT.

Default setting:

00:00:00.

Naming conventions for a SYMDAT

Maximum length of a SYMDAT name:
20 characters

Permitted characters in a SYMDAT name:
“0” . . “9”, “A” . . “Z”, “#”, “.”, “\$”

Restrictions:

The “.” (period) must not appear in the first or last position.

Two consecutive periods are not permitted.

Uppercase letters (“A” . . “Z”) and the characters “#” and “\$” are permitted in the first position of a SYMDAT name.

Cyclic and non-cyclic SYMDATs

A distinction is made between symbolic dates that are cyclic and those that are non-cyclic.

Non-cyclic SYMDATs are those whose associated date specifications, i.e. calendar days, cannot be calculated because the intervals between the entries are not known.

The desired calendar days, each one individually, must be explicitly assigned for a non-cyclic SYMDAT.

Cyclic SYMDATs are those for which the intervals between the assigned calendar days can be calculated, because these intervals are repeated cyclically.

The following information, and not the assigned calendar days, is defined for cyclic SYMDATs:

- the start date
- the cycle type: monthly cycle, weekly cycle, workday cycle, daily cycle
- the cycle value: number of months / weeks / workdays / days to be added
- the free-day rule (i.e. the alternative for the days which would be assigned to a SYMDAT in accordance with the cycle, but were defined as free days):
AFTER / BEFORE / SKIP / ON.

With these four specifications, a SYMDAT can be calculated in full for a calendar, i.e. all affected calendar days can be assigned to the cyclic SYMDAT.

The **free-day rule** refers to the user-defined alternative which determines what happens in the defined cycle when a calculated SYMDAT falls on a free day. The user who defines a cyclic SYMDAT has four options in this case:

- BEFORE (enter before the calculated day)
Here, the next workday is sought in the range between the calculated date as the starting point and the date of the previous entry + 1 (i.e. backward). If no workday is found in this period, the entry is omitted.
- AFTER (enter after the calculated day)
Here, the next workday is sought in the range between the calculated date as the starting point and the date of the next entry - 1 (i.e. forward). If no workday is found in this period, the entry is omitted.
- SKIP
The entry for this day is omitted.
- ON (enter)
The entry is made unconditionally on the calculated day.

Four different **cycles** are available:

The monthly cycle

The SYMDAT is calculated from the start date for each month on the same day. The free-day rule is also applied here.

If a calculated calendar day for a SYMDAT falls on a date that does not exist (e.g. February 29 or 30, April 31), the last day of the month is assumed instead.

The weekly cycle

The SYMDAT is calculated from the start date by adding seven calendar days. The free-day rule is also applied here.

If the entry would clash with the preceding or next entry as a result of the free-day rule, this entry is omitted.

Example

Start date: 5.3.1993

Cycle: weekly

Free-day rule: BEFORE

Free days: 6.3.1993 through 15.3.1993

Calculating the SYMDAT results in the following entries:

The first entry is made for 5.3.1993 (start date). The next entry would be on 12.3.1993 (weekly cycle). Because this is a free day, the next workday is sought (free-day rule).

As no workday is found within the period of a week (free days), this entry is omitted. The next entries are made for 19.3.1993 and 26.3.1993.

The workday cycle

To calculate the SYMDAT, the first workday is sought from the specified start date. The cycle value specified by the user defines the number of workdays to be added to an entry in order to make the next entry before the upper limit of the calendar.

As only workdays are considered in the calculation, the free-day rule is not applied for the workday cycle.

The daily cycle

The first entry is the start date. The cycle value is added to this date. The resultant value is the next entry, if not otherwise defined by the free-day rule for any free day calculated.

2.1.4 Holidays

A holiday is a name to which calendar days are assigned. If the holiday is activated, all assigned calendar days become free days, unless explicitly defined as workdays (see the description of the attribute rule on [page 13f](#)).

Naming conventions for a holiday

Maximum length of a holiday name:
30 characters

Permitted characters in a holiday name:
"0" . . "9", "A" . . "Z", "#", ".", "\$"

Restrictions:

The "." (period) must not appear in the first or last position.

Two consecutive periods are not permitted.

Uppercase letters ("A" . . "Z") and the characters "#" and "\$" are permitted in the first position of a holiday name.

Holiday types

A distinction is made between cyclic and non-cyclic holidays.

Cyclic holidays occur annually on the same date, e.g. New Year.

CALENDAR itself automatically assigns cyclic holidays to calendar days.

Non-cyclic holidays that occur on varying days or months, e.g. Easter, must be explicitly assigned the appropriate calendar days of the respective year within the range of the calendar limits.

Activation of holidays

A defined holiday does not come into effect, i.e. a calendar day assigned to the holiday does not become a free day, until this day is activated as a holiday.

The user can define this as desired for each holiday (see mask H010 and the description on [page 63](#)).

When a holiday is created it is activated automatically. The predefined holidays are activated.

The holiday file

To save users from having to define holidays for the validity range of each of their calendar files, a holiday file is supplied with CALENDAR. This file contains predefined holidays which are automatically transferred to the calendar file of the user when a calendar is created.

The user can edit the file using the standard editor EDT.

File attributes

- file name: SYSDAT.CALENDAR.ver.HOLIDAY
- FCBTYPE = SAM
- RECFORM = V

File structure

The records in the holiday file are structured as follows:

```

1   5   10  15  20  25  30  35  40  45  50  55  60  65  70  75  80
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
EASTERMONDAY           ,N,1991-04-01,1992-04-20,1993-04-12,1994-04-04
NEWYEAR                 ,C,****-01-01

```

Explanation

- Each entry is separated from the preceding one by a comma.
- The first entry in each line is the name of the holiday and contains 30 characters (with padding blanks).
- The second entry in each line defines whether the holiday is cyclic or non-cyclic.
- The remaining entries specify the calendar days assigned to the respective holiday. Because EDT only supports a maximum record length of 256 characters, no more than 20 calendar days can be assigned to a (non-cyclic) holiday. In principle, only one date is specified for cyclic holidays (see "NEWYEAR" in the example above). The year specification is replaced by the string "****" in this case.

2.2 Calendar files

As already mentioned at the start, a separate calendar file is created for each calendar. There is no “central” calendar file that contains all existing calendars.

Each calendar file contains precisely one calendar.

Calendar files can be contained on a shared pubset and can thus be made available to a number of computers. Because the name of the calendar file and the name of the calendar are identical, the storage location of the calendar file is automatically defined using the calendar name (CATID and USERID).

2.2.1 Naming conventions for calendar files

Maximum length of a calendar file name:
54 characters

Otherwise, there are no restrictions on the name of a calendar file.

2.2.2 Security mechanisms for calendar files

All security mechanisms for BS2000 files are also permitted for calendar files. However, the user or creator of a calendar file cannot prevent the option of read or write access to the calendar file.

In order to read a calendar file using the command interface, the calendar file must be opened with one of the OPEN modes OPENMOD=*READ or *READALL. This presupposes that read access is possible.

To edit a calendar file using the dialog interface calendar editor, the calendar file must be opened with the open mode OPENMOD=*UPDATE. This presupposes that write access is possible.

Users can only create calendar files under their own user ID.

Two or more users cannot edit a calendar file simultaneously. This is guaranteed by opening the calendar file with SHARUPD=WEAK and OPEN=INOUT with the open mode *UPDATE.

Notes

- A calendar file can only be edited using the CALENDAR subsystem.
- A calendar file can also be read outside this subsystem, e.g. with the SHOW-FILE command.

2.2.3 File attributes

A calendar file is created with the following attributes (see the output of the SHOW-FILE-ATTRIBUTES command in the “Commands” manual [1]):

- FILE-STRUC=PAM
- BUF-LEN=STD(2)
- BLK-CONTR=NO.

2.3 Installation

2.3.1 Components

The product CALENDAR is part of the BS2000 basic configuration and runs under BS2000/OSD-BC as of V2.0.

The product comprises the following components:

- module library: SYSLIB.CALENDAR.<version> .
The library contains the GCs for the CALENDAR subsystem.
- module library: SYSLNK.CALENDAR.<version>
The library contains the modules for the CALENDAR subsystem and the calendar editor.
- module library: SYSLNK.CALENDAR-TU.<version> .
The library contains the modules for the calendar editor.
- syntax file: SYSSDF.CALENDAR.<version>
- subsystem declaration: SYSSSC.CALENDAR.<version>
- message file: SYSMES.CALENDAR.<version>
- REP file for object corrections: SYSREP.CALENDAR.<version>
- structure and installation information for the CALENDAR subsystem for IMON: SYSSII.CALENDAR.<version>
- structure and installation information for the calendar editor: SYSSII.CALENDAR-TU.<version> .
- Example of an holiday file: SYSDAT.CALENDAR.<version> .HOLIDAY
This file contains predefined holidays that are transferred to the calendar when it is created. It can be extended if required.
This file can be taken as a template for the holiday file \$TSOS.SYSDAT.CALENDAR.HOLIDAY.
The supplied sample file provides the system administrator with the option of retaining his old holiday file \$TSOS.SYSDAT.CALENDAR.HOLIDAY and, if necessary, updating it with the contents of the sample file, or regenerating it completely (by copying).

2.3.2 Installing the subsystem

CALENDAR is a subsystem of BS2000 and is managed by DSSM (Dynamic SubSystem Management). For this reason, the subsystem declaration SYSSSC.CALENDAR.<version> must be incorporated in the subsystem catalog with the aid of the program SSCM (Static Subsystem Catalog Manager).

The CALENDAR subsystem and associated REP file are activated by default with BS2000 startup.



The syntax and message files must be activated globally in the system by the system administrator.

The CALENDAR subsystem can be stopped and restarted during operation by means of the /STOP-SUBSYSTEM and /START-SUBSYSTEM commands.

The calendar editor is started by means of the START-CALENDAR-EDITOR command. The CALENDAR subsystem must be activated in order to work with the calendar editor.

3 Dialog interface: the calendar editor

The calendar editor is a utility with which new calendar files can be created and existing calendar files can be read and modified. The dialog interface screens are in the form of FHS masks.

The calendar file is not saved automatically during editing (this must be carried out explicitly). The data in the main memory is written to disk when the calendar file is explicitly closed or saved.

3.1 Overview of functions

The data to be edited with the calendar editor must be assigned to four areas, which correspond to the four basic structure elements of a calendar:

- basic information
- calendar days
- SYMDATs
- holidays

The following overview lists the functions of the calendar editor and assigns these tasks to the respective mask in which the user can interactively implement the corresponding function.

Function	Mask name
Create a new calendar file	Mask C000
Read / edit the basic information in a calendar file	Mask B020
Output / copy / delete / create / activate holidays	Mask H010
Edit holiday information	Mask H020
Select the calendar information function to be read / edited	Mask C010
Create new calendar file	Mask C000
Open calendar file to read / edit an existing calendar	Mask C000
Output calendar days with their properties and select them for further editing	Mask D010
Edit calendar days, i.e. day information	Mask D020
Open an existing calendar file to read / edit a calendar	Mask C000
Save and close an open calendar file	Mask C000
Output / copy / delete / create SYMDATs (symbolic dates)	Mask S010
Edit, i.e. modify SYMDATs (symbolic dates)	Mask S020
Output day information (calendar days) and select them for further editing	Mask D010
Edit day information, i.e. calendar days	Mask D020

3.2 Operating the calendar editor

This section describes how the calendar editor is called and terminated, gives a general description of the mask layout, provides an overview of the general mask sequence in interactive mode, and offers a detailed explanation of each individual mask.

3.2.1 Starting and terminating

The calendar editor is started by means of the START-CALENDAR-EDITOR command.

After the editor is started the main menu appears, i.e. mask C000.

Selecting the file opening mode in this mask determines whether

- information from an existing calendar is to be output
- information from an existing calendar is to be output and edited
- a new calendar file is to be created

The program is terminated normally by selecting function “5” in the function selection mask C000.

The program can be interrupted by pressing function key **K2**; in this case, the program branches to system mode. The user should remember here that opened files are not closed and that the main memory may contain information that has not yet been saved. You are thus strongly advised to return to the calendar editor after the program interrupt.

Format

START-CALENDAR-EDITOR

VERSION = ***STD** / <product-version 6..11> / <product-version 4..9 without-cor> /
 <product-version 3..8 without-man>

,**MONJV** = ***NONE** / <full-filename 1..54 without-gen-vers>

,**CPU-LIMIT** = ***JOB-REST** / <integer 1..32767>

Operands:

VERSION =

Specifies the element version; only permitted for a program library (element type R or L).

VERSION = ***STD**

The default value for the highest element version in program libraries is assumed (see the “LMS” manual).

VERSION = <product-version 6..11>

Complete specification of the product version.

VERSION = <product-version 4..9 without-correction-state>

Specification of the product version without correction state.

VERSION = <product-version 3..8 without-manual-release>

Specification of the product version without release or correction state.

MONJV = *NONE / <full-filename 1..54 without-gen-vers>

Name of the job variable (JV) that is to monitor the program.

During the program run, the system sets the JV to the appropriate values:

\$R program is running
\$T program terminated normally
\$A program terminated abnormally

This operand is only available to users with the software product JV (see also the manual “Job Variables” [2]).

CPU-LIMIT = *JOB-REST / <integer 1..32767 seconds>

Maximum CPU time in seconds that the program can spend running.

If the job was started without a time limit (CPU-LIMIT=*NO), the program likewise runs *without* a time limit (i.e. any entry made is ignored).

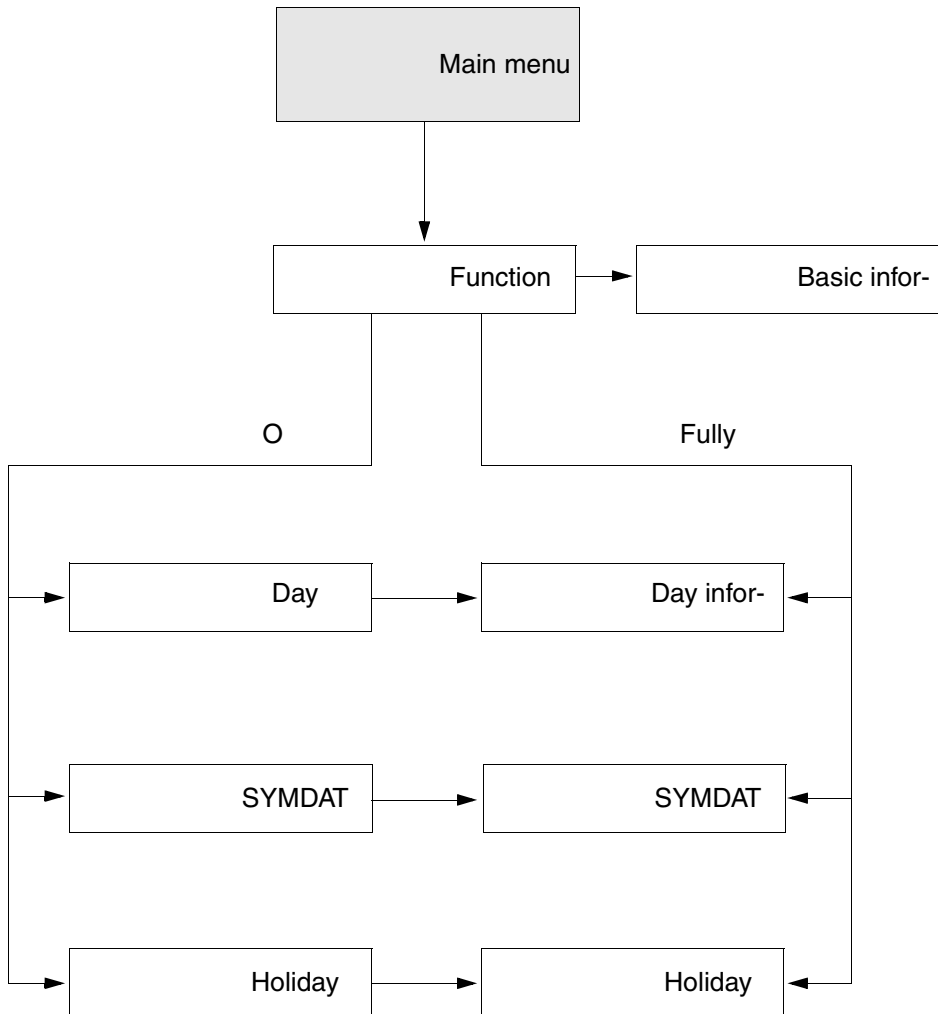
If the program run exceeds the specified time, the program is interrupted interactively and message EXC0075 is output. The user can request a dump, cancel the program, or continue. In batch mode, the program is terminated.

CPU-LIMIT = *JOB-REST

Default:

If the job was started with a time limit, the value defined at system generation is used as the time limit for the program. Otherwise, the program runs without a time limit.

3.2.2 Overview of masks



Display and edit area, lines 4 - 19

The structure and content of this area are specific to each mask.

Operation selection area, lines 20 - 21

The information in lines 21 and 22 provides the user with an overview of the operations that are permitted in the display and edit area of this mask.

Statement input area, line 22

The “Stmt” (statement) input field is followed by a list of all input options. This list is omitted in mask B020. In this case, only the character strings “F12” and “F13” can be entered to produce the same effect as pressing the function keys of the same names.

Message area, lines 23 - 24

The message line provides information on any operating errors and incorrect input or internal errors (e.g. FHS errors) that occurred.

3.2.4 Editor functions

All of the basic functions that can be executed in the calendar editor are summarized in this section and are not dealt with separately in the description of the individual masks.

Exit mask

Masks C000 (main menu) and C010 (function selection) are exited by specifying a function number.

All other masks can be exited using function key **F12** or **F13**.

If you press function key **F12**, the data entered on the screen is not evaluated.

If you press function key **F13**, the data entered is evaluated.

In both cases, you are then returned to the preceding mask.

If these two functions are not available to a user, the character string “F12” or “F13” can be entered in the “Stmt” input field.

Under no circumstances can these exit functions be used to skip one or more levels.

The “page” function

The page function, i.e. page forward and back, is provided in the CALENDAR editor for

- all masks on the object selection level: D010, S010, H010
- all masks on the object editing level: D020, S020, H020.

This page function can be called in two different ways:

- either using a function key
- or by direct input in the “Stmt” input field in line 22 of the relevant mask.

Page back is achieved

- either by pressing function key **F7**
- or by direct input of the “-” character or the character string “F7” in the “Stmt” input field (if the function key is not available to the user).

Page forward is achieved

- either by pressing function key **F8**
- or by direct input of the “+” character or the character string “F8” in the “Stmt” input field (if the function key is not available to the user).

As far as possible, the system pages through whole pages (screen pages).

If you hit an area boundary when paging (start or end of the listed objects), this is indicated in the message area of the mask.

The calendar editor always attempts to fill all the lines of the output area if possible. For example, if the mask output area for listing objects covers 14 lines and if the user hits the end of the list when paging, the last 14 data records are output (if there are 14 or more entries).

The “position” function

All masks on the object selection level and the object editing level contain a “*Goto*” input field. The name of the desired object can be entered in this field. A partially qualified name can be specified in order to select a group of objects. The list of objects is always refreshed following a GOTO specification.

If the GOTO specification (alphabetically) precedes the first element of the list, the list begins with the first existing object. A message in message line 2 notifies the user that the specified object does not exist or precedes the first object output.

If the object specified in GOTO is contained in the list, the list begins with the specified object.

If the GOTO specification lies between two elements of the list, the list begins with the next object (alphabetically). A corresponding message is output in message line 2.

If the GOTO specification (alphabetically) follows the last element of the list, the list begins with the last existing object, followed by the one before that, etc. until the screen is full or no more objects exist. A corresponding message is output in message line 2.

If an entry is syntactically incorrect, a corresponding message is output and a correct entry is expected.

Selecting an object

The objects that exist in a calendar file are listed in the masks on the object selection level, i.e. D010, H010, and S010.

The user selects an object by entering the character “X” in the selection field provided. If the user does not use the character “X” to mark an object but instead uses any other characters apart from a blank, these characters are likewise interpreted as “X”.

However, the following must be noted in this case:

Specifying a character other than “X” in some masks can invoke another function for the selected object. The corresponding letters and associated functions are clearly identifiable to the user in each relevant mask as an alternative selection with a different function. This is explained in the individual descriptions of the masks.

Evaluating input

Multiple actions required simultaneously in a mask (e.g. marking in input fields and calling the page function) are processed in the following sequence:

1. Check function key **F12**
F12 returns you to the preceding mask. If this function was called, no further input (e.g. marking or paging) is evaluated.
2. Check input
If the program discovers an incorrect entry (e.g. an invalid date), the mask is output again with a corresponding error message. The values entered by the user are unchanged in this case.
3. Evaluate existing markings
After the markings are evaluated, the corresponding functions are executed. The markings “X” (object selection) and “C” (object duplication) cause the program to branch to a follow-up mask.
4. Check the “Stmt” input field for the value “New”
In the case of masks where the value “New” is permitted for “Stmt” (S010, H010), the program checks whether this value was specified. If it was (i.e. if the user wants to create a new object), the program branches to the appropriate follow-up mask.

5. Check function key **[F13]**
[F13] returns you to the preceding mask. If this function was called, neither the “page” function nor any value in the “Goto” field (for positioning) is evaluated.
6. Evaluate the “Goto” input field
If “Goto” is used for positioning, the “page” function is ignored.
7. Evaluate the specifications for the “page” function and execute.

If the evaluation of the user’s input results in new values in this mask, the mask is refreshed to display the modified data.

All specifications made at the same time which were to be handled later in accordance with the above sequence (e.g. page function or “Goto” specification for positioning) are no longer evaluated, i.e. they may have to be reentered in the modified mask.

Editing an object

If the calendar file was opened in the main menu mask C000 with “Open Mode = Update”, the data can be edited in one of the masks D020, S020 or H020 (object editing level).


Transferring data

The data is transferred from the screen to the CALENDAR editor using the function key **[DUE]** or **[ENTER]** or the function key **[F13]**.

If no data was entered on the screen, pressing the function key **[DUE]** or **[ENTER]** repeats the previous mask output.

Function keys (overview)

Function keys can be used in the following ways:

Key	Meaning
[DUE] / [ENTER]	Evaluate input on the screen
[F7] 	Page back
[F8]	Page forward
[F12]	Return to preceding mask without evaluating the input on the screen
[F13]	Evaluate the input on the screen and return to preceding mask
[K2]	Interrupt program

3.2.5 Description of the masks

Each calendar editor mask is described in detail in this section under the following areas:

- sequence of masks (graphics)
- function description
- illustration of mask
- description of fields
- follow-up mask

Note

In the field description, all input fields are identified by an input arrow “▶”.

3.2.5.1 Mask C000: Main Main menu of the calendar editor

After the program is called, the first mask to appear is the main menu.

Function

The main menu is the function selection level of the calendar editor.

To read or edit a calendar file, the file must first be opened.

The program thus expects:

- the function number “1” for Open
- the open mode: Read / Update / Create
- the name of the existing calendar file or the calendar file to be created

Before a calendar file is opened (and also before it is closed), only the “Exit” function can be used as an alternative; this terminates the program.

After an existing calendar file is opened for editing (Update), the program branches to mask C010.

After a new calendar file is opened or created (Create), the program branches to mask B020.

The data input must be confirmed with **ENTER**.

Mask

```

C000      Main                               Calendar Editor Vxxx
Calendar Name: :10SN:$QM211.WERK.KALENDER

      Function: 1

      1. Open      Open Mode:  U  (R=Read/U=Update/C=Create)
                   Name: WERK.KALENDER

      2. Edit

      3. Save

      4. Close     Save (Y/N): Y

      5. Exit

Select Function with Number + Enter

Msg:
CLD2007 CALENDAR SUCCESSFULLY OPENED.

LTG                                           TAST

```

Description of fields

► Function:

Input field for the digit assigned to the desired function (1..5).
The function is selected by entering this digit in this field.

If no calendar file is open, only functions 1 or 5 can be specified.

1. OPEN

Entering the function digit “1” in the “Function” field is one of the prerequisites for opening a calendar file. (In addition, the open mode for the calendar file and its name must be specified.)

► Open Mode:

Open mode for the calendar file.

Possible values:

R (=Read), U (=Update), C (=Create)

If no value is specified, the field is preset with the default value “U”.

To create a new calendar file, the default value must be overwritten with the value C (CREATE).

To read an existing calendar file, the default value should be overwritten with the value R (READ) so that the file can be accessed simultaneously by a number of users.

R=Read

Read access to an existing calendar.

The file is transferred to the main memory and can be read simultaneously by a number of users, but cannot be modified. With this access mode, the calendar file need not be closed when exited.

U=Update

Write access to an existing calendar.

The file is transferred to the main memory and can only be edited there by one user at a time. While a user is editing the calendar, it is locked to all other users.

The changes made are not implemented in the calendar file until the function “Save” (3) or “Close” (4) with Save = Y is activated.

C=Create

Create a new calendar file, i.e. generate a new calendar.

After the function digit (1), the open mode (C), and the name of the new calendar are confirmed with **[DUE]** or **[ENTER]**, the program branches to mask B020 where the calendar limits and the standard working week can be defined.

► Name :

Name of the calendar file to be read / updated / created.

The name must be specified, as otherwise a corresponding error message is output in the message area.

The name must be a fully qualified BS2000 file name; the storage location of the file is defined by the catalog ID and the user ID.

2. Edit (for an open calendar file only)

If the function digit “2” is entered in the “Function” field, the program branches to mask C010 where a function can be selected; the program then branches to another menu where objects can be edited.

3. Save (for an open calendar file only)

If the function digit “3” is entered in the “Function” field, the user can intermittently save the modified data before closing the calendar file.

4. Close (for an open calendar file only)

If the function digit “4” is entered in the “Function” field, the calendar file is closed.

► Save (Y/N):

Specifies whether the changes made to the calendar file in the main memory are to be saved when the file is closed.

Default:

“Y” (=YES), i.e. any changes made are implemented in the calendar file. If the user overwrites the default value by explicitly specifying “N” (=NO), all changes since the last “Save” are lost.

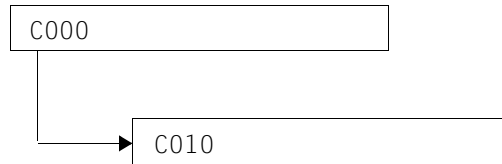
5. Exit

If the function digit “5” is entered in the “Function field”, the calendar editor is exited.

Note

Function “5” cannot be executed if a calendar file opened with “Update” or “Create” is not closed beforehand.

3.2.5.2 Mask C010: Edit Select functions



Function

In this mask you can select a function in order to read or edit the desired objects in the appropriate follow-up mask.

Functions selected via an assigned function digit relate to the following objects:

- basic information
- calendar days
- SYMDATs
- holidays

Basic information can only be selected as a unit, i.e. individual pieces of information cannot be addressed by name. The corresponding function digit (1) always branches to the object editing level in mask B020 so that information can be read or edited there.

Fully qualified names can be specified for the remaining functions relating to the objects “calendar days”, “SYMDATs” and “holidays”. In this case, the program branches to the appropriate mask on the object editing level.

If the object name is not specified or is only given in partially qualified form, the program branches to the corresponding mask on the object selection level, where it is not possible to edit objects.

The function selected (with or without additional specifications for object selection) must be confirmed with **ENTER**.

Mask

```

C010      Edit                               Calendar Editor Vxxx
Calendar Name: :10SN:$QM211.WERK.KALENDER

Function: 1
  1. Edit Basic Information
  2. Edit Days                Start with : 1995-03-15
  3. Edit Symbolic Dates      :
  4. Edit Holidays           :
  5. Return to Main Menu

Select Function with Number + Enter
Msg:
-----
LTG                                           TAST

```

Description of fields

► Function:

Input field for the digit assigned to the desired function (1..5).
The function is selected by entering this digit in this field.

1. Edit Basic Information

Selects the “basic information” function.

Basic information (i.e. calendar limits and standard working week) can only be addressed as a unit. Because a calendar contains only *one* lower limit and only *one* upper limit and also only *one* standard working week, you cannot and need not specify a name for the individual elements.

The program always branches to mask B020, i.e. the object editing level.

The open mode of a calendar file determines whether the basic information of this file can merely be read or can also be edited (for more information, see the description of mask B020 on [page 42ff](#)).

2. Edit Days

Selects the “calendar days” function.

If the user specifies no further information in the “Start with” input field, the program branches to mask D010 where the calendar days of the open calendar file are output, starting with the first entry found.

It is not possible to edit the calendar days here.

► Start with:

Select the start value with which the list of calendar days is to begin.

Input in this field is not mandatory. Fully and partially qualified dates are permitted as input here. There is no default value.

If the user does not specify a value here, the output list begins with the first day of the calendar.

If a *partially qualified* date is entered in the form yyyy-.-.. (yyyy=year, i.e. only the year is specified), the output list begins with the first day of the year or the first calendar day (if the calendar starts at a later date).

If a *partially qualified* date is entered in the form yyyy-mm-.. (mm=month, i.e. both the year and month are specified), the output list begins with the first day of the month in the specified year or with the first calendar day (if the calendar starts at a later date).

If no value or only a partially qualified value is specified, the program branches to mask D010 where the calendar days of the open calendar file are output, starting with the first entry found.

It is not possible to edit the calendar days here.

If a fully qualified date is entered in the form yyyy-mm-dd (dd=day specification), the program branches to mask D020 (object editing level) where the specified calendar day is output for editing with the list of SYMDATs assigned to this day.

Of course, the specified value must actually exist as a calendar day in the calendar file. Otherwise, the user remains in mask C010 and a corresponding error message is output.

3. Edit Symbolic Dates

Selects the “SYMDAT” function (symbolic dates).

If the user specifies no further information in the associated “Start with” input field, the program branches to mask S010 where the SYMDATs of the opened calendar file are output, starting with the first element found.

It is not possible to edit the SYMDATs here.

► Start with : _ _
: :

Select the start value with which the list of SYMDATs is to begin.

Input in this field is not mandatory.

There is no default value.

Partially and fully qualified SYMDAT names are permitted as input.

If a *partially qualified* SYMDAT name “name*” is entered (part of a SYMDAT name, terminated with the “*” character), the following applies:

The program branches to mask D010 where the SYMDATs of the open calendar file are output in alphabetical order, starting with the first entry found (in accordance with the input “name*”).

It is not possible to edit the SYMDATs here.

If a *fully qualified* SYMDAT name is entered, the program branches to mask S020 (object editing level) where the specified SYMDAT is output with the list of calendar days assigned to this SYMDAT. If the SYMDAT is cyclic, the cycle information and the assigned calendar days are locked for editing. If the SYMDAT is non-cyclic, the cycle specifications are likewise locked but the assigned calendar days can be edited.

Of course, the specified value must actually exist as a SYMDAT in the calendar file. Otherwise, the user remains in mask C010 and a corresponding error message is output.

4. Edit Holidays

Selects the “holidays” function.

If the user enters no additional data in the associated “Start with” input field, the program branches to mask H010 where the holidays of the open calendar file are output, starting with the first element.

It is not possible to edit the holidays here.

```

Start with      :   _ _
                :
▶               :
  
```

Select the start value with which the list of holidays is to begin.

Input in this field is not mandatory.

There is no default value.

Partially and fully qualified holiday names are permitted as input.

If a *partially qualified* holiday name is entered (part of a holiday name terminated with the “*” character), the following applies:

The program branches to mask H010 where the holidays of the open calendar file are output in alphabetical order, starting with the first entry found (in accordance with the input “name*”).

It is not possible to edit the holidays here.

If a *fully qualified* holiday name is entered, the program branches to mask H020 (object editing level) where the specified holiday is output with the list of calendar days assigned to this holiday. If the holiday is cyclic, the cycle data can be accessed but the assigned calendar days are locked for editing. If the holiday is non-cyclic, the cycle data is locked but the assigned calendar days can be edited.

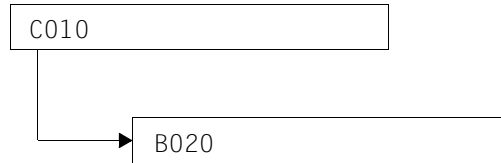
Of course, the specified holiday must actually exist in the calendar file. Otherwise, the user remains in mask C010 and a corresponding error message is output.

5. Return to main menu

Returns you to the main menu C000.

3.2.5.3 Mask B020: Basic Information

Output and edit basic information



Function

The following actions can be performed in this mask:

- The basic information for an existing calendar can be output to the screen.
Prerequisite:
The calendar file must have been opened with “Open Mode = Read”.
- The basic information for an existing calendar can be output to the screen and edited.
Prerequisite:
The calendar file must have been opened with “Open Mode = Update”.
- The basic information for a new calendar can be defined.
Prerequisite:
The calendar file must have been opened with “Open Mode = Create”.

The following basic information is defined as the framework for each calendar:

- the calendar limits with
 - the first calendar day
 - the last calendar day
- the standard working week with
 - the definition of each standard weekday as a workday or a free day
 - the definition of the standard working hours for each standard weekday

The data input must be confirmed with **ENTER**.

Mask

B020	Basic Information	Calendar Editor Vxxx
Calendar Name:	:10SN:\$QM211.WERK.KALENDER	
Calendar Limits	Begin : 1994-01-30 End : 1996-01-30	
Week Definitions	Attribute (W=Work, F=Free)	Working Hours Begin End (HH:MM) (HH:MM)
Day		
Mon	W	08:00 16:30
Tue	W	08:00 16:30
Wed	W	08:00 16:30
Thu	W	08:00 16:30
Fri	W	08:00 16:30
Sat	F	00:00 00:00
Sun	F	00:00 00:00
Store with Enter		F12=Cancel, F13=Return
Stmt:		
Msg :		
LTG		TAST

Description of fields**Calendar Limits**► **Begin : ####-##-##**

Specify the first calendar day with which the calendar is to begin, in the form yyyy-mm-dd, e.g. 1995-06-01.

New calendar limits can be defined here for an existing calendar. In this case, new calendar days are created and/or old ones are deleted.

► **End : ####-##-##**

Specify the last calendar day with which the calendar is to end, in the form yyyy-mm-dd, e.g. 1999-12-31.

A new upper limit can be defined here for an existing calendar. In this case, new calendar days are created and/or old ones are deleted.

Note

Any modifications to the calendar limits affect cyclic SYMDATs (see the description on [page 14ff](#)).

Week Definitions (standard working week)

- ▶ Mon –
- ▶ Tue –
- ▶ Wed –
- ▶ Thu –
- ▶ Fri –
- ▶ Sat –
- ▶ Sun –

Weekdays of the standard working week.

The attribute of a day, i.e. whether it is a workday or a free day, can be modified for each day of the standard working week.

If Monday is to be defined as a workday by default, the character “W” (work) must be entered in the input field to the right of “Mon”.

If Sunday is to be defined as a free day by default, the character “F” (free) must be entered in the input field to the right of “Sun”.

The same applies to the other weekdays, i.e. “Tue”, “Wed”, “Thu”, “Fri”, and “Sat”.

The values defined apply (by default!) to the entire period of the calendar, for each Monday, Tuesday, etc. unless exceptions are defined for specific calendar days.

Working Hours (standard working hours)

- ▶ In the column under “Begin”, the start of the standard working hours is defined for each day of the standard working week in the form hh:mm (hh=hours, mm=minutes).
- ▶ In the column under “End”, the end of the standard working time is defined for each day of the standard working week in the form hh:mm (hh=hours, mm=minutes).

Note

Changing the attribute of a day in the standard working week affects cyclic SYMDATs (see the description on [page 14ff](#)).

- ▶ Stmt:

Possible input:

- the character string “F12”
- the character string “F13”

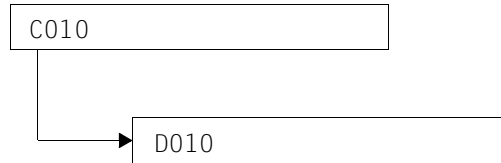
This has the same effect as pressing the function key of the same name (for a description of these function keys, see [page 29](#)).

Follow-up mask

After the mask is exited with **F12** or **F13**, the program returns to the initial mask C010.

3.2.5.4 Mask D010: List of Days

List calendar days and mark them for selection



Function

This mask appears if function “2” is called in mask C010. If no preselection was made for the calendar days to be output, the list begins with the current day.

A line containing the following information is output to the screen for each calendar day:

- the date
- the weekday
- the attribute (workday / free day)
- the holiday identifier (activated / not activated)
- the working hours
- the sum of all SYMDATs assigned to this date

From the list of calendar days, the desired days can be selected by entering an “X” in the selection field at the start of the respective lines.

The data input must be confirmed with .

The program then branches to the follow-up mask D020 where the selected calendar days can be edited.

If several calendar days are selected, these can be edited in succession in the follow-up mask without having to return to mask D010 after each individual one is edited.

Mask

Date	Day	Attr	Holiday	Working Hours	Sym.Date Nr.
1995-03-19	SUN	F	N	00:00 - 00:00	0000
1995-03-20	MON	W	N	08:00 - 16:03	0000
1995-03-21	TUE	W	N	08:00 - 16:30	0000
1995-03-22	WED	W	N	08:00 - 16:30	0000
1995-03-23	THU	W	N	08:30 - 16:30	0001
1995-03-24	FRI	W	N	08:00 - 16:30	0000
1995-03-25	SAT	F	N	00:00 - 00:00	0000
1995-03-26	SUN	F	N	00:00 - 00:00	0000
1995-03-27	MON	W	N	08:00 - 16:03	0000
1995-03-28	TUE	W	N	08:00 - 16:30	0000
1995-03-29	WED	W	N	08:00 - 16:30	0000
1995-03-30	THU	W	N	08:30 - 16:30	0001
1995-03-31	FRI	W	N	08:00 - 16:30	0001
1995-04-01	SAT	F	N	00:00 - 00:00	0000

Select with X + Enter
 Stmt: (+ , -)
 Msg :

F12=Cancel, F13=Return
 Goto: - -

LTG TAST

Description of fields

► -

The calendar day can be marked with "X" in this input field at the start of each line in the list of calendar days.

The respective days are thereby selected for editing, which can be performed in the follow-up mask D020.

Each line in the area listing the calendar days contains the information for *one* calendar day only.

Date

Date of the calendar day.

Day

Weekday of the calendar day.

Attr

Attribute (workday / free day) of the calendar day.

Holiday

Holiday identifier (holiday activated / not activated, see [page 63](#)).

Working Hours

Start and end of the working hours defined for this calendar day, specified in the form hh:mm (hh=hours, mm=minutes).

Sym.Date Nr.

Number of SYMDATs assigned to this calendar day.

- ▶ Stmt: (+ , -)

Possible input:

- the character “-” or the equivalent character string “F7”
- the character “+” or the equivalent character string “F8”
- the character string “F12”
- the character string “F13”

This has the same effect as pressing the function key of the same name (for an overview, see [page 33](#)).

- ▶ Goto: - -

If a fully or partially qualified date is specified, the screen is refreshed. The list output starts with the first element found. If the specified calendar day does not exist, a list is output to the screen beginning with the next element (see the description on [page 30](#)), accompanied by a corresponding message in message line 2.

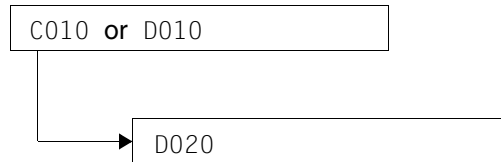
Follow-up mask

After the mask is exited with **F12** or **F13**, the program returns to the initial mask C010.

If at least one calendar day was marked and this selection was confirmed with **ENTER**, the program branches to the follow-up mask D020.

3.2.5.5 Mask D020: Day Information

Edit calendar days, i.e. day information



Function

The calendar days selected for editing in mask D010 are output one after the other in this mask.

The following information appears in the mask for each calendar day selected:

- the date
- the weekday
- the attribute (workday / free day)
- the working hours
- the number of SYMDATs assigned to this date
- the holiday name, if applicable
- the SYMDAT name assigned to the calendar day
- the times of this SYMDAT

In this mask, only the values in the fields “Attr” (attribute) and “Working Hours” which are preset with default values from the basic information can now be modified by the user for the calendar day displayed.

The sequence in which the assigned SYMDATs are listed can be controlled using the field “Change order”. The SYMDATs are listed either in alphabetical order (SYMDAT names) or in numeric order (time specification assigned to the SYMDAT).

The sort sequence is changed in the input field “Change order”:

- by entering the letter “S” for sorting according to SYMDAT names
- by entering the letter “T” for sorting according to time

The data input must be confirmed with **ENTER**.

Mask

Date	Day	Attr (W/F/S)	Working Hours Begin End	SdNr	Holiday Name
1995-03-31	FRI	S	08:00 - 16:30	0001	
Symbolic Date			Time	Symbolic Date	Time
ULTIMO			17:00:00		: : : : : : : : : : : : : : : : : : : :
Store with Enter			F12=Cancel, F13=Return		
Stmt:	(+, -)	S Change Order	Goto:	:	:
Msg :					
LTG			TAST		

Description of fields**Date**

Date of the calendar day.

Day

Weekday of the calendar day.

► **Attr (W/F/S)**

Attribute of the calendar day:

- W: workday
- F: free day
- S: value from the standard working week (see description on [page 12](#))

Working Hours

Begin End

► 00:00 - 23:59

Values specified for the start and end of the working hours to be defined for this calendar day, each specified in the form hh:mm (hh=hours, mm=minutes).

SdNr

Number of SYMDATs assigned to this calendar day.

Holiday Name

Holiday name for this calendar day (if applicable).

Symbolic Date

List of SYMDAT names.

Time

Time of the day assigned to a SYMDAT name in the form hh:mm:ss (hh=hours, mm=minutes, ss=seconds).

► Stmt: (+, -)

Possible input:

- the character “-” or the equivalent character string “F7”
- the character “+” or the equivalent character string “F8”
- the character string “F12”
- the character string “F13”

This has the same effect as pressing the function key of the same name (for an overview, see [page 33](#))

► _ Change order

Input field for switching the sort sequence in which the SYMDATs assigned to the calendar day are listed.

The SYMDATs are sorted either alphabetically (SYMDAT names) or numerically (time specification assigned to the SYMDAT name).

The sort sequence is changed by entering the character “S” or “T” in this field, as appropriate.

► Goto: : :

In this mask, the sort sequence (see “Change order”) determines whether you should specify a SYMDAT name in the left-hand side of the “Goto” positioning field or specify a time (assigned to a SYMDAT) in the right-hand side of this field. One of the fields is always locked, depending on the sort logic. If a value is specified in this field and the sort logic is simultaneously switched, this value is not evaluated (see the description on [page 30](#)).

Follow-up mask

After the mask is exited with or , the program returns to the initial mask C010 or D010.

3.2.5.6 Mask S010: List of Symbolic Dates Output SYMDAT list / mark for selection



Function

This mask is displayed if function “3” is called in mask C010. If no preselection was made for the SYMDAT to be output, the list begins with the first SYMDAT in alphabetical order.

The SYMDATs listed cannot be edited in this mask.

SYMDATs can, however be

- deleted from the list
- selected for copying
- selected for editing
- created

An input field appears in front of each SYMDAT name (Symbolic Date) in this mask. Various values can be entered in this selection field, each relating to a different function:

- The character “X”
A SYMDAT is selected for editing.
The program branches to mask S020 where the SYMDAT is offered for editing.
- The character “C”
A SYMDAT is selected for copying.
The program branches to mask S020.
It is essential that the user enter a new SYMDAT name in the “Symbolic Date” field (which is only available as an input field in this case). The copy process is thus concluded and the copied object can be edited further in mask S020. The SYMDAT type cannot be modified, i.e. a new cyclic SYMDAT cannot be produced after copying a non-cyclic SYMDAT, and vice versa.
- The character “D”
A SYMDAT marked with “D” is deleted from the calendar.

Note

All characters other than “C” or “D” are interpreted as “X”.

To create a new SYMDAT, the value “New” must be entered in the “Stmt” field. The program then branches to mask S020. Here, as in the copy process, the name of the new SYMDAT must be entered in the “Symbolic Date” field. All other values for the new SYMDAT can then be specified.

The data input must be confirmed with **ENTER**.

Mask

S010 List of Symbolic Dates				Calendar Editor Vxxx	
Calendar Name: :10SN:\$QM211.WERK.KALENDER					
Symbolic Date	Type	Symbolic Date	Type	Symbolic Date	Type
FRIDAYFORUM	C	ULTIMO	C		
Select with X or C(opy) or D(elete) + Enter			F12=Cancel, F13=Return		
Stmt: (+, -, New)			Goto:		
Msg :					
LTG			TAST		

Description of fields

An input or selection field, with which the user selects objects and simultaneously defines which function is to be applied to a selected object, appears in front of each SYMDAT name (Symbolic Date) in this mask.

Input values: “C”, “D”, “X” or any character other than “C” or “D”.

For a description of the functions, see [page 53](#).

Symbolic Date

Name of a SYMDAT.

Type

Type of a SYMDAT (cyclic / non-cyclic).

► Stmt: (+, -, New)

Possible input:

– “New”

The program is informed that a new SYMDAT is to be created.

It branches to mask S020 and expects a new SYMDAT name to be entered in the “Symbolic Date” field.

– the character “-” or the equivalent character string “F7”

– the character “+” or the equivalent character string “F8”

– the character string “F12”

– the character string “F13”

This has the same effect as pressing the function key of the same name (for an overview, see [page 33](#))

► Goto:

If a fully or partially qualified SYMDAT name is specified, the screen is refreshed. The list output begins with the first element found. If the specified SYMDAT does not exist, the list output to the screen begins with the next element (see the description on [page 30](#)), accompanied by a corresponding message in message line 2.

Follow-up mask

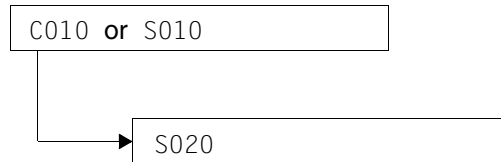
After the mask is exited with **F12** or **F13**, the program returns to the initial mask C010.

If the list of SYMDATs was modified by copying, deleting, or specifying “New”, the list is refreshed in mask S010.

If a SYMDAT was marked with “X” (or any character other than “C” or “D”) and if this selection was confirmed with **ENTER**, the program branches to the follow-up mask D020.

If a SYMDAT was marked with “C” or if the creation of a new SYMDAT was initiated in the “Stmt” field and the respective entries were confirmed with **ENTER**, the program branches to the follow-up mask S020.

3.2.5.7 Mask S020: Symbolic Date Information Output and edit SYMDAT information



Function

This mask outputs all the definitions of the selected SYMDAT and lists all of the days assigned to this SYMDAT.

A *cyclic SYMDAT* is identified by the following definitions:

- Symbolic Date (SYMDAT name)
- Type: C (= cyclic)
- Time
- Cycle Value
- Cycle Type: day / workday / week / month
- Alternative (for a description of the free-day rule, see [page 15](#))
- Cycle Limits

A *non-cyclic SYMDAT* is identified by the following definitions:

- Symbolic Date (SYMDAT name)
- Type: N (=non-cyclic)
- Time
- Assigned Dates

With the exception of the SYMDAT type, all values in the fields listed can be modified by the user for the selected SYMDAT. The Type field is always locked. The Type field can only be assigned a value if a new SYMDAT (New) is created.

The data input must be confirmed with **ENTER**.

Mask

S020		Symbolic Date Information		Calendar Editor Vxxx	
Calendar Name: :10SN:\$QM211.WERK.KALENDER					
Symbolic Date : ULTIMO		Type : C (N/C)		Time : 17:00:00	
Cycle Value: 0001		Cycle Type: 4		Alternative: 1	
Cycle Limits: Begin: 1995-03-31		1. Day		1. Before	
End: 1995-12-31		2. Workday		2. After	
		3. Week		3. Skip	
		4. Month		4. On	
Assigned	1995-03-31	1995-04-28	1995-05-31	1995-06-30	
Dates	1995-07-31	1995-08-31	1995-09-29	1995-10-31	
	1995-11-30	1995-12-29	- -	- -	
	- -	- -	- -	- -	
	- -	- -	- -	- -	
	- -	- -	- -	- -	
	- -	- -	- -	- -	
	- -	- -	- -	- -	
	- -	- -	- -	- -	
Store with Enter				F12=Cancel, F13=Return	
Stmt: (+, -)				Goto: - -	
Msg :					
LTG			TAST		

Description of fields

► Symbolic Date :

Name of the new SYMDAT to be created (if the value "New" was entered in the "Stmt" field of mask S010).

Name of the copied SYMDAT (if an existing SYMDAT was marked with "C" in mask S010).

► Type : _ (N/C)

Type of a SYMDAT.

Values:

N: non-cyclic

C: cyclic

(See description on [page 14ff](#)).

In principle, the SYMDAT type cannot be modified. However, if this is required, the relevant SYMDAT must be deleted and then recreated.

- ▶ Time : 00:00:00

Time assigned to the SYMDAT

(hh:mm:ss; hh = hours, mm = minutes, ss = seconds).

- ▶ Cycle Value: 0000

For cyclic SYMDATs only.

Value for calculating the cycle.

The value is evaluated together with the specification in the “Cycle Type” field.

This field is ignored for non-cyclic SYMDATs.

- ▶ Cycle Type:

For cyclic SYMDATs only.

Type of cycle defined for the selected SYMDAT.

The type is selected by specifying the assigned number:

1 for day	cycle = day
2 for workday	cycle = workday
3 for week	cycle = week
4 for month	cycle = month

A fuller description can be found on [page 14ff.](#)

This field is locked for non-cyclic SYMDATs.

- ▶ Alternative:

For cyclic SYMDATs only.

Definition of an alternative in the event that a calculated entry falls on a free day. The selection is made by specifying the assigned number:

1 for before	i.e. enter before
2 for after	i.e. enter after
3 for skip	i.e. do not enter
4 for on	i.e. enter anyway

A fuller description can be found on [page 15](#).
This field is locked for non-cyclic SYMDATs.

- ▶ Cycle Limits: Begin: - -
- ▶ End: - -

For cyclic SYMDATs only.

Date limits (i.e. time period within which the days must lie) that can be assigned to the SYMDAT.

“Begin” identifies the first day of this time period, i.e. the lower limit.

“End” identifies the last day of this time period, i.e. the upper limit.

The values for “Begin” and “End” are entered in the form yyyy-mm-dd (yyyy=year, 4-digit; mm=month, 2-digit; dd=day, 2-digit).

The “Begin” field must be assigned a value. Specification of a value in the “End” field is left to the discretion of the user.

This field is locked for non-cyclic SYMDATs.

- ▶ Assigned yyyy-mm-dd yyyy-mm-dd yyyy-mm-dd yyyy-mm-dd
 Dates

Dates for each of the calendar days assigned to a *non-cyclic* SYMDAT.

The value for each calendar day is entered in the form yyyy-mm-dd (yyyy=year, 4-digit; mm=month, 2-digit; dd=day, 2-digit).

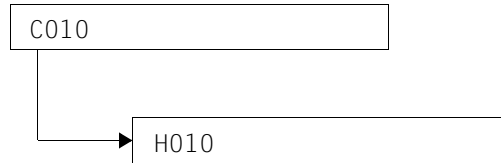
A date is deleted from the list by overwriting the entry with zeros (0).

If the area in this mask for listing the dates is not big enough to hold all of the desired entries, the user can request an additional table by entering the character “+” in the “Stmt” field (for more information, see the description of this field below).

If a *cyclic* SYMDAT was selected to edit the data for this mask, the calculated assigned days are output in the “Assigned Dates” area. However, these dates cannot be modified.

3.2.5.8 Mask H010: List of Holidays

List holidays and mark for selection



Function

This mask is displayed if function “4” is called in mask C010. If no preselection was made for the holidays to be output, the list begins with the first holiday in alphabetical order.

The holidays listed cannot be edited in this mask.

Holidays can, however, be

- deleted from the list
- selected for copying
- selected for editing
- and the creation of a new holiday can be initiated

An input field appears in front of each holiday name in this mask.

The following values can be entered in this field:

- The character “X”
The marked holiday is selected for editing.
The program branches to mask H020 where the holiday information (apart from the name and type) can be modified.
- The character “C”
The marked holiday is selected for copying.
The program branches to mask H020.
It is essential that the user enter a new holiday name in the “Holiday” field (which is only available as an input field in this case). All other information is taken over from the copied holiday and can be modified by the user, with the exception of the type.
- The character “D”
A holiday marked with “D” is deleted from the calendar.

Note

All characters other than “C” or “D” are interpreted as “X”.

To create a new holiday, the value “New” must be entered in the “Stmt” field. The program then branches to mask H020. Here, as in the copying process, the name of the new holiday must be entered in the “Holiday” field. All other values for this holiday can then be specified.

The data input must be confirmed with **ENTER**.

Mask

H010 List of Holidays		Calendar Editor Vxxx	
Calendar Name: :10SN:\$QM211.WERK.KALENDER			
Holiday	Type	Active	
CHRISTMAS	C	Y	
EARLYMAYDAY	C	Y	
EASTER	N	Y	
Select with X or C(opy) or D(elete) + Enter Activate with Y(es) or N(o) + Enter Stmt: (+, -, New) Goto: F12=Cancel, F13=Return Msg :			
LTG		TAST	

Description of fields



An input or selection field, with which the user selects objects and simultaneously defines which function is to be applied to a selected object, appears in front of each holiday name in this mask.

Input values: “C”, “D”, “X”, or any character other than “C” or “D”.

For a description of the functions, see [page 61](#)

Holiday

Name of holiday.

Type

Type of holiday: cyclic / non-cyclic

► Active

This field at the end of the mask line is used to determine whether or not the respective holiday is to be activated.

To activate a holiday means to define whether this holiday is to be handled as a holiday for the relevant calendar. Only an “active” holiday is considered as a holiday when deciding whether a day is a workday or a free day.

Values:

“Y” (yes): the holiday is activated

“N” (no): the holiday is not activated

► Stmt:(+, -, New)

Possible input:

– “New”

The program is informed that a new holiday is to be created.

It branches to mask H020 and expects a new holiday name to be entered in the “Holiday” field.

- the character “-” or the equivalent character string “F7”
- the character “+” or the equivalent character string “F8”
- the character string “F12”
- the character string “F13”

This has the same effect as pressing the function key of the same name (for an overview, see [page 33](#)).

► Goto:

If a fully or partially qualified holiday name is specified, the screen is refreshed. The list output begins with the first element found. If the specified holiday does not exist, the list output to the screen begins with the next element (see the description on [page 30](#)), accompanied by a corresponding message in message line 2.

Follow-up mask

After the mask is exited with **[F12]** or **[F13]**, the program returns to the initial mask C010.

If

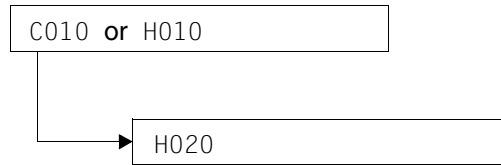
- a holiday is marked with “X” (or any character other than “C” or “D”)
- a holiday is marked with “C”
- the creation of a new holiday is initiated in the “Stmt” field

and the respective function selection is confirmed with **[ENTER]**, the program branches to the follow-up mask H020.

If the list of holidays was modified by copying, deleting, or specifying “New”, the list is refreshed in mask H010.

3.2.5.9 Mask H020: Holiday Information

Output and edit holiday information



Function

This mask displays all the definitions of the selected holiday.
If the holiday is non-cyclic, all of the days assigned to this holiday are listed.

Holidays are divided into

- cyclic holidays
- non-cyclic holidays

Cyclic holidays fall on the same day every year, e.g. Christmas.

Non-cyclic holidays fall on different days, e.g. Easter.

A cyclic holiday is identified by the following definitions:

- Holiday
- Type: C (= cyclic)
- Date for Cyclic Holiday

A non-cyclic holiday is identified by the following definitions:

- Holiday
- Type: N (=non-cyclic)
- Assigned Dates

All values in the fields listed can be modified by the user in this mask for the selected holiday, according to its type.

The data input must be confirmed with **ENTER**.

- ▶ Date for Cyclic Holiday: -

For cyclic holidays only.

The month and day specification for a cyclic holiday in the form mm-dd (mm=month, 2-digit; dd=day, 2-digit).

The definition applies to every year within the defined calendar limits.

Example (definition of New Year's day for the time period of a calendar)

01-01

- ▶ Assigned Dates yyyy-mm-dd yyyy-mm-dd yyyy-mm-dd yyyy-mm-dd

For non-cyclic holidays only.

Dates for each of the calendar days assigned to a *non-cyclic* holiday.

The value for each calendar day is entered in the form yyyy-mm-dd (yyyy=year, 4-digit; mm=month, 2-digit; dd=day 2-digit).

A date is deleted from the list by overwriting the entry with zeros (0).

If the area for listing dates is not big enough to hold all the desired entries, the user can request an additional table by entering the character "+" in the "Stmt" field (for more information, see the description of this field below).

- ▶ Stmt:(+, -)

Possible input:

- the character "-" or the equivalent character string "F7"
- the character "+"

In addition to the standard "page forward" function, this character has the function here of requesting a continuation table for the "Assigned Dates" area, if the area for entering dates is not sufficient for a non-cyclic holiday.

- the character string "F8"
- the character string "F12"
- the character string "F13"

This has the same effect as pressing the function key of the same name (for an overview, see [page 33](#)).

4 Program interface

The CALENDAR subsystem offers a program interface for Assembler and C.

Interface name for Assembler: CALENDR.

Interface name for C: CALEND.h.

This interface enables you to create, modify and output calendar data. The function calls are controlled using the FUNCT operand, and must be specified in the following sequence:

1. Open or create a calendar.
2. Create, modify, delete or output data.
3. Close the calendar.

An overview of possible function calls is given on page 94. This is followed by detailed descriptions of the individual function calls arranged in alphabetical order.

The objects of a calendar file (i.e. the basic information, calendar days, SYMDATs, holidays) which can be accessed via the program interface are described in section 2.1, “The structure of a calendar” (page 11ff).

For performance reasons, the following rules must always be observed:

- The calendar should be created for the required period only.
- When creating a calendar, the standard working week, calendar days and holidays must be defined before creating the SYMDATs.
- If the attribute for several calendar days is modified, a Show function (SH..) should not be called in the meantime.
- If several consecutive holidays are created, deleted, activated or deactivated, a Show function (SH..) should not be called in the meantime.

The following operations are performance-critical:

- The opening of a very large calendar.
The size of the calendar depends on the number of calendar days within the calendar limits, the number of SYMDATs, and the number of holidays.

- The calculation of large numbers of cyclic SYMDATs.
SYMDAT calculations are not performed until data involving cyclic SYMDATs is requested using Show functions.
They are initiated by the following actions:
 - modification of the attribute of a weekday in the standard working week
 - modification of the attribute of a calendar day
 - activation or deactivation of holidays

4.1 Format overview

Macro	Operands
CALENDR	<pre> MF = C / D / L / M / E ,XPAND = PARAM / ENTSTDW / ENTDATE / ENTDAY / ENTSYM / ENTSYMS / ENTHOL / STDWEEK / CCB / DAYLIST / BASINF / LODAY / DAYINF / DAYHOL / LOSYM / SYMINF / LOHOL / HOLINF / NESTM ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,FUNCT = *CREACAL / *OPENCAL / *CLSECAL / *SAVECAL / *MODBAS / *MODDAY / *CRENSYM / *CRECSYM / *DELSYM / *MODNSYM / *MODCSYM / *CRENHOL / *CRECHOL / *DELHOL / *MODNHOL / *MODCHOL / *AKTHOL / *DEAKHOL / *SHBAS / *SHLODAY / *SHDAYIN / *SHDAYHL / *SHLOSYP / *SHSYMIN / *SHLOHOL / *SHHOLIN / *SHNESTM / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,CCB = <var: pointer> / (<reg: pointer>) ,CLDNAME = <c-string 1..54: filename 1..54 with catid> / <var: char:54 filename 1..54 with catid> / <reg: A(char:54 filename 1..54 with catid)> ,OPENMOD = *READ / *READALL / *UPDATE / <var: enum-of _open_s:1> / <reg: enum-of _open_s:1> ,LIMIT1 = *UNCHANGED / *STD / <c-string 10..10: date 10..10> <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,LIMIT2 = *UNCHANGED / *STD / <c-string 10..10: date 10..10> <var: char:10: date 10..10> <reg: A(char:10: date 10..10)> ,WEEKADR = *NO / <var: pointer> / (<reg: pointer>) ,SAVE = *<u>YES</u> / *NO / <var: enum-of _save_s:1> <reg: enum-of _save_s:1> </pre>

Macro	Operands
CALENDR (cont.)	<pre> ,DATE = <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,ATTRIBT = *UNCHANGED / *FREE / *WORK / *STD / <var: enum-of _attribute_s:1> / <reg: enum-of _attribute_s:1> ,WKTIME1 = *UNCHANGED / *STD / <c-string 5..5: time 5..5> / <var: char:5: time 5..5> / <reg: A(char:5: time 5..5)> ,WKTIME2 = *UNCHANGED / *STD / <c-string 5..5: time 5..5> / <var: char:5: time 5..5> <reg: A(char:5: time 5..5)> ,NAME = <c-string 1..30: name 1..30> / <var: char:30: name 1..30> / <reg: A(char:30: name 1..30)> ,ADDDAY = (address, number) address : *NO / <var: pointer> / (<reg: pointer>) number : <integer 0..1860> / <var: int:4: integer 0..1860> <reg: int:4: integer 0..1860> ,DELDAY = (address, number) address : *NO / <var: pointer> / (<reg: pointer>) number : <integer 0..1860> / <var: int:4: integer 0..1860> <reg: int:4: integer 0..1860> ,TIME = *UNCHANGED / *STD / <c-string 8..8: time 8..8> / <var: char:8: time 8..8> / <reg: A(char:8: time 8..8)> ,CYCLTYP = *UNCHANGED / *DAY / *WORKDAY / *WEEK / *MONTH / <var: enum-of _cycle_s:1> <reg: enum-of _cycle_s:1> ,CYCLVAL = *UNCHANGED / <integer 1..9999> / <var: int:4: integer 1..9999> <reg: int:4: integer 1..9999> ,ALTERN = *UNCHANGED / *BEFORE / *AFTER / *SKIP / *ON / <var: enum-of _alternative_s:1> / <reg: enum-of _alternative_s:1> </pre>

Macro	Operands
CALENDR (cont.)	<pre> ,DATE1 = *UNCHANGED / <c-string 10..10: date 10..10> / *FIRST / *NEXT / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,DATE2 = *UNCHANGED / *UNDEFINED / *LAST / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,NAME1 = *FIRST / <c-string 1..30: name 1..30> <var: char:30: name 1..30> / <reg: A(char:30: name 1..30)> ,NAME2 = *LAST / <c-string 1..30: name 1..30> <var: char:30: name 1..30> / <reg: A(char:30: name 1..30)> ,NUMB = *NO / <integer -4096..4096> / <var: int:4: integer -4096..4096> / <reg: int:4: integer -4096..4096> ,OUTPUT = (address, length) address: <var: pointer> / (<reg: pointer>) length: <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000> ,SORT = *TIME / *SYMDAT / <var: enum-of _sort_s:1> / <reg: enum-of _sort_s:1> </pre>

4.2 Description of operands

MF=

The MF operand controls the form of macro expansion (see page 201ff). The operand Version = 1 must always be specified.

XPAND=

Creation of layouts:

- for the parameter list
- for the standard working week
- for areas containing a list of days
- for various output areas

Operand values:

PARAM	Layout for the parameter list
ENTSTDW	Layout for the standard working week
ENTDATE	Layout for a date entry
ENTDAY	Layout for a day entry
ENTSYM	Layout for a SYMDAT entry
ENTSYMS	Layout for a short SYMDAT entry
ENTHOL	Layout for a holiday entry
STDWEEK	Layout for the standard working week (short entry)
CCB	Layout for the control block
DAYLIST	Layout for a list of calendar days that are to be assigned or deleted
BASINF	Layout for the basic information
LODAY	Layout for a list of calendar days
DAYINF	Layout for a calendar day with assigned SYMDATs
DAYHOL	Layout for a calendar day with assigned holidays
LOSYM	Layout for a list of SYMDATs
SYMINF	Layout for a SYMDAT with assigned calendar days
LOHOL	Layout for a list of holidays
HOLINF	Layout for a holiday with assigned calendar days
NESTM	Layout for time (DATE, TIME) and SYMDAT name

VERSION=1

Version of the interface. The layout of the output areas is defined specific to that version. A check is carried out to determine whether the version is supported by the subsystem. At present, only Version 1 is supported.

CALLER=

Evaluated only if MF=E applies.

Caller of the interface: user (TU) or system (TPR). The function call is generated in accordance with this entry.

USER / SYSTEM

For a description of the operand values, see the individual formats on page 94ff.

FUNCT=

Function code with which the desired calendar function is specified.

***CREACAL / *OPENCAL / *CLSECAL / *SAVECAL / *MODBAS / *MODDAY
 *CRENSYM / *CRECSYM / *DELSYM / *MODNSYM / *MODCSYM / *CRENHOL
 *CRECHOL / *DELHOL / *MODNHOL / *MODCHOL / *AKTHOL / *DEAKHOL
 *SHBAS / *SHLODAY / *SHDAYIN / *SHDAYHL / *SHLOSYM / *SHSYMIN
 *SHLOHOL / *SHHOLIN / *SHNESTM / <var: enum-of _funct_s:1> / <reg: enum-of
 _funct_s:1>**

For a description of the operand values, see the individual formats on page 94ff.

CCB=

Control block. The address of the control block must be specified in each function call for a calendar file. The control block must not be modified by the user. When opening a calendar file, this control block is supplied automatically. When closing the calendar file, the values are deleted again.

Following the function calls for creating a new calendar or for opening an existing calendar, the associated calendar file is referred to by means of the control block address as opposed to the file name.

If several calendars are opened simultaneously, the user requires a separate control block for each calendar file.

The layout for the area is created with MF=D and XPAND=CCB.

<var: pointer> / (<reg: pointer>)

For a description of the operand values, see the individual formats on page 94ff.

CLDNAME=

Name of the calendar file, i.e. name of the calendar.

This operand is evaluated only when creating a new calendar or opening an existing one.

**<c-string 1..54: filename 1..54 with catid> /
 <var: char:54 filename 1..54 with catid> /
 <reg: A(char:54 filename 1..54 with catid)>**

For a description of the operand values, see the individual formats on page 94ff.

OPENMOD=

Open mode. This determines how the user accesses the data in the calendar file.

***READ / *READALL / *UPDATE / <var: enum-of _open_s:1> /
<reg: enum-of _open_s:1>**

For a description of the operand values, see the individual formats on page 94ff.

LIMIT1=

Lower limit of the calendar, i.e. the date of the first calendar day in the calendar file.
The lower limit must be less than the upper limit.

***UNCHANGED / *STD / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> /
<reg: A(char:10: date 10..10)>**

For a description of the operand values, see the individual formats on page 94ff.

LIMIT2=

Upper limit of the calendar, i.e. the date of the last calendar day in the calendar file.
The upper limit must be greater than the lower limit.
The period between the lower and upper limits must not be greater than five years.

***UNCHANGED / *STD / <c-string 10..10: date 10..10> /
<var: char:10: date 10..10> / <reg: A(char:10: date 10..10)>**

For a description of the operand values, see the individual formats on page 94ff.

WEEKADR=

Address of the area containing the standard working week.

This can be specified when creating a calendar (FUNCT=*CREACAL) and when modifying the basic information (FUNCT=*MODBAS).

The data area contains an entry for each weekday, for the attribute of that weekday (workday / free day), and for the standard working hours on that weekday.

If this operand is used with the *MODBAS function, i.e. when modifying the basic information, the area in which the user receives output from the *SHBAS function can be used as the input area. This area can be modified beforehand.

***NO / <var: pointer> / (<reg: pointer>)**

For a description of the operand values, see the individual formats on page 94ff.

SAVE=

Specification indicating whether the changes made in the calendar file are to be saved. This operand is not evaluated if the calendar was opened with OPENMOD=*READ or *READALL, as modifications are theoretically impossible in these cases.

***YES / *NO / <var: enum-of _save_s:1> / <reg: enum-of _save_s:1>**

For a description of the operand values, see the individual formats on page 94ff.

DATE=

Date of a calendar day that is to be output or modified.

This must be specified as a full date in the format yyyy-mm-dd (yyyy=year, mm=month, dd=day).

In the case of a cyclic holiday, the character string “****” must be entered for the year specification. In this case, this operand is used to define the day assigned to the holiday.

This operand can be used together with the TIME operand to specify a time as of which the next SYMDAT time is to be output.

**<c-string 10..10: date 10..10> / <var: char:10: date 10..10> /
<reg: A(char:10: date 10..10)>**

For a description of the operand values, see the individual formats on page 94ff.

ATTRIBT=

Attribute of a calendar day.

The calendar day can be defined as either a workday or a free day.

***UNCHANGED / *FREE / *WORK / *STD / <var: enum-of _attribute_s:1> /
<reg: enum-of _attribute_s:1>**

For a description of the operand values, see the individual formats on page 94ff.

WKTIME1=

Start of the working hours for the corresponding calendar day. This is specified in the format hh:mm (hh=hours, mm=minutes).

***UNCHANGED / *STD / <c-string 5..5: time 5..5> / <var: char:5: time 5..5> /
<reg: A(char:5: time 5..5)>**

For a description of the operand values, see the individual formats on page 94ff.

WKTIME2=

End of the working hours for the corresponding calendar day. This is specified in the format hh:mm (hh=hours, mm=minutes).

***UNCHANGED / *STD / <c-string 5..5: time 5..5> / <var: char:5: time 5..5> /
<reg: A(char:5: time 5..5)>**

For a description of the operand values, see the individual formats on page 94ff.

NAME=

Name of a SYMDAT (see description on page 14ff) or of a holiday (see description on page 17f).

This must be specified as a full name. The name of a SYMDAT can be up to 20 characters in length, while that of a holiday can be up to 30 characters in length.

**<c-string 1..30: name 1..30> / <var: char:30: name 1..30> /
<reg: A(char:30: name 1..30)>**

For a description of the operand values, see the individual formats on page 94ff.

ADDDAY = (address, number)

Data area containing the calendar days to be assigned when creating and modifying non-cyclic SYMDATs or non-cyclic holidays.

This area contains the date for each assigned calendar day.

The layout for the area is created with MF=D and XPAND=DAYLIST.

The layout for an individual date entry is created using XPAND=ENTDATE. It is thus possible to access the specifications for the year, month, and day.

address: *NO / <var: pointer> / (<reg: pointer>)

For a description of the operand values, see the individual formats on page 94ff.

**number : <integer 0..1860> / <var: int:4: integer 0..1860> /
<reg: int:4: integer 0..1860>**

For a description of the operand values, see the individual formats on page 94ff.

DELDAY = (address, number)

Data area containing the calendar days whose assignment to non-cyclic SYMDATs or non-cyclic holidays is to be deleted during editing.

This area contains the date for each calendar day whose assignment is to be deleted.

The layout for the area is created with MF=D and XPAND=DAYLIST.

The layout for an individual date entry is created using XPAND=ENTDATE. It is thus possible to access the specifications for the year, month and day.

address: *NO / <var: pointer> / (<reg: pointer>)

For a description of the operand values, see the individual formats on page 94ff.

**number : <integer 0..1860> / <var: int:4: integer 0..1860> /
<reg: int:4: integer 0..1860>**

For a description of the operand values, see the individual formats on page 94ff.

TIME=

Time specification that can be assigned to a SYMDAT. This is specified in the format hh:mm:ss (hh=hours, mm=minutes, ss=seconds).

This operand can be used together with the DATE operand to specify a time as of which the next SYMDAT time is to be output. In this case the operand values *UNCHANGED and *STD are not allowed.

***UNCHANGED / *STD / <c-string 8..8: time 8..8> / <var: char:8: time 8..8> /
<reg: A(char:8: time 8..8)>**

For a description of the operand values, see the individual formats on page 94ff.

CYCLTYP=

Type of cycle (see description on page 15f). Together with the CYCLVAL and ALTERN operands, this operand defines the cycle type for cyclic SYMDATs.

***UNCHANGED / *DAY / *WORKDAY / *WEEK / *MONTH /
<var: enum-of _cycle_s:1> / <reg: enum-of _cycle_s:1>**

For a description of the operand values, see the individual formats on page 94ff.

CYCLVAL=

Cycle.

Depending on the cycle type (CYCLTYP), this operand determines for cyclic SYMDATs the number of days, workdays, weeks, or months between two entries. This entry therefore always refers to the CYCLTYP operand.

***UNCHANGED / <integer 1..9999> / <var: int:4: integer 1..9999> /
<reg: int:4: integer 1..9999>**

For a description of the operand values, see the individual formats on page 94ff.

ALTERN=

Alternative (see the description of the free-day rule on page 15).

With cyclic SYMDATs, this operand defines the alternative in a case where a calculated SYMDAT entry falls on a free day.

***UNCHANGED / *BEFORE / *AFTER / *SKIP / *ON /
<var: enum-of _alternative_s:1> / <reg: enum-of _alternative_s:1>**

For a description of the operand values, see the individual formats on page 94ff.

DATE1=

First date in a range.

With cyclic SYMDATs, this operand defines the start date for calculating the cyclic SYMDAT. In this case, you must specify a full date in the format yyyy-mm-dd (yyyy=year, mm=month, dd=day). If this start date is changed, the calculated SYMDAT entries are also modified. When outputting calendar data, this operand is used to restrict the output. It defines the first entry in a range of calendar days.

The day and month specifications can be omitted. The following entries are valid:

yyyy-mm-dd, yyyy-mm, or yyyy.

The last entry in the output area can be defined using either the DATE2 operand or the NUMB operand.

***UNCHANGED / <c-string 10..10: date 10..10> / *FIRST / *NEXT /
<var: char:10: date 10..10> / <reg: A(char:10: date 10..10)>**

For a description of the operand values, see the individual formats on page 94ff.

DATE2=

Last date in a range.

With cyclic SYMDATs, this operand defines the end date for calculating the cyclic SYMDAT. In this case, you must specify a full date in the format yyyy-mm-dd (yyyy=year, mm=month, dd=day).

If the end date is outside the calendar limits, the SYMDAT entries are calculated only as far as the upper calendar limit.

When outputting calendar data, this operand is used to restrict the output. It defines the last entry in a range of calendar days.

The day and month specifications can be omitted. The following entries are valid:
yyyy-mm-dd, yyyy-mm, or yyyy.

The first entry in the output area is defined using the DATE1 operand.

***UNCHANGED / *UNDEFINED / *LAST / <c-string 10..10: date 10..10> /
<var: char:10: date 10..10> / <reg: A(char:10: date 10..10)>**

For a description of the operand values, see the individual formats on page 94ff.

NAME1=

First entry in a range of SYMDATs or holidays.

The name of a SYMDAT can be up to 20 characters in length, while that of a holiday can be up to 30 characters in length. A name can be partially qualified with an asterisk (*) at the end of the name. In this way, the user can restrict the output of SYMDAT or holiday entries. The last entry in the range can be defined using either the NAME2 operand or the NUMB operand.

***FIRST / <c-string 1..30: name 1..30> / <var: char:30: name 1..30> /
<reg: A(char:30: name 1..30)>**

For a description of the operand values, see the individual formats on page 94ff.

NAME2=

Last entry in a range of SYMDATs or holidays.

The name of a SYMDAT can be up to 20 characters in length, while that of a holiday can be up to 30 characters in length. A name can be partially qualified with an asterisk (*) at the end of the name. In this way, the user can restrict the output of SYMDAT or holiday entries. The first entry in the range can be defined using the NAME1 operand.

***LAST / <c-string 1..30: name 1..30> / <var: char:30: name 1..30> /
<reg: A(char:30: name 1..30)>**

For a description of the operand values, see the individual formats on page 94ff.

NUMB=

Number of desired entries. This specification restricts the output of data. It always refers to one of the operands NAME1 or DATE1.

If you specify a negative value, the entries before that defined by NAME1 or DATE1 are output.

If the value of NUMB ≠ *NO, the NAME2 and DATE2 operands are not evaluated.

***NO / <integer -4096..4096> / <var: int:4: integer -4096..4096> /
<reg: int:4: integer -4096..4096>**

For a description of the operand values, see the individual formats on page 94ff.

OUTPUT=

Output area.

With all output functions, the address and length of an output area for storing the desired data must be specified.

If the output area is too small, a corresponding return code is output informing the user that the output area contains only some of the data or no data at all.

address: <var: pointer> / (<reg: pointer>)

For a description of the operand values, see the individual formats on page 94ff.

length : <integer 0..1000000> / <var: int:4: integer 0..1000000>

For a description of the operand values, see the individual formats on page 94ff.

SORT=

For the *SHDAYIN function only (output day information).

Sort criterion: the assigned SYMDATs should be sorted either by name or by time.

***TIME / *SYMDAT / <var: enum-of _sort_s:1> / <reg: enum-of _sort_s:1>**

For a description of the operand values, see the individual formats on page 94ff.

4.3 Data areas for functions

Several functions require the availability of areas with a specific layout. These areas are created using the XPAND operand.

An area containing the standard working week is created for the *CREACAL and *MODBAS functions.

An area containing a list of days is required for the *CRENSYM, *MODNSYM, *CRENHOL and *MODNHOL functions.

Various output areas are required for the output functions *SHBAS, *SHLODAY, *SHDAYIN, *SHDAYHL, *SHLOSYM, *SHSYMIN, *SHLOHOL and *SHHOLIN.

The output area for the *SHNESTM function contains a DATE, a TIME and the name of a SYMDAT.

The table on the next page lists the individual function calls and the layouts they require (which must be created using the XPAND operand).

Overview

XPAND=	03	06	07	11	15	16	19	20	21	22	23	24	25	26	27
STDWEEK	x			x											
ENTSTDW	x			x			x								
DAYLIST		x	x		x	x									
ENTDATE		x	x		x	x				x				x	
BASINF							x								
LODAY											x				
ENTDAY											x				
DAYINF									x						
ENTSYMS									x						
DAYHOL								x							
ENTHOL								x				x			
LOSYM													x		
ENTSYM													x		
SYMINF														x	
LOHOL												x			
HOLINF										x					
SHNESTM															x

Assignment of numbers to functions (corresponds to the order in which the functions are

described in section 4.6):

03	*CREACAL	Create calendar
06	*CRENHOL	Create non-cyclic holiday
07	*CRENSYM	Create non-cyclic SYMDAT
11	*MODBAS	Modify basic information
15	*MODNHOL	Modify non-cyclic holiday
16	*MODNSYM	Modify non-cyclic SYMDAT
19	*SHBAS	Output basic information
20	*SHDAYHL	Output calendar day with assigned holidays
21	*SHDAYIN	Output calendar day with assigned SYMDATs
22	*SHHOLIN	Output holiday with assigned calendar days
23	*SHLODAY	Output list of calendar days
24	*SHLOHOL	Output list of holidays
25	*SHLOSYM	Output list of SYMDATs
26	*SHSYMIN	Output SYMDAT with assigned calendar days
27	*SHNESTM	Output next time for a set of SYMDATs

4.4 Return codes

Following each macro call, a return code is stored in the standard header. This return code contains information on the success or failure of the macro called.

The standard header has the same structure for all macros, the only difference being the symbolic field names, whose first character (PREFIX=<p>) and three subsequent characters (MACID=<mac>) may vary.

The return code fields have the names **MRET**, **SR1** and **SR2**, and are shown below in bold type.

Standard header

<p><mac>HDR	DS	0A		
<p><mac>FHE	DS	0XL8	0	GENERAL PARAMETER AREA HEADER
*				
<p><mac>IFID	DS	0A	0	INTERFACE IDENTIFIER
<p><mac>FCTU	DS	AL2	0	FUNCTION UNIT NUMBER
*			BIT 15	HEADER FLAG BIT,
*				MUST BE RESET UNTIL FURTHER NOTICE
*			BIT 14-12	UNUSED, MUST BE RESET
*			BIT 11-0	REAL FUNCTION UNIT NUMBER
<p><mac>FCT	DS	AL1	2	FUNCTION NUMBER
<p><mac>FCTV	DS	AL1	3	FUNCTION INTERFACE VERSION NUMBER
*				
<p><mac>RET	DS	0A	4	GENERAL RETURN CODE
<p><mac>SRET	DS	0AL2	4	SUB RETURN CODE
<p><mac> SR2	DS	AL1	4	SUB RETURN CODE 2
<p><mac> SR1	DS	AL1	5	SUB RETURN CODE 1
<p><mac> MRET	DS	0AL2	6	MAIN RETURN CODE
<p><mac>MR2	DS	AL1	6	MAIN RETURN CODE 2
<p><mac>MR1	DS	AL1	7	MAIN RETURN CODE 1
<p><mac>FHL	EQU	8	8	GENERAL OPERAND LIST HEADER LENGTH

Meaning of subcode 1 (SR1)

Subcode 1 divides the return codes into classes.

00 No error

The call was executed in full without errors.

01 Parameter error

The job data is incorrect. This generally indicates a programming error. After correcting the job data, you can repeat the call.

20 Internal error

An error occurred in a system component required by CALENDAR during processing. An error of this class cannot be rectified by the user alone. In most cases, the only practical response is to terminate the application program.

40 Other error

This class includes errors that require a specific response on the part of the caller.

In the case of a return code $\neq 0$, the function call is generally aborted and the action is not performed.

The following return codes are exceptions to this:

- Return code X'0008':
There is no holiday file. The calendar is created, but does not contain any predefined holidays.
- Return code X'001D':
The output area is too small to accommodate all the requested data. However, it does contain at least the volume of existing and selected data. You can then use this information to reset the size of the output area.
- Return code X'0025':
The start date of a cyclic SYMDAT with CYCLTYP=WORKDAY was set to zero when moving the calendar limits, as the SYMDAT does not have any assigned days within the new calendar limits.

(SC2)	SC1	Maincode	Meaning
00	00	0000	No error
01	01	0001	Operand error: incorrect value
02	01	0001	Operand error: address not specified
00	40	0002	Calendar already exists
00	40	0003	Calendar does not exist
00	40	0004	DMS error when accessing the calendar file
00	40	0005	File is not a calendar file
00	40	0006	Calendar inconsistent
00	40	0007	Calendar file cannot be modified at present
00	40	0008	Holiday file not available

(SC2)	SC1	Maincode	Meaning
00	40	0009	DMS error when accessing the holiday
00	40	000A	Too many calendar days within the calendar limits
00	40	000B	End date earlier than start date
00	40	000C	Insufficient storage space
00	40	000D	Control block incorrect
00	40	000E	Date outside calendar limits
00	40	000F	SYMDAT already exists
00	40	0010	SYMDAT does not exist
00	40	0011	SYMDAT is a cyclic SYMDAT
00	40	0012	SYMDAT is a non-cyclic SYMDAT
00	40	0013	Date assignment already exists
00	40	0014	Date assignment does not exist
00	40	0015	Incorrect date specification
00	40	0016	Holiday already exists
00	40	0017	Holiday does not exist
00	40	0018	Holiday is a cyclic holiday
00	40	0019	Holiday is a non-cyclic holiday
00	40	001A	Holiday is already activated
00	40	001B	Holiday not activated
00	40	001C	Output area too small: no data available
00	40	001D	Output area too small: some data available
00	40	001E	Definition of the SYMDAT area incorrect
00	40	001F	Definition of the holiday area incorrect
00	40	0020	Area for standard working week incorrect
00	40	0021	Entry in holiday file incorrect
00	40	0022	LIMIT1 < 1970 or LIMIT2 > 2030 or new LIMIT1 > old LIMIT2
00	40	0023	Warning: start date of cyclic SYMDAT outside calendar limits
00	40	0024	Function not permitted with OPENMOD = *READ or *READALL
00	40	0025	Start date for cyclic SYMDAT with CYCLTYP= WORKDAY was deleted because it was outside calendar limits
00	40	0026	Internal program error
00	40	0027	Calendar version not supported by interface version
00	40	0028	More than 4096 SYMDATs created
00	40	0029	More than 1024 holidays created
00	01	FFFF	Function not supported
00	03	FFFF	Version not supported
00	04	FFFF	Alignment error

4.5 Example

This example contains the following calls of the CALENDR interface:

1. Open a calendar file. *UPDATE open mode is used here, as data is modified under step 4.
2. Output information on a SYMDAT. The output area is then evaluated. The year, month, and day are transferred for each assigned day.
3. Output a list of calendar days. The NUMB operand is used to restrict the output to a single day. The attribute flags of that day are then evaluated. The first flag indicates whether the day is a free day or a workday. The second flag indicates how the day was defined:
 - ,W': The day was defined explicitly as a workday.
 - ,F': The day was defined explicitly as a free day.
 - ,S': The attribute of the day was not explicitly defined. It is taken from the standard working week.
4. Modify the attribute of the calendar day.
5. Close the calendar file. All modifications are saved.

Program

```

TESTCAL  START
TESTCAL  AMODE ANY
TESTCAL  RMODE ANY
          BASR  10,0
          USING *,10
*
          REQM  1           Request memory for parameter list
          LR   4,1
          USING GCLDPMDL,4   DSECT for the parameter list
          REQM  6           Request memory for output area
          LR   5,1
*
          MVC   GCLDPMDL(256),PARAM      Transfer parameter list
          MVC   GCLDPMDL+256(GCLD#-256),PARAM+256
*
          LA   3,KAL1           Address of control block
*
*
* Modify parameter list (open calendar)
*
OPEN     CALENDR MF=M,VERSION=1,FUNCT=*OPENCAL,CCB=(3),      -

```

```

OPENMOD=*UPDATE,CLDNAME='KALENDER'
*
* Call function
*
      CALENDR MF=E,VERSION=1,PARAM=(4)
*
* Evaluate return code
*
STOP1   CLC   GCLDMRET,=AL2(GCLDRSUC)
        BNE   FEHLER
*
*
* Modify parameter list (show SYMDAT information)
*
SHOWSYM CALENDR MF=M,VERSION=1,FUNCT=*SHSYMIN,OUTPUT=((5),18656),      -
        NAME='ULTIMO',DATE1=*FIRST,DATE2=*LAST,CCB=(3)
*
* Call function
*
      CALENDR MF=E,VERSION=1,PARAM=(4)
*
* Evaluate return code
*
        CLC   GCLDMRET,=AL2(GCLDRSUC)
        BNE   FEHLER
*
* Evaluate output area
*
        USING GCLDOSI,5           DSECT for the output area
        LA    6,GCLDSIA           R6 = address of first
*                                 assigned day
        USING GCLDDTEN,6         DSECT for date entry
        LH    7,GCLDSIGD         R7 = number of days available
M1      MVC   JAHR,GCLDDTY       Transfer year
        MVC   MONAT,GCLDDTMO     Transfer month
        MVC   TAG,GCLDDTDY       Transfer day
*
* Evaluate data. If the day assigned to the SYMDAT is the
* same as the date, certain actions can be performed.
*
        LA    6,GCLDDTE#(6)      Move to next day
STOP2   BCT   7,M1              Process next day
*
*
* Modify parameter list (show list of days)
* Only a single day is requested using the NUMB=1 operand
*

```



```

SHOWL0D  CALENDR MF=M,VERSION=1,FUNCT=*SHLODAY,OUTPUT=((5),18656),  -
          DATE1='1995-04-25',NUMB=1,CCB=(3)
*
* Call function
*
          CALENDR MF=E,VERSION=1,PARAM=(4)
*
* Evaluate return code
*
          CLC   GCLDMRET,=AL2(GCLDRSUC)
          BNE   FEHLER
*
* Evaluate output area
*
          USING GCLD0LOD,5           DSECT for the output area
          LA    6,GCLDLDA           R6 = address of first day entry
          USING GCLDDYEN,6           DSECT for day entry
          MVC   ATTR,GCLDDEA         Transfer attribute of day
          MVC   ATTRDEF,GCLDDEAD     Explicitly defined attribute
STOP3    CLI   GCLDDEA,C'F'         Is the day a free day?
          BE    FREEDAY             Perform action for free day
          CLI   GCLDDEAD,C'W'       Day defined as workday
          BNE   CLOSE               Close calendar
*
*
* Modify parameter list (modify calendar day)
*
MODDAY   CALENDR MF=M,VERSION=1,FUNCT=*MODDAY,DATE='1995-04-25',  -
          ATTRIBT=*STD,WKTIME1=*UNCHANGED,WKTIME2=*UNCHANGED,  -
          CCB=(3),PARAM=(4)
*
* Call function
*
          CALENDR MF=E,VERSION=1,PARAM=(4)
*
* Evaluate return code
*
          MVC   RETCODE,GCLDMRET
STOP4    CLC   RETCODE,=AL2(GCLDRSUC)
          BNE   FEHLER
*
*
* Modify parameter list (close calendar)
*
CLOSE    CALENDR MF=M,VERSION=1,FUNCT=*CLSECAL,SAVE=*YES,CCB=(3)
*
* Call function
*

```

```

                CALENDR MF=E,VERSION=1,PARAM=(4)
*
* Evaluate return code
*
STOP5    CLC    GCLDMRET,=AL2(GCLDRSUC)
          BNE   FEHLER
          B     ENDE
*
*
* Perform actions for free day
*
FREEDAY  EQU   *
*
*
ENDE     TERM                                Terminate program
*
*
FEHLER   EQU   *
*
* Error handling (e.g. output return code)
*
                TERM  MODE=ABNORMAL,UNIT=STEP
*
*
* Storage areas
*
JAHR     DS    CL4                          Year
MONAT    DS    CL2                          Month
TAG      DS    CL2                          Day
*
RETCODE  DS    AL2
*
ATTR     DS    CL1
ATTRDEF  DS    CL1
*
* Storage area for the control block
*
                DS    0F
KAL1     DS    0CL12
          DS    CL4
          DS    A
          DS    F
*
* Parameter list
*
PARAM    CALENDR MF=L,VERSION=1
*

```

```

* DSECT for parameter list
*
      CALENDR MF=D,VERSION=1,XPAND=PARAM
*
* DSECT for date entry
*
      CALENDR MF=D,VERSION=1,XPAND=ENTDATE
*
* DSECT for SYMDAT information
*
      CALENDR MF=D,VERSION=1,XPAND=SYMINF
*
* DSECT for list of days
*
      CALENDR MF=D,VERSION=1,XPAND=LODAY
*
* DSECT for day entry
*
      CALENDR MF=D,VERSION=1,XPAND=ENTDAY
*
      END

```

Assembly listing

```

(IN)    DEL-SYS-FILE *OMF
(IN)    START-PROG $ASSEMBH
(OUT)   % BLS0500 PROGRAM ,ASSEMBH', VERSION ,1.1A00' OF ,<date>' LOADED
(OUT)   % BLS0552 COPYRIGHT (C) FUJITSU TECHNOLOGY SOLUTIONS. ALL
        RIGHTS RESERVED
(OUT)   (MSG) % % BLS0519 PROGRAM ,:V4B:$TSOS.ASSEMBH' LOADED
(OUT)   % ASS6010 V 1.1A10 OF BS2000 ASSEMBH READY
(IN)    COMPILE SOURCE=S.TESTCAL,MACROLIB=INTLIB.CALENDAR.112,LISTING=PAR
        (OUTPUT=LST.TESTCAL),TEST-SUPPORT=YES
(OUT)   % ASS6011 ASSEMBLY TIME: 581 MSEC
(OUT)   % ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
(OUT)   % ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
(OUT)   % ASS6006 LISTING GENERATOR TIME: 167 MSEC
(IN)    END
(OUT)   % ASS6012 END OF ASSEMBH

```

Tracer listing

```

(IN)    LOAD-PROG *MOD(*OMF),TEST=AID
(OUT)   % BLS0517 MODULE ,TESTCAL' LOADED
(IN)    %IN STOP1
(IN)    %IN STOP2

```

```

(IN)      %IN STOP3
(IN)      %IN STOP4
(IN)      %IN STOP5
(IN)      %R
(EMP ON)  STOPPED AT LABEL: STOP1 , SRC_REF: 83 , SOURCE: TESTCAL , PROC: TESTCAL
(IN)      %D KAL1 %XL12 -----(01)
(OUT)     *** TID: 00010074 *** TSN: 0C25 *****
(NL)      CURRENT PC: 0000004E   CSECT: TESTCAL *****
(NL)      V'00000188' = KAL1      + #'00000000'
(NL)      00000188 (00000000) C3C3C240 01000000 7EFFFFFF          CCB ....=
(IN)      %R
(EMP ON)  STOPPED AT LABEL: STOP2 , SRC_REF: 128 , SOURCE: TESTCAL , PROC: TESTCAL
(IN)      %D JAHR -----(02)
(OUT)     SRC_REF:   128 SOURCE: TESTCAL  PROC: TESTCAL *****
(NL)      JAHR              = |1995|
(IN)      %D MONAT
(OUT)     MONAT              = |01|
(IN)      %D TAG
(OUT)     TAG                = |31|
(IN)      %D %7
(OUT)     CURRENT PC: 000000AA   CSECT: TESTCAL *****
(NL)      %7                  = 0000000C
(IN)      %D %5-> %XL256 -----(03)
(OUT)     V'00002000' = ABSOLUT + #'00002000'
(NL)      00002000 (00002000) 0001000C 000C000C E4D3E3C9 D4D64040          .....ULTIMO
(NL)      00002010 (00002010) 40404040 40404040 40404040 D5F1F67A          N16:
(NL)      00002020 (00002020) F1F57AF0 F0400000 40404040 40404040          15:00 ..
(NL)      00002030 (00002030) 40404040 40404040 40404040 4000F1F9          .19
(NL)      00002040 (00002040) F9F560F0 F160F3F1 F1F9F9F5 60F0F260          95-01-311995-02-
(NL)      00002050 (00002050) F2F8F1F9 F9F560F0 F360F3F1 F1F9F9F5          281995-03-311995
(NL)      00002060 (00002060) 60F0F460 F3F0F1F9 F9F560F0 F560F3F1          -04-301995-05-31
(NL)      00002070 (00002070) F1F9F9F5 60F0F660 F3F0F1F9 F9F560F0          1995-06-301995-0
(NL)      00002080 (00002080) F760F3F1 F1F9F9F5 60F0F860 F3F1F1F9          7-311995-08-3119
(NL)      00002090 (00002090) F9F560F0 F960F3F0 F1F9F9F5 60F1F060          95-09-301995-10-
(NL)      000020A0 (000020A0) F3F1F1F9 F9F560F1 F160F3F0 F1F9F9F5          311995-11-301995
(NL)      000020B0 (000020B0) 60F1F260 F3F10000 00000000 00000000          -12-31.....
(NL)      000020C0 (000020C0) 00000000 00000000 00000000 00000000          .....
(NL)      REPEATED LINES:      2
(NL)      000020F0 (000020F0) 00000000 00000000 00000000 00000000          .....
(IN)      %R
:
:
:
(EMP ON)  STOPPED AT LABEL: STOP2 , SRC_REF: 128 , TESTCAL , PROC: TESTCAL
(IN)      %R
(EMP ON)  STOPPED AT LABEL: STOP3 , SRC_REF: 163 , SOURCE: TESTCAL , PROC: TESTCAL
(IN)      %D ATTR -----(4)
(OUT)     SRC_REF:   163 SOURCE: TESTCAL  PROC: TESTCAL *****

```

```

(NL)   ATTR           = |W|
(IN)   %D ATTRDEF
(OUT)  ATTRDEF       = |W|
(IN)   %D %5-> %XL40 -----(05)
(OUT)  CURRENT PC: 000000EC   CSECT: TESTCAL *****
(NL)   V*00002000* = ABSOLUT + #'00002000'
(NL)   00002000 (00002000) 0001016D 00010001 F1F9F9F5 60F0F460 ..._....1995-04-
(NL)   00002010 (00002010) F2F5E3E4 C5E6E6E6 D5F0F07A F0F0F2F3 25TUEWWWN00:0023
(NL)   00002020 (00002020) 7AF5F9F0 00000000 :590....
(IN)   %R
(EMP ON) STOPPED AT LABEL: STOP4 , SRC_REF: 191 , SOURCE: TESTCAL , PROC: TESTCAL
(IN)   %D RETCODE -----(06)
(OUT)  SRC_REF: 191 SOURCE: TESTCAL PROC: TESTCAL *****
(NL)   RETCODE           = 0000 ..
(IN)   %R
(EMP ON) STOPPED AT LABEL: STOP5 , SRC_REF: 213 , SOURCE: TESTCAL , PROC: TESTCAL
(IN)   %D KAL1 %XL12 -----(07)
(OUT)  CURRENT PC: 00000146   CSECT: TESTCAL *****
(NL)   V*00000188* = KAL1 + #'00000000'
(NL)   00000188 (00000000) 40404040 00000000 00000000 .....
(IN)   %R

```

Explanation

- (01) The calendar was opened. The control block was supplied with values.
- (02) The SYMDAT information on SYMDAT 'ULTIMO' was output. The variables JAHR (year), MONAT (month), and TAG (day) contain the values of the first assigned day.
- (03) Display of the complete output area for the *SHSYMIN function.
- (04) The list of days was output; output was restricted to one day. The ATTR variable contains the attribute that applies for this day on the basis of the definitions. The ATTRDEF variable contains the attribute that was explicitly defined for this day.
- (05) Display of the complete output area for the *SHLODAY function.
- (06) The *MODDAY function was executed. The attribute for the calendar day was modified. The RETCODE variable contains the return code supplied by the function.
- (07) The calendar is closed again. The control block (KAL1) is deleted and can now be used for another calendar.

4.6 Individual formats

Overview of functions

Function	Brief description
FUNCT = *AKTHOL	Activate holiday
FUNCT = *CLSECAL	Close calendar
FUNCT = *CREACAL	Create calendar
FUNCT = *CRECHOL	Create cyclic holiday
FUNCT = *CRECSYM	Create cyclic SYMDAT
FUNCT = *CRENHOL	Create non-cyclic holiday
FUNCT = *CRENSYM	Create non-cyclic SYMDAT
FUNCT = *DEAKHOL	Deactivate holiday
FUNCT = *DELHOL	Delete holiday
FUNCT = *DELSYM	Delete SYMDAT
FUNCT = *MODBAS	Modify cyclic holiday
FUNCT = *MODCHOL	Modify basic information
FUNCT = *MODCSYM	Modify calendar day
FUNCT = *MODDAY	Modify cyclic SYMDAT
FUNCT = *MODNHOL	Modify non-cyclic holiday
FUNCT = *MODNSYM	Modify non-cyclic SYMDAT
FUNCT = *OPENCAL	Open calendar
FUNCT = *SAVECAL	Save calendar
FUNCT = *SHBAS	Output basic information
FUNCT = *SHDAYHL	Output calendar day with assigned holidays
FUNCT = *SHDAYIN	Output calendar day with assigned SYMDATs
FUNCT = *SHHOLIN	Output holiday with assigned calendar days
FUNCT = *SHLODAY	Output list of calendar days
FUNCT = *SHLOHOL	Output list of holidays
FUNCT = *SHLOSYM	Output list of SYMDATs
FUNCT = *SHSYMIN	Output SYMDAT with assigned calendar days
FUNCT = *SHNESTM	Output next time for a set of SYMDATs

4.6.1 *AKTHOL Activate holiday

The *AKTHOL function is used to activate a holiday.

Only then do the assigned calendar days become holidays (and possibly free days).

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *AKTHOL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..30: name 1..30> / <var: char:30: name 1..30> <reg: A(char:30: name 1..30)></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

***AKTHOL**

Activate holiday.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block whose address is used to refer to the opened calendar file in which a holiday is to be activated. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the holiday to be activated. For a detailed description, see page [77](#).

<c-string 1..30: name 1..30>

Name of the holiday as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the name of the holiday; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the name of the holiday; only permitted in conjunction with MF=M.

4.6.2 *CLSECAL Close calendar

The *CLSECAL function is used to close a calendar file.

The calendar to be closed is specified by means of the control block address. This ensures that only a calendar file that has already been opened (using *CREACAL or *OPENCAL) is closed.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *CLSECAL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,SAVE = *YES / *NO / <var: enum-of _save_s:1> / <reg: enum-of _save_s:1></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

***CLSECAL**

Close calendar file.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file to be closed. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

SAVE=

Specification indicating whether the changes made in the calendar file are saved and transferred to the calendar file. Unsaved changes are lost after the file is closed.

This operand is not evaluated if the calendar file was opened with OPEN=*READ or *READALL, as modification of the file is impossible in these cases.

***YES**

The changes are saved.

***NO**

The changes are not saved.

<var: enum-of _save_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page [196](#). This can be specified only in conjunction with MF=M.

<reg: enum-of _save_s:1>

Register containing the value of the equate in right-justified format.

4.6.3 *CREACAL Create calendar

The *CREACAL function is used to create a new calendar. This function can be executed only if a calendar file with the same name does not already exist.

The calendar file is created and opened with SHARUPD=NO and OPEN=OUTIN. It thus remains locked to other users until it is closed.

When creating a new calendar file, CALENDAR sets up a control block. The address of this area must be specified in the CCB operand for all function calls. The layout of this area can be created with XPAND=CCB.

If a holiday file exists, the holidays it contains are transferred to the calendar file to be created. This also has the effect of activating these predefined holidays (see *AKTHOL).

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *CREACAL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,CLDNAME = <c-string 1..54: filename 1..54 with catid> / <var: char:54 filename 1..54 with catid> / <reg: A(char:54 filename 1..54 with catid)> ,LIMIT1 = *STD / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,LIMIT2 = *STD / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,WEEKADR = *NO / <var: pointer> / (<reg: pointer>)</pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

*CREACAL

Create calendar file.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block for the calendar file to be created. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

CLDNAME=

Name of the calendar file to be created. For a detailed description, see page 75.

<c-string 1..54: filename 1..54 with catid>

Name of the calendar as a direct specification.

<var: char:54 filename 1..54 with catid>

Name of the field containing the calendar name; only permitted in conjunction with MF=M.

<reg: A(char:54 filename 1..54 with catid)>

Register with the address of the field containing the calendar name; only permitted in conjunction with MF=M.

LIMIT1=

Date of the first calendar day in the calendar file to be created.
This value must be less than that specified for LIMIT2.

***STD**

The current date is taken as the date of the first calendar day.

<c-string 10..10: date 10..10>

Date of the first calendar day in the calendar file as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the date of the first calendar day in the calendar file; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date of the calendar day; only permitted in conjunction with MF=M.

LIMIT2=

Date of the last calendar day in the calendar file to be created.
The period between the upper and lower limits of a calendar must not be greater than five years. For a detailed description, see page [76](#).

***STD**

The date of the first calendar day + 1 year is taken as the date of the last calendar day.

<c-string 10..10: date 10..10>

Date of the last calendar day in the calendar file as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the date of the last calendar day in the calendar file; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date of the last calendar day; only permitted in conjunction with MF=M.

WEEKADR=

Address of the area containing the standard working week. For a detailed description, see page [76](#).

***NO**

The default setting for the standard working week is used:
Workdays: Monday, Tuesday, Wednesday, Thursday, Friday
Free days: Saturday, Sunday
Working hours: 00:00 through 23:59.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the standard working week; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the standard working week.

Required layout areas:

Standard working week (XPAND = STDWEEK)

126	1	*	Standard-week		
127	1	GCLDSWA	DS	7CL14	area for standard week entry
128	1	*			is described with
129	1	*			XPAND=ENTSTDW
130	1	GCLDSWA#	EQU	7	
131	1	GCLDPSW#	EQU	*-GCLDSWA	

Standard working week (XPAND = ENTSTDW)

11	1	*	Standard-week-entry		
12	1	GCLDWEDY	DS	CL3	weekday (MON - SUN)
13	1	GCLDWEAT	DS	CL1	attribute of day (W=workday,
14	1	*			F=free day)
15	1	GCLDWET1	DS	CL5	beginning of working hours
16	1	GCLDWET2	DS	CL5	end of working hours
17	1	GCLDSWE#	EQU	*-GCLDWEDY	

4.6.4 *CRECHOL Create cyclic holiday

The *CRECHOL function is used to create a cyclic holiday (see description on page 17f). This function can be executed only if a cyclic or non-cyclic holiday with the same name does not already exist in the respective calendar.

A holiday created by the user is also activated.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *CRECHOL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..30: name 1..30> / <var: char:30: name 1..30> / <reg: A(char:30: name 1..30)> ,DATE = <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

*CRECHOL

Create cyclic holiday.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file in which a cyclic holiday is created. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

<reg: pointer>

Register with the address of the control block.

NAME=

Name of the cyclic holiday to be created. For a detailed description, see page [77](#).

<c-string 1..30: name 1..30>

Name of the holiday as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the name of the holiday; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the name of the holiday; only permitted in conjunction with MF=M.

DATE=

Date of the cyclic holiday to be created. Because a cyclic holiday falls on the same date each year, the string “****” must be entered for the year specification, i.e. “****-mm-dd”. For a detailed description, see page [77](#).

<c-string 10..10: date 10..10>

Date of the holiday as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the date of the holiday; only permitted in conjunction with MF=M.

<reg: A(char:10: name 1..10)>

Register with the address of the field containing the date of the holiday; only permitted in conjunction with MF=M.

4.6.5 *CRECSYM Create cyclic SYMDAT

The *CRECSYM function is used to create a cyclic SYMDAT (see description on page 14). This function can only be executed if a cyclic SYMDAT with the same name does not already exist in the respective calendar.

Format

Macro	Operands
CALENDR	<pre> MF = C / D / L / M / E ,FUNCT = *CRECSYM / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..20: name 1..20> / <var: char:20: name 1..20> / <reg: A(char:20: name 1..20)> ,TIME = *STD / <c-string 8..8: time 8..8> / <var: char:8: time 8..8> / <reg: A(char:8: time 8..8)> ,CYCLTYP = *DAY / *WORKDAY / *WEEK / *MONTH / <var: enum-of _cycle_s:1> / <reg: enum-of _cycle_s:1> ,CYCLVAL = <integer 1..9999> / <var: int:4: integer 1..9999> / <reg: int:4: integer 1..9999> ,ALTERN = *BEFORE / *AFTER / *SKIP / *ON / <var: enum-of _alternative_s:1> / <reg: enum-of _alternative_s:1> ,DATE1 = <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,DATE2 = *UNDEFINED / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> </pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

***CRECSYM**

Create cyclic SYMDAT.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page [195](#). This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file for which the cyclic SYMDAT is created. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the cyclic SYMDAT to be created. For a detailed description, see page [77](#).

<c-string 1..20: name 1..20>

Name of the cyclic SYMDAT as a direct specification.

<var: char:20: name 1..20>

Name of the field containing the SYMDAT name; only permitted in conjunction with MF=M.

<reg: A(char:20: name 1..20)>

Register with the address of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

TIME=

Time specification to be assigned to the cyclic SYMDAT to be created. This is specified in the format hh:mm:ss (hh=hours, mm=minutes, ss=seconds).

***STD**

Default time specification: 00:00:00

<c-string 8..8: time 8..8>

Time specification for the cyclic SYMDAT as a direct specification.

<var: char:8: time 8..8>

Name of the field containing the time specification for the cyclic SYMDAT; only permitted in conjunction with MF=M.

<reg: A(char:8: time 8..8)>

Register with the address of the field containing the time specification for the SYMDAT; only permitted in conjunction with MF=M.

CYCLTYP=

Type of cycle (see description on page 15f). Together with the CYCLVAL and ALTERN operands, this operand defines the cycle.

***DAY**

Cycle = day.

***WORKDAY**

Cycle = working day.

***WEEK**

Cycle = week.

***MONTH**

Cycle = month.

<var: enum-of _cycle_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 196. This can be specified only in conjunction with MF=M.

<reg: enum-of _cycle_s:1>

Register containing the value of the equate in right-justified format.

CYCLVAL=

Cycle value.

Depending on the cycle type (CYCLTYP), this operand determines the number of days, workdays, weeks or months between two entries for the cyclic SYMDAT to be created. This entry therefore always refers to the CYCLTYP operand.

<integer 1..9999>

Number of days / workdays / weeks / months between two entries as a direct specification.

<var: int:4: integer 1..9999>

Name of the field containing the cycle value for the cyclic SYMDAT.

<reg: int:4: integer 1..9999>

Register containing the cycle value; only permitted in conjunction with MF=M.

ALTERN=

Alternative (see the description of the free-day rule on page 15).

For the cyclic SYMDAT to be created, this operand defines the alternative in a case where a calculated SYMDAT entry falls on a free day.

***BEFORE**

The entry is inserted before the free day.

***AFTER**

The entry is inserted after the free day.

***SKIP**

The entry is skipped.

***ON**

The entry is inserted on the free day.

<var: enum-of _alternative_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 196. This can be specified only in conjunction with MF=M.

<reg: enum-of _alternative_s:1>

Register containing the value of the equate in right-justified format.

DATE1=

Start date for calculating the cyclic SYMDAT to be created. For a detailed description, see page 79.

<c-string 10..10: date 10..10>

Start date as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the start date for the cyclic SYMDAT; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the start date; only permitted in conjunction with MF=M.

DATE2=

End date for calculating the cyclic SYMDAT. For a detailed description, see page [79](#).

***UNDEFINED**

Because the end date for calculating the cyclic SYMDAT is not defined, the end date of the calendar is used instead.

<var: char:10: date 10..10>

Name of the field containing the end date for the cyclic SYMDAT; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the end date; only permitted in conjunction with MF=M.

4.6.6 *CRENHOL Create non-cyclic holiday

The *CRENHOL function is used to create a non-cyclic holiday (see description on page 17f). This function can only be executed if a (cyclic or non-cyclic) holiday with the same name does not already exist in the respective calendar.

A holiday created by the user is also activated.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *CRENHOL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..30: name 1..30> / <var: char:30: name 1..30> <reg: A(char:30: name 1..30)> ,ADDDAY = (address, number) address : *NO / <var: pointer> / (<reg: pointer>) number : <integer 0..1860> / <var: int:4: integer 0..1860> / <reg: int:4: integer 0..1860></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

*CRENHOL

Create non-cyclic holiday.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file in which the non-cyclic holiday is created. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the non-cyclic holiday to be created. For a detailed description, see page [77](#).

<c-string 1..30: name 1..30>

Name of the holiday as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the name of the holiday; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the name of the holiday; only permitted in conjunction with MF=M.

ADDDAY = (address, number)

Data area containing the calendar days assigned to the holiday. For a detailed description, see page [78](#).

address: *NO / <var: pointer> / (<reg: pointer>)

Operand value	Meaning
*NO:	No area specification
<var: pointer>:	Alternatively, the following two specifications are possible: - address of the area as a direct address specification in the format A(field) - name of the field containing the address of the area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the area

**number : <integer 0..1860> / <var: int:4: integer 0..1860> /
<reg: int:4: integer 0..1860>**

Operand value	Meaning
<integer 0..1860>:	Number of calendar days in this area as a direct specification
<var: int:4: integer 0..1860>:	Name of the field containing the number of calendar days as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1860>	Register containing the number of calendar days; only permitted in conjunction with MF=M

Required layout areas:

List of calendar days (XPAND = DAYLIST)

```

145 1 *   day-list
146 1 GCLDPDA DS   1860CL10           area for days entry is
147 1 *                                     described with XPAND=ENTDATE
148 1 GCLDPDA# EQU   1860
149 1 GCLDPDY# EQU  *-GCLDPDA

```

Date entry (XPAND = ENTDATE)

```

300 1 *   Date-entry
301 1 GCLDDTY DS   CL4                year
302 1 GCLDDTH1 DS  CL1                hyphen
303 1 GCLDDTMO DS  CL2                month
304 1 GCLDDTH2 DS  CL1                hyphen
305 1 GCLDDTDY DS  CL2                day
306 1 GCLDDTE# EQU *-GCLDDTY

```


4.6.7 *CRENSYM Create non-cyclic SYMDAT

The *CRENSYM function is used to create a non-cyclic SYMDAT (see description on page 14f). This function can only be executed if a (cyclic or non-cyclic) SYMDAT with the same name does not already exist in the respective calendar.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *CRENSYM / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..20: name 1..20> / <var: char:20: name 1..20> / <reg: A(char:20: name 1..20)> ,ADDDAY = (address, number) address : *NO / <var: pointer> / (<reg: pointer>) number : <integer 0..1860> / <var: int:4: integer 0..1860> / <reg: int:4: integer 0..1860> ,TIME = *STD / <c-string 8..8: time 8..8> / <var: char:8: time 8..8> / <reg: A(char:8: time 8..8)></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

***CRENSYM**

Create non-cyclic SYMDAT.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file in which the non-cyclic SYMDAT is created. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the non-cyclic SYMDAT to be created. For a detailed description, see page 77.

<c-string 1..20: name 1..20>

Name of the SYMDAT as a direct specification.

<var: char:20: name 1..20>

Name of the field containing the SYMDAT name; only permitted in conjunction with MF=M.

<reg: A(char:20: name 1..20)>

Register with the address of the field containing the SYMDAT name; only permitted in conjunction with MF=M.

ADDDAY = (address, number)

Data area containing the calendar days assigned to the SYMDAT.

All assigned calendar days must lie within the calendar limits. For a detailed description, see page 78.

address: *NO / <var: pointer> / (<reg: pointer>)

Operand value	Meaning
*NO:	No area specification
<var: pointer>:	Alternatively, the following two specifications are possible: - address of the area as a direct address specification in the format A(field) - name of the field containing the address of the area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the area

**number : <integer 0..1860> / <var: int:4: integer 0..1860> /
<reg: int:4: integer 0..1860>**

Operand value	Meaning
<integer 0..1860>:	Number of calendar days in this area as a direct specification
<var: int:4: integer 0..1860>:	Name of the field containing the number of calendar days as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1860>	Register containing the number of calendar days; only permitted in conjunction with MF=M

TIME=

Time specification to be assigned to the SYMDAT to be created. This is specified in the format hh:mm:ss (hh=hours, mm=minutes, ss=seconds).

***STD**

Default time specification: 00:00:00

<c-string 8..8: time 8..8>

Time specification for the non-cyclic SYMDAT as a direct specification.

<var: char:8: time 8..8>

Name of the field containing the time specification for the non-cyclic SYMDAT; only permitted in conjunction with MF=M.

<reg: A(char:8: time 8..8)>

Register with the address of the field containing the time specification; only permitted in conjunction with MF=M.

Required layout areas:

List of calendar days (XPAND = DAYLIST)

145	1	*	day-list		
146	1	GCLDPDA	DS	1860CL10	area for days entry is
147	1	*			described with XPAND=ENTDATE
148	1	GCLDPDA#	EQU	1860	
149	1	GCLDPDY#	EQU	*-GCLDPDA	

Date entry (XPAND = ENTDATE)

300	1	*	Date-entry		
301	1	GCLDDTY	DS	CL4	year
302	1	GCLDDTH1	DS	CL1	hyphen
303	1	GCLDDTMO	DS	CL2	month
304	1	GCLDDTH2	DS	CL1	hyphen
305	1	GCLDDTDY	DS	CL2	day
306	1	GCLDDTE#	EQU	*-GCLDDTY	

4.6.8 ***DEAKHOL Deactivate holiday**

The *DEAKHOL function is used to deactivate a holiday.

This means that the holiday is no longer considered as such, without actually being deleted immediately from the calendar file. It can be reactivated at a later stage if desired.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *DEAKHOL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..30: name 1..30> / <var: char:30: name 1..30> / <reg: A(char:30: name 1..30)></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

***DEAKHOL**

Deactivate holiday.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page [195](#). This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block whose address is used to refer to the opened calendar file in which a holiday is to be deactivated. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the holiday to be deactivated. For a detailed description, see page 77.

<c-string 1..30: name 1..30>

Name of the holiday as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the name of the holiday; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the name of the holiday; only permitted in conjunction with MF=M.

4.6.9 *DELHOL Delete holiday

The *DELHOL function is used to delete a holiday. Cyclic and non-cyclic holidays are deleted in the same way, i.e. by specifying the control block address of the calendar file and the file name.

Predefined holidays from the holiday file should not be deleted; instead they should be deactivated. Once the calendar has been created, it is not possible to transfer a predefined holiday from the holiday file.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *DELHOL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..30: name 1..30> / <var: char:30: name 1..30> / <reg: A(char:30: name 1..30)></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

***DELHOL**

Delete holiday.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page [195](#). This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block whose address is used to refer to the opened calendar file in which a holiday is to be deleted. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the holiday to be deleted. For a detailed description, see page [77](#).

<c-string 1..30: name 1..30>

Name of the holiday as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the name of the holiday; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the name of the holiday; only permitted in conjunction with MF=M.

4.6.10 *DELSYM Delete SYMDAT

The *DELSYM function is used to delete a SYMDAT entry. Cyclic and non-cyclic SYMDATs are deleted in the same way, i.e. by specifying the control block address of the calendar file and the file name.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *DELSYM / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..20: name 1..20> / <var: char:20: name 1..20> / <reg: A(char:20: name 1..20)></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

***DELSYM**

Delete SYMDAT.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page [195](#). This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block whose address is used to refer to the opened calendar file in which a SYMDAT is to be deleted. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the SYMDAT to be deleted. For a detailed description, see page [77](#).

<c-string 1..20: name 1..20>

Name of the SYMDAT as a direct specification.

<var: char:20: name 1..20>

Name of the field containing the SYMDAT name; only permitted in conjunction with MF=M.

<reg: A(char:20: name 1..20)>

Register with the address of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

4.6.11 ***MODBAS Modify basic information**

The *MODBAS function is used to modify the calendar limits (start and end date) and the standard working week.

Please note the following when modifying the calendar limits:

- The data on calendar days that lie outside the new limits following modification is lost.
- Links to calendar days that lie outside the new limits following modification are deleted for non-cyclic SYMDATs.
- In the case of cyclic SYMDATs with CYCLTYP=*WORKDAY, the start date is corrected if it lies outside the new limits following modification.
- If new days are created by modifying the calendar limits, the attribute and working hours for these days are preset to their default values (see also page 12f). Activated holidays are also taken into consideration for the new days.
- The next time information involving cyclic SYMDATs is accessed following modification of the calendar limits, these cyclic SYMDATs must be recalculated.

In order to modify the standard working week, you must specify the address of the area containing the definitions for the standard working week (WEEKADR).

The layout is created with XPAND = STDWEEK. The contents of this area should be transferred from the output area of the *SHBAS function. You should only modify the required values.

Format

Macro	Operands
CALENDR	MF = C / D / L / M / E , FUNCT = *MODBAS / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> , VERSION = 1 , CALLER = <u>USER</u> / SYSTEM , CCB = <var: pointer> / (<reg: pointer>) , LIMIT1 = *UNCHANGED / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> , LIMIT2 = *UNCHANGED / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> , WEEKADR = *NO / <var: pointer> / (<reg: pointer>)

Operands**MF=**

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

***MODBAS**

Modify basic information.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page [195](#). This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file in which the basic information is to be modified. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

<reg: pointer>

Register with the address of the control block.

LIMIT1=

Lower limit of the calendar, i.e. date of the first calendar day in the calendar file.

If this value is to remain unchanged, you must specify the operand value *UNCHANGED in order to prevent a value entered for another function from being transferred. For a detailed description, see page [76](#).

***UNCHANGED**

The existing value remains unchanged.

<c-string 10..10: date 10..10>

Date of the first calendar day in the calendar file as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the date of the first calendar day; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date; only permitted in conjunction with MF=M.

LIMIT2=

Upper limit of the calendar, i.e. date of the last calendar day in the calendar file.

If this value is to remain unchanged, you must specify the operand value *UNCHANGED in order to prevent a value entered for another function from being transferred. For a detailed description, see page [76](#).

***UNCHANGED**

The existing value remains unchanged.

<c-string 10..10: date 10..10>

Date of the last calendar day in the calendar file as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the date of the last calendar day; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date; only permitted in conjunction with MF=M.

WEEKADR=

Address of the area containing the standard working week.

If the standard working week is to remain unchanged, you must specify the operand value *NO in order to prevent a value entered for another function from being transferred. For a detailed description, see page 76.

***NO**

The standard working week remains unchanged.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the standard working week of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the standard working week; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the standard working week.

Required layout areas:**Standard working week (XPAND = STDWEEK)**

126	1	*	Standard-week		
127	1	GCLDSWA	DS	7CL14	area for standard week entry
128	1	*			is described with
129	1	*			XPAND=ENTSTDW
130	1	GCLDSWA#	EQU	7	
131	1	GCLDPSW#	EQU	*-GCLDSWA	

Standard working week (XPAND = ENTSTDW)

11	1	*	Standard-week-entry		
12	1	GCLDWEDY	DS	CL3	weekday (MON – SUN)
13	1	GCLDWEDY	DS	CL1	attribute of day (W=workday,
14	1	*			F=free day)
15	1	GCLDWET1	DS	CL5	beginning of working hours
16	1	GCLDWET2	DS	CL5	end of working hours
17	1	GCLDSWE#	EQU	*-GCLDWEDY	

4.6.12 *MODCHOL Modify cyclic holiday

The *MODCHOL function is used to modify a cyclic holiday. However, it is not possible to convert a cyclic holiday to a non-cyclic holiday.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *MODCHOL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..30: name 1..30> / <var: char:30: name 1..30> / <reg: A(char:30: name 1..30)> ,DATE = <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

***MODCHOL**

Modify cyclic holiday.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file in which a cyclic holiday is to be modified. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the cyclic holiday to be modified. For a detailed description, see page [77](#).

<c-string 1..30: name 1..30>

Name of the holiday as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the name of the holiday; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the name of the holiday; only permitted in conjunction with MF=M.

DATE=

Date of the cyclic holiday to be modified. Because a cyclic holiday falls on the same date each year, the string “****” must be entered for the year specification, i.e. “****-mm-dd”. For a detailed description, see page [77](#).

<c-string 10..10: date 10..10>

Date of the holiday as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the date of the holiday; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date of the holiday; only permitted in conjunction with MF=M.

4.6.13 *MODDAY Modify calendar day

The *MODDAY function is used to modify the attribute and working hours of a calendar day. Changes to the attribute value (ATTRBT) and to the working hours (WKTIME1 and WKTIME2) affect the specified calendar day (DATE), and are deviations from the definitions of the standard working week. The new values can be reset later on, i.e. the definitions of the standard working week can be reactivated. Modification of the attribute has repercussions for the calculation of cyclic SYMDATs. The cyclic SYMDATs of the calendar file are therefore recalculated the next time data involving cyclic SYMDATs is accessed.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *MODDAY / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,DATE = <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,ATTRIBT = *UNCHANGED / *FREE / *WORK / *STD / <var: enum-of _attribute_s:1> <reg: enum-of _attribute_s:1> ,WKTIME1 = *UNCHANGED / *STD / <c-string 5..5: time 5..5> / <var: char:5: time 5..5> / <reg: A(char:5: time 5..5)> ,WKTIME2 = *UNCHANGED / *STD / <c-string 5..5: time 5..5> / <var: char:5: time 5..5> / <reg: A(char:5: time 5..5)></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

***MODDAY**

Modify calendar day.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file in which a calendar day is to be modified. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

DATE=

Date of the calendar day to be modified. For a detailed description, see page 77.

<c-string 10..10: date 10..10>

Date of the calendar day as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the date of the calendar day; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date of the calendar day; only permitted in conjunction with MF=M.

ATTRIB=

Attribute of the calendar day. The calendar day may be defined as either a workday or a free day.

If this value is to remain unchanged, you must specify the operand value *UNCHANGED.

***UNCHANGED**

The existing value remains unchanged.

***FREE**

The calendar day is a free day.

***WORK**

The calendar day is a workday.

***STD**

The specification for the attribute is taken from the standard working week.

<var: enum-of _attribute_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 196. This can be specified only in conjunction with MF=M.

<reg: enum-of _attribute_s:1>

Register containing the value of the equate in right-justified format.

WKTIME1=

Start of the working hours for the specified calendar day. This is specified in the format hh:mm (hh=hours, mm=minutes).

If this value is to remain unchanged, you must specify the operand value *UNCHANGED.

***UNCHANGED**

The existing value remains unchanged.

***STD**

The specification for the start of the working hours is taken from the standard working week.

<c-string 5..5: time 5..5>

Start of the working hours as a direct specification.

<var: char:5: time 5..5>

Name of the field containing the start of the working hours; only permitted in conjunction with MF=M.

<reg: A(char:5: time 5..5)>

Register with the address of the field containing the start of the working hours; only permitted in conjunction with MF=M.

WKTIME2=

End of the working hours for the corresponding calendar day. This is specified in the format hh:mm (hh=hours, mm=minutes).

If this value is to remain unchanged, you must specify the operand value *UNCHANGED.

***UNCHANGED**

The existing value remains unchanged.

***STD**

The specification for the end of the working hours is taken from the standard working week.

<c-string 5..5: time 5..5>

End of the working hours as a direct specification.

<var: char:5: time 5..5>

Name of the field containing the end of the working hours; only permitted in conjunction with MF=M.

<reg: A(char:5: time 5..5)>

Register with the address of the field containing the end of the working hours; only permitted in conjunction with MF=M.

4.6.14 *MODCSYM Modify cyclic SYMDAT

The *MODCSYM function can be used to modify a cyclic SYMDAT (see description on page 14ff). However, the cyclic SYMDAT cannot be converted to a non-cyclic SYMDAT.

The start date (DATE1) must lie within the calendar limits and must be before the end date (DATE2), if this is defined. The end date can be outside the calendar limits if desired.

Format

Macro	Operands
CALENDR	<pre> MF = C / D / L / M / E ,FUNCT = *MODCSYM / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..20: name 1..20> / <var: char:20: name 1..20> / <reg: A(char:20: name 1..20)> ,TIME = *UNCHANGED / *STD / <c-string 8..8: time 8..8> / <var: char:8: time 8..8> / <reg: A(char:8: time 8..8)> ,CYCLTYP = *UNCHANGED / *DAY / *WORKDAY / *WEEK / *MONTH / <var: enum-of _cycle_s:1> / <reg: enum-of _cycle_s:1> ,CYCLVAL = *UNCHANGED / <integer 1..9999> / <var: int:4: integer 1..9999> / <reg: int:4: integer 1..9999> ,ALTERN = *UNCHANGED / *BEFORE / *AFTER / *SKIP / *ON / <var: enum-of _alternative_s:1> / <reg: enum-of _alternative_s:1> </pre>

Macro	Operands
CALENDR (cont.)	<pre> ,DATE1 = *UNCHANGED / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,DATE2 = *UNCHANGED / *UNDEFINED / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> </pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

*MODCSYM

Modify cyclic SYMDAT.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page [195](#). This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file in which the cyclic SYMDAT is modified. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the cyclic SYMDAT to be modified. For a detailed description, see page [77](#).

<c-string 1..20: name 1..20>

Name of the SYMDAT as a direct specification.

<var: char:20: name 1..20>

Name of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

<reg: A(char:20: name 1..20)>

Register with the address of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

TIME=

Time specification to be assigned to the cyclic SYMDAT to be modified. This is specified in the format hh:mm:ss (hh=hours, mm=minutes, ss=seconds).

If this value is to remain unchanged, you must specify the operand value *UNCHANGED.

***UNCHANGED**

The value remains unchanged.

***STD**

Default time specification: 00:00:00

<c-string 8..8: time 8..8>

Time specification for the cyclic SYMDAT as a direct specification.

<var: char:8: time 8..8>

Name of the field containing the time specification; only permitted in conjunction with MF=M.

<reg: A(char:8: time 8..8)>

Register with the address of the field containing the time specification; only permitted in conjunction with MF=M.

CYCLTYP=

Type of cycle (see description on page [15f](#)). If this value is to remain unchanged, you must specify the operand value *UNCHANGED. For a detailed description, see page [78](#).

***UNCHANGED**

The value remains unchanged.

***DAY**

Cycle = day

***WORKDAY**

Cycle = workday

***WEEK**

Cycle = week

***MONTH**

Cycle = month

<var: enum-of _cycle_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 196. This can be specified only in conjunction with MF=M.

<reg: enum-of _cycle_s:1>

Register containing the value of the equate in right-justified format.

CYCLVAL=

Cycle value.

If this value is to remain unchanged, you must specify the operand value *UNCHANGED. For a detailed description, see page 79.

***UNCHANGED**

The value remains unchanged.

<integer 1..9999>

Number of days / workdays / weeks / months between two entries as a direct specification.

<var: int:4: integer 1..9999>

Name of the field containing the number of days / workdays / weeks / months as an integer; field length = 4 bytes.

<reg: int:4: integer 1..9999>

Register containing the cycle value; only permitted in conjunction with MF=M.

ALTERN=

Alternative (see description of the free day rule on page 15).

If this value is to remain unchanged, you must specify the operand value *UNCHANGED. For a detailed description, see page 79.

***UNCHANGED**

The value remains unchanged.

***BEFORE**

The entry is inserted before the free day.

***AFTER**

The entry is inserted after the free day.

***SKIP**

The entry is skipped.

***ON**

The entry is inserted on the free day.

<var: enum-of _alternative_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 196. This can be specified only in conjunction with MF=M.

<reg: enum-of _alternative_s:1>

Register containing the value of the equate in right-justified format.

DATE1=

Start date for calculating the cyclic SYMDAT to be modified.

If this value is to remain unchanged, you must specify the operand value *UNCHANGED.

For a detailed description, see page 79.

***UNCHANGED**

The value remains unchanged.

<c-string 10..10: date 10..10>

Start date as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the start date; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the start date; only permitted in conjunction with MF=M.

DATE2=

End date for calculating the cyclic SYMDAT.

If this value is to remain unchanged, you must specify the operand value *UNCHANGED.

For a detailed description, see page 79.

***UNCHANGED**

The value remains unchanged.

***UNDEFINED**

Because the end date for calculating the cyclic SYMDAT is not defined, the end date of the calendar is used instead.

<c-string 10..10: date 10..10>

End date as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the end date; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the end date; only permitted in conjunction with MF=M.

4.6.15 *MODNHOL Modify non-cyclic holiday

The *MODNHOL function is used to modify a non-cyclic holiday, i.e. additional calendar days are linked to the holiday (ADDDAY) or existing links are deleted (DELDAY). If one of these two operands is to remain unchanged, the operand value *NO must be specified for that operand.

Format

Macro	Operands
CALENDR	<pre> MF = C / D / L / M / E ,FUNCT = *MODNHOL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..30: name 1..30> / <var: char:30: name 1..30> / <reg: A(char:30: name 1..30)> ,ADDDAY = (address, number) address : *NO / <var: pointer> / (<reg: pointer>) number: <integer 0..1860> / <var: int:4: integer 0..1860> / <reg: int:4: integer 0..1860> ,DELDAY = (address, number) address : *NO / <var: pointer> / (<reg: pointer>) number: <integer 0..1860> / <var: int:4: integer 0..1860> / <reg: int:4: integer 0..1860> </pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

***MODNHOL**

Modify non-cyclic holiday.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file in which a non-cyclic holiday is to be modified. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the non-cyclic holiday to be modified. For a detailed description, see page 77.

<c-string 1..30: name 1..30>

Name of the holiday as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the name of the holiday; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the name of the holiday; only permitted in conjunction with MF=M.

ADDDAY = (address, number)

Data area containing the calendar days to be assigned to the holiday.

If the holiday is already linked to one of these days, another link is not established; instead the function call is aborted. For a detailed description, see page 78.

address: *NO / <var: pointer> / (<reg: pointer>)

Operand value	Meaning
*NO:	No area specification
<var: pointer>:	Alternatively, the following two specifications are possible: <ul style="list-style-type: none"> - address of the area as a direct address specification in the format A(field) - name of the field containing the address of the area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the area

number : <integer 0..1860> / <var: int:4: integer 0..1860> / <reg: int:4: integer 0..1860>

Operand value	Meaning
<integer 0..1860>:	Number of calendar days in this area as a direct specification
<var: int:4: integer 0..1860>:	Name of the field containing the number of calendar days as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1860>:	Register containing the number of calendar days; only permitted in conjunction with MF=M.

DELDAY = (address, number)

Data area containing the calendar days whose assignment to the non-cyclic holiday specified under NAME is to be deleted. If this area contains a calendar day that is not linked to the holiday, the function call is aborted and no links are deleted. For a detailed description, see page [78](#).

address: *NO / <var: pointer> / (<reg: pointer>)

Operand value	Meaning
*NO:	No area specification
<var: pointer>:	Alternatively, the following two specifications are possible: <ul style="list-style-type: none"> - address of the area as a direct address specification in the format A(field) - name of the field containing the address of the area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the area

number : <integer 0..1860> / <var: int:4: integer 0..1860> /
 <reg: int:4: integer 0..1860>

Operand value	Meaning
<integer 0..1860>:	Number of calendar days in this area as a direct specification
<var: int:4: integer 0..1860>:	Name of the field containing the number of calendar days as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1860>:	Register containing the number of calendar days; only permitted in conjunction with MF=M.

Required layout areas:

List of calendar days (XPAND = DAYLIST)

```

145 1 *   day-list
146 1 GCLDPDA DS   1860CL10
147 1 *
148 1 GCLDPDA# EQU 1860
149 1 GCLDPDY# EQU *-GCLDPDA

```

area for days entry is
described with XPAND=ENTDATE

Date entry (XPAND = ENTDATE)

```

300 1 *   Date-entry
301 1 GCLDDTY DS   CL4
302 1 GCLDDTH1 DS CL1
303 1 GCLDDTMO DS CL2
304 1 GCLDDTH2 DS CL1
305 1 GCLDDTDY DS CL2
306 1 GCLDDTE# EQU *-GCLDDTY

```

year
hyphen
month
hyphen
day

4.6.16 *MODNSYM Modify non-cyclic SYMDAT

The *MODNSYM function is used to modify a non-cyclic SYMDAT, i.e. additional calendar days are linked to the SYMDAT entry (ADDDAY) or existing links are deleted (DELDAY). If one of these two operands is to remain unchanged, the operand value *NO must be specified for that operand. If the assigned time is to remain unchanged, the value *UNCHANGED must be specified for TIME.

Format

Macro	Operands
CALENDR	<pre> MF = C / D / L / M / E ,FUNCT = *MODNSYM / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..20: name 1..20> / <var: char:20: name 1..20> / <reg: A(char:20: name 1..20)> ,TIME = *UNCHANGED / *STD / <c-string 8..8: time 8..8> / <var: char:8: time 8..8> / <reg: A(char:8: time 8..8)> ,ADDDAY = (address, number) address : *NO / <var: pointer> / (<reg: pointer>) number: <integer 0..1860> / <var: int:4: integer 0..1860> / <reg: int:4: integer 0..1860> ,DELDAY = (address, number) address : *NO / <var: pointer> / (<reg: pointer>) number: <integer 0..1860> / <var: int:4: integer 0..1860> / <reg: int:4: integer 0..1860> </pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

***MODNSYM**

Modify non-cyclic SYMDAT.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page [195](#). This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file in which the non-cyclic SYMDAT is to be modified. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the non-cyclic SYMDAT to be modified. For a detailed description, see page [77](#).

<c-string 1..20: name 1..20>

Name of the SYMDAT as a direct specification.

<var: char:20: name 1..20>

Name of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

<reg: A(char:20: name 1..20)>

Register with the address of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

TIME=

Time specification assigned to the SYMDAT to be modified. This is specified in the format hh:mm:ss (hh=hours, mm=minutes, ss=seconds).

***UNCHANGED**

The value remains unchanged.

***STD**

Default time specification: 00:00:00

<c-string 8..8: time 8..8>

Time specification for the cyclic SYMDAT as a direct specification.

<var: char:8: time 8..8>

Name of the field containing the time specification; only permitted in conjunction with MF=M.

<reg: A(char:8: time 8..8)>

Register with the address of the field containing the time specification; only permitted in conjunction with MF=M.

ADDDAY = (address, number)

Data area containing the calendar days to be assigned to the SYMDAT. All days must lie within the calendar limits. If the SYMDAT is already linked to one of these days, another link is not established; instead the function call is aborted. For a detailed description, see page 78.

address: *NO / <var: pointer> / (<reg: pointer>)

Operand value	Meaning
*NO:	No area specification
<var: pointer>:	Alternatively, the following two specifications are possible: <ul style="list-style-type: none"> - address of the area as a direct address specification in the format A(field) - name of the field containing the address of the data area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the data area

number : <integer 0..1860> / <var: int:4: integer 0..1860> /
 <reg: int:4: integer 0..1860>

Operand value	Meaning
<integer 0..1860>:	Number of calendar days in this area as a direct specification
<var: int:4: integer 0..1860>:	Name of the field containing the number of calendar days as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1860>:	Register containing the number of calendar days; only permitted in conjunction with MF=M.

DELDAY = (address, number)

Data area containing the calendar days whose assignment to the non-cyclic SYMDAT specified under NAME is to be deleted. If this area contains a calendar day that is not linked to the SYMDAT, the function call is aborted and no links are deleted. For a detailed description, see page [78](#).

address: *NO / <var: pointer> / (<reg: pointer>)

Operand value	Meaning
*NO:	No area specification
<var: pointer>:	Alternatively, the following two specifications are possible: - address of the area as a direct address specification in the format A(field) - name of the field containing the address of the data area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the data area

number : <integer 0..1860> / <var: int:4: integer 0..1860> /
 <reg: int:4: integer 0..1860>

Operand value	Meaning
<integer 0..1860>:	Number of calendar days in this area as a direct specification
<var: int:4: integer 0..1860>:	Name of the field containing the number of calendar days as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1860>:	Register containing the number of calendar days; only permitted in conjunction with MF=M.

Required layout areas:

List of calendar days (XPAND = DAYLIST)

145	1	*	day-list		
146	1	GCLDPDA	DS	1860CL10	area for days entry is
147	1	*			described with XPAND=ENTDATE
148	1	GCLDPDA#	EQU	1860	
149	1	GCLDPDY#	EQU	*-GCLDPDA	

Date entry (XPAND = ENTDATE)

300	1	*	Date-entry		
301	1	GCLDDTY	DS	CL4	year
302	1	GCLDDTH1	DS	CL1	hyphen
303	1	GCLDDTMO	DS	CL2	month
304	1	GCLDDTH2	DS	CL1	hyphen
305	1	GCLDDTDY	DS	CL2	day
306	1	GCLDDTE#	EQU	*-GCLDDTY	

4.6.17 *OPENCAL Open calendar

The *OPENCAL function is used to open the specified calendar.

A calendar can only be opened if a calendar file with this name exists. Otherwise, the function call is aborted.

When opening a calendar, CALENDAR creates a control block which is then used to address the opened calendar. The address of the control block must be specified in the CCB operand. The layout for the area is created with XPAND=CCB.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *OPENCAL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,CLDNAME = <c-string 1..54: filename 1..54 with catid> / <var: char:54 filename 1..54 with catid> / <reg: A(char:54 filename 1..54 with catid)> ,OPENMOD = *READ / *READALL / *UPDATE / <var: enum-of _open_s:1> / <reg: enum-of _open_s:1></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

*OPENCAL

Open calendar file.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file to be opened. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

CLDNAME=

Name of the calendar file, i.e. the calendar, to be opened. For a detailed description, see page [75](#).

<c-string 1..54: filename 1..54 with catid>

Name of the calendar as a direct specification.

<var: char:54 filename 1..54 with catid>

Name of the field containing the name of the calendar; only permitted in conjunction with MF=M.

<reg: A(char:54 filename 1..54 with catid)>

Register with the address of the field containing the name of the calendar; only permitted in conjunction with MF=M.

OPENMOD=

Open mode. For a detailed description, see page [76](#).

***READ**

The calendar file is opened for reading using SHARUPD=WEAK and OPEN=INPUT. The calendar data can be output, but not modified. With this open mode, any number of users can simultaneously access the calendar file. Each subsequent access to the calendar (e.g. *SHBAS, *SHLOSYM, etc.) involves file access.

***READALL**

The calendar file is opened for reading using SHARUPD=WEAK and OPEN=INPUT. The data can be output, but not modified. With this open mode, any number of users can simultaneously access the calendar file.

Unlike *READ open mode, this mode causes the calendar to be loaded in its entirety into the memory and kept there until it is closed. File access is therefore not required for subsequent calls. However, the memory requirement here is greater than with *READ open mode.

***UPDATE**

The calendar file is opened for editing with SHARUPD=WEAK and OPEN=INOUT. Only one user at a time can open the calendar with OPENMOD=*UPDATE. The data can be output and modified.

The calendar is loaded in its entirety into the memory. All changes are then made in the memory. The modified data is not saved to the calendar file until the file is closed.

<var: enum-of _open_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _open_s:1>

Register containing the value of the equate in right-justified format.

Layout of the control block (XPAND = CCB)

```

136 1 *   control-block
137 1 GCLDCBTX DS    CL4           text
138 1 GCLDCBAD DS    A             address
139 1 GCLDCBCP DS    F             check pattern
140 1 GCLDPCB# EQU   *-GCLDCBTX

```

4.6.18 *SAVECAL Save calendar

The *SAVECAL function is used to save the opened calendar specified by means of the control block address. All calendar data is saved to the calendar file on disk. The calendar file in memory remains open and the calendar data is retained in memory, so that the file can be edited again immediately after saving.

Format

Macro	Operands
CALENDR	MF = C / D / L / M / E ,FUNCT = *SAVECAL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>)

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

*SAVECAL

Save calendar.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file to be saved. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

4.6.19 *SHBAS Output basic information

The *SHBAS function is used to request output of the basic information of the specified calendar file. The layout of the output area is created with XPAND=BASINF.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *SHBAS / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,OUTPUT = (address, length) address: <var: pointer> / (<reg: pointer>) length: <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

*SHBAS

Output basic information.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file whose basic information is to be output. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

OUTPUT= (address,length)

Output area for the basic information. For a detailed description, see page 81.

address: <var: pointer> / (<reg: pointer>)

Operand value	Meaning
<var: pointer>:	Alternatively, the following two specifications are possible: - address of the output area as a direct address specification in the format A(field) - name of the field containing the address of the output area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the output area

length : <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000>

Operand value	Meaning
<integer 0..1000000>:	Length in bytes of the output area as a direct specification
<var: int:4: integer 0..1000000>:	Name of the field containing the number of bytes as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1000000>:	Register containing the length specification for the output area; only permitted in conjunction with MF=M

Required layout areas:**Basic information (XPAND=BASINF)**

154	1	*	Basis-Information		
155	1	GCLDBIVS	DS	H	version of interface
156	1	GCLDBIND	DS	H	number of days in calendar
157	1	GCLDBINS	DS	H	number of SYMDAT in calendar
158	1	GCLDBINH	DS	H	number of holidays in calendar
159	1	GCLDBICN	DS	CL54	calendar name
160	1	GCLDBIL1	DS	CL10	first date in calendar
161	1	GCLDBIL2	DS	CL10	last date in calendar
162	1	GCLDBIFS	DS	CL20	first SYMDAT in calendar
163	1	GCLDBILS	DS	CL20	last SYMDAT in calendar
164	1	GCLDBIFH	DS	CL30	first holiday in calendar
165	1	GCLDBILH	DS	CL30	last holiday in calendar
166	1	GCLDBIS	DS	7CL14	area for standard week entry
167	1	*			is described with
168	1	*			XPAND=ENTSTDW
169	1	GCLDBIS#	EQU	7	
170	1	GCLDOBI#	EQU	*-GCLDBIVS	

Standard working week (XPAND = ENTSTDW)

11	1	*	Standard-week-entry		
12	1	GCLDWEDY	DS	CL3	weekday (MON - SUN)
13	1	GCLDWEAT	DS	CL1	attribute of day (W=workday, F=free day)
14	1	*			
15	1	GCLDWET1	DS	CL5	beginning of working hours
16	1	GCLDWET2	DS	CL5	end of working hours
17	1	GCLDSWE#	EQU	*-GCLDWEDY	

4.6.20 *SHDAYHL Output calendar day with assigned holidays

The *SHDAYHL function is used to request output of a calendar day with associated holidays from the specified calendar file.

Only activated holidays are taken into consideration.

The range of output cannot be restricted; normally only one holiday should be assigned to a particular calendar day.

The layout of the output area is created with XPAND=DAYHOL.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *SHDAYHL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,DATE = <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,OUTPUT = (address, length) address: <var: pointer> / (<reg: pointer>) length: <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

***SHDAYHL**

Output calendar day with assigned holidays.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file from which output of the calendar day is requested. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

DATE=

Date of the calendar day to be output. For a detailed description, see page 77.

<c-string 10..10: date 10..10>

Date of the calendar day as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the date of the calendar day; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date of the calendar day; only permitted in conjunction with MF=M.

OUTPUT=

Output area of the calendar day and assigned holidays. For a detailed description, see page 81.

address: <var: pointer> / (<reg: pointer>)

Operand value	Meaning
<var: pointer>:	Alternatively, the following two specifications are possible: <ul style="list-style-type: none"> - address of the output area as a direct address specification in the format A(field) - name of the field containing the address of the output area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the output area

length : <integer 0..1000000> / <var: int:4: integer 0..1000000> /
<reg: int:4: integer 0..1000000>

Operand value	Meaning
<integer 0..1000000>:	Length in bytes of the output area as a direct specification
<var: int:4: integer 0..1000000>:	Name of the field containing the number of bytes as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1000000>:	Register containing the length specification for the output area; only permitted in conjunction with MF=M

Required layout areas:

Calendar day with assigned holidays (XPAND = DAYHOL)

215	1	*	day_holiday		
216	1	GCLDDHVS	DS	H	version of interface
217	1	GCLDDHAH	DS	H	number of assigned holidays
218	1	GCLDDHGH	DS	H	number of given holidays
219	1	GCLDDHDT	DS	CL10	date
220	1	GCLDDHA	DS	1024CL34	area for holidays entry is
221	1	*			described with XPAND=ENTHOL
222	1	GCLDDHA#	EQU	1024	
223	1	GCLDODH#	EQU	*-GCLDDHVS	

Holiday entry (XPAND = ENTHOL)

114	1	*	holiday-entry		
115	1	GCLDHEHN	DS	CL30	holiday name
116	1	GCLDHEHT	DS	CL1	holiday type (C=cyclic, N=non-cyclic)
117	1	*			holiday is assigned (Y=yes, N=no)
118	1	GCLDHEAS	DS	CL1	number of assigned days
119	1	*			
120	1	GCLDHEAD	DS	H	
121	1	GCLDHLE#	EQU	*-GCLDHEHN	

4.6.21 *SHDAYIN Output calendar day with assigned SYMDATs

The *SHDAYIN function is used to request output of a calendar day with assigned SYMDATs from the specified calendar file.

The layout of the output area is created with XPAND=DAYINF.

Format

Macro	Operands
CALENDR	<pre> MF = C / D / L / M / E ,FUNCT = *SHDAYIN / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,DATE = <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,OUTPUT = (address, length) address: <var: pointer> / (<reg: pointer>) length: <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000> ,NAME1 = *FIRST / <c-string 1..30: name 1..30> <var: char:30: date 1..30> / <reg: A(char:30: date 1..30)> ,NAME2 = *LAST / <c-string 1..30: name 1..30> <var: char:30: date 1..30> <reg: A(char:30: date 1..30)> ,NUMB = *NO / <integer -4096..4096> / <var: int:4: integer -4096..4096> / <reg: int:4: integer -4096..4096> </pre>

Operands**MF=**

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

***SHDAYIN**

Output day information (calendar day).

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file from which output of the calendar day is requested. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

DATE=

Date of the calendar day to be output. For a detailed description, see page 77.

<c-string 10..10: date 10..10>

Date of the calendar day as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the date of the calendar day; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date of the calendar day; only permitted in conjunction with MF=M.

OUTPUT= (address,length)

Output area for the calendar day and assigned SYMDATs. For a detailed description, see page 81.

address: <var: pointer> / (<reg: pointer>)

Operand value	Meaning
<var: pointer>:	Alternatively, the following two specifications are possible: <ul style="list-style-type: none"> - address of the output area as a direct address specification in the format A(field) - name of the field containing the address of the output area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the output area

length : <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000>

Operand value	Meaning
<integer 0..1000000>:	Length in bytes of the output area as a direct specification
<var: int:4: integer 0..1000000>:	Name of the field containing the number of bytes as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1000000>:	Register containing the length specification for the output area; only permitted in conjunction with MF=M

NAME1=

Name at which output of the assigned SYMDATs is to begin.

***FIRST**

The first SYMDAT entry in the calendar is the first SYMDAT to be output.

<c-string 1..30: name 1..30>

Name of the first SYMDAT to be output as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the SYMDAT name; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the SYMDAT name; only permitted in conjunction with MF=M.

NAME2=

Name at which output of the assigned SYMDATs is to end.

If the last SYMDAT to be output is defined by means of the NUMB operand, this operand is not evaluated.

***LAST**

The last SYMDAT entry in the calendar is the last SYMDAT to be output.

<c-string 1..30: name 1..30>

Name of the last SYMDAT to be output as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the SYMDAT name; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the SYMDAT name; only permitted in conjunction with MF=M.

NUMB=

Number of SYMDATs to be output beginning with the SYMDAT specified for NAME1. If your entry ≠ *NO, the NAME2 operand is not evaluated.

***NO**

Output is restricted by the specification in NAME2.

<integer -4096..4096>

Number of SYMDATs to be output as a direct specification.

<var: int:4: integer -4096..4096>

Name of the field containing the number of SYMDATs to be output as an integer; field length = 4 bytes.

<reg: int:4: integer -4096..4096>

Register containing the number of SYMDATs to be output; only permitted in conjunction with MF=M.

SORT=

Sort criterion. The assigned SYMDATs should be sorted either by name or by time.

***TIME**

Sort criterion: time.

***SYMDAT**

Sort criterion: SYMDAT name.

<var: enum-of _sort_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 196. This can be specified only in conjunction with MF=M.

<reg: enum-of _sort_s:1>

Register containing the value of the equate in right-justified format.

Required layout areas:**Calendar day with assigned SYMDATs (XPAND = DAYINF)**

188	1	*	day_information		
189	1	GCLDDIVS	DS	H	version of interface
190	1	GCLDDIAS	DS	H	number of assigned SYMDATs
191	1	GCLDDISS	DS	H	number of selected SYMDATs
192	1	GCLDDIGS	DS	H	number of given SYMDATs
193	1	GCLDDIDT	DS	CL10	date
194	1	GCLDDID	DS	CL3	weekday (MON - SUN)
195	1	GCLDDIA	DS	CL1	attribute (W=workday, F=free day)
196	1	*			day)
197	1	GCLDDIA1	DS	CL1	attribute defined for day
198	1	*			(W=workday, F=free day,
199	1	*			S=standard)
200	1	GCLDDIA2	DS	CL1	attribute of day in
201	1	*			standard week (W=workday,
202	1	*			F=free day)
203	1	GCLDDIHO	DS	CL1	holiday (Y=yes, N=no)
204	1	GCLDDIT1	DS	CL5	beginning of working hours
205	1	GCLDDIT2	DS	CL5	end of working hours
206	1	GCLDDIF1	DS	CL1	filler
207	1	GCLDDIS	DS	4096CL28	area for SYMDAT entry is
208	1	*			described with XPAND=ENTSYMS
209	1	GCLDDIS#	EQU	4096	
210	1	GCLDODI#	EQU	*-GCLDDIVS	

Short SYMDAT entry (XPAND = ENTSYMS)

106	1	*	SYMDAT_short-entry		
107	1	GCLDSSSN	DS	CL20	SYMDAT name
108	1	GCLDSSST	DS	CL8	time
109	1	GCLDSSE#	EQU	*-GCLDSSSN	

4.6.22 *SHHOLIN Output holiday with assigned calendar days

The *SHHOLIN function is used to request output of a holiday with the assigned calendar days from the specified calendar file.

The layout of the output area is created with XPAND=HOLINF.

In the case of a non-cyclic holiday, all calendar days within a time span defined by the user are output, even if they lie outside the calendar limits.

In the case of a cyclic holiday, only those calendar days that lie within the calendar limits are output.

If the calendar limits cover a period less than a year, it is possible that a cyclic holiday may not have any assigned calendar days. If this is the case, the counter remains at 0, but the date ("****-mm-dd") defined for that holiday is nevertheless output.

Format

Macro	Operands
CALENDR	<pre> MF = C / D / L / M / E ,FUNCT = *SHHOLIN / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..30: name 1..30> / <var: char:30: name 1..30> / <reg: A(char:30: name 1..30)> ,OUTPUT = (address, length) address: <var: pointer> / (<reg: pointer>) length: <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000> ,DATE1 = <c-string 10..10: date 10..10> / *FIRST / *NEXT / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,DATE2 = *LAST / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> </pre>

Macro	Operands
CALENDR (cont.)	,NUMB = *NO / <integer -4096..4096> / <var: int:4: integer -4096..4096> / <reg: int:4: integer -4096..4096>

Operands

MF=

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

***SHHOLIN**

Output holiday information.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page [195](#). This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file from which the output of the holiday is requested. For a detailed description, see page [75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the holiday to be output. For a detailed description, see page [77](#).

<c-string 1..30: name 1..30>

Name of the holiday as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the name of the holiday; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the name of the holiday; only permitted in conjunction with MF=M.

OUTPUT= (address,length)

Output area for the holiday and assigned calendar days. For a detailed description, see page 81.

address: <var: pointer> / (<reg: pointer>)

Operand value	Meaning
<var: pointer>:	Alternatively, the following two specifications are possible: <ul style="list-style-type: none"> - address of the output area as a direct address specification in the format A(field) - name of the field containing the address of the output area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the output area

length : <integer 0..1000000> / <var: int:4: integer 0..1000000> /

<reg: int:4: integer 0..1000000>

Operand value	Meaning
<integer 0..1000000>:	Length in bytes of the output area as a direct specification
<var: int:4: integer 0..1000000>:	Name of the field containing the number of bytes as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1000000>:	Register containing the length specification for the output area; only permitted in conjunction with MF=M

DATE1=

Date at which output of the assigned calendar days is to begin. For a detailed description, see page 79.

<c-string 10..10: date 10..10>

Name of the first assigned calendar day as a direct specification.

***FIRST**

The output of assigned calendar days begins with the first day in this calendar.

***NEXT**

The output of assigned calendar days begins with the day following the current date.

<var: char:10: date 10..10>

Name of the field containing the first assigned calendar day; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date; only permitted in conjunction with MF=M.

DATE2=

Date at which output of the assigned calendar days is to end.

If the last assigned calendar day is defined by means of the NUMB operand, DATE2 is not evaluated. For a detailed description, see page [79](#)

***LAST**

The last calendar day in the calendar is the last calendar day to be output.

<c-string 1..30: name 1..30>

Name of the last assigned calendar day as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the last assigned calendar day; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date; only permitted in conjunction with MF=M.

NUMB=

Number of calendar days to be output beginning with the calendar day specified for DATE1. If your entry \neq *NO, the DATE2 operand is not evaluated. For a detailed description, see page [80](#).

***NO**

Output is restricted by the specification in DATE2.

<integer -4096..4096>

Number of calendar days to be output as a direct specification.

<var: int:4: integer -4096..4096>

Name of the field containing the number of calendar days to be output as an integer; field length = 4 bytes.

<reg: int:4: integer -4096..4096>

Register containing the number of calendar days to be output; only permitted in conjunction with MF=M.

Required layout areas:**Holiday with assigned calendar days (XPAND = HOLINF)**

281	1	*	holiday-information		
282	1	GCLDHIVS	DS	H	version of interface
283	1	GCLDHIAD	DS	H	number of assigned days
284	1	GCLDHISD	DS	H	number of selected days
285	1	GCLDHIGD	DS	H	number of given days
286	1	GCLDHIHN	DS	CL30	holiday name
287	1	GCLDHIHT	DS	CL1	holiday type (C=cyclic N=non-cyclic)
288	1	*			holiday is assigned (Y=yes, N=no)
289	1	GCLDHIAS	DS	CL1	
290	1	*			
291	1	GCLDHIA	DS	1860CL10	area for assigned days entry is described with
292	1	*			
293	1	*			XPAND=ENTDATE
294	1	GCLDHIA#	EQU	1860	
295	1	GCLDOHI#	EQU	*-GCLDHIVS	

Date entry (XPAND = ENTDATE)

300	1	*	Date-entry		
301	1	GCLDDTY	DS	CL4	year
302	1	GCLDDTH1	DS	CL1	hyphen
303	1	GCLDDTMO	DS	CL2	month
304	1	GCLDDTH2	DS	CL1	hyphen
305	1	GCLDDTDY	DS	CL2	day
306	1	GCLDDTE#	EQU	*-GCLDDTY	

4.6.23 *SHLODAY Output list of calendar days

The *SHLODAY function is used to request output of a list of calendar days from the specified calendar file.

The layout of the output area is created using XPAND=LODAY.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *SHLODAY / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,OUTPUT = (address, length) address: <var: pointer> / (<reg: pointer>) length: <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000> ,DATE1 = <c-string 10..10: date 10..10> / *FIRST / *NEXT / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,DATE2 = *LAST / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,NUMB = *NO / <integer -4096..4096> / <var: int:4: integer -4096..4096> / <reg: int:4: integer -4096..4096></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

***SHLODAY**

Output list of calendar days.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file from which output of the list of calendar days is requested. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

OUTPUT= (address,length)

Output area for the list of calendar days. For a detailed description, see page 81.

address: <var: pointer> / (<reg: pointer>)

Operand value	Meaning
<var: pointer>:	Alternatively, the following two specifications are possible: <ul style="list-style-type: none"> - address of the output area as a direct address specification in the format A(field) - name of the field containing the address of the output area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the output area

length : <integer 0..1000000> / <var: int:4: integer 0..1000000> /
<reg: int:4: integer 0..1000000>

Operand value	Meaning
<integer 0..1000000>:	Length in bytes of the output area as a direct specification
<var: int:4: integer 0..1000000>:	Name of the field containing the number of bytes as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1000000>:	Register containing the length specification for the output area; only permitted in conjunction with MF=M

DATE1=

First calendar day to be output. For a detailed description, see page [79](#).

<c-string 10..10: date 10..10>

Name of the first calendar day to be output as a direct specification.

***FIRST**

The output of calendar days begins with the first day in this calendar.

***NEXT**

The output of calendar days begins with the day following the current date.

<var: char:10: date 10..10>

Name of the field containing the first calendar day to be output; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the name of the calendar day; only permitted in conjunction with MF=M.

DATE2=

Last calendar day to be output.

If the last calendar day is defined by means of the NUMB operand, DATE2 is not evaluated.

For a detailed description, see page [79](#).

***LAST**

The last calendar day in the calendar is the last calendar day to be output.

<c-string 1..10: name 1..10>

Name of the last calendar day to be output as a direct specification.

<var: char:10: name 1..10>

Name of the field containing the last calendar day to be output; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the name of the calendar day; only permitted in conjunction with MF=M.

NUMB=

Number of calendar days to be output, beginning with the calendar day specified for DATE1. If your entry \neq *NO, the DATE2 operand is not evaluated. For a detailed description, see page [80](#).

***NO**

Output is restricted by the specification in DATE2.

<integer -4096..4096>

Number of calendar days to be output as a direct specification.

<var: int:4: integer -4096..4096>

Name of the field containing the number of calendar days to be output as an integer; field length = 4 bytes.

<reg: int:4: integer -4096..4096>

Register containing the number of calendar days to be output; only permitted in conjunction with MF=M.

Required layout areas:**List of calendar days (XPAND = LODAY)**

175	1	*	List-of-days		
176	1	GCLDLDVS	DS	H	version of interface
177	1	GCLDLDAD	DS	H	number of days in calendar
178	1	GCLDLDSD	DS	H	number of selected days
179	1	GCLDLDGD	DS	H	number of given days
180	1	GCLDLDA	DS	1860CL32	area for days entry is
181	1	*			described with XPAND=ENTDAY
182	1	GCLDLDA#	EQU	1860	
183	1	GCLDOLD#	EQU	*-GCLDLDVS	

Calendar day (XPAND = ENTDAY)

64	1	*	Day-entry		
65	1	GCLDDEDT	DS	CL10	date
66	1	GCLDDED	DS	CL3	weekday (MON - SUN)
67	1	GCLDDEA	DS	CL1	attribute (W=workday, F=free day)
68	1	*			
69	1	GCLDDEAD	DS	CL1	attribute defined for day
70	1	*			(W=workday, F=free day, S=standard)
71	1	*			
72	1	GCLDDEAS	DS	CL1	attribute of day in
73	1	*			standard week (W=workday, F=free day)
74	1	*			
75	1	GCLDDEHO	DS	CL1	holiday (Y=yes, N=no)
76	1	GCLDDET1	DS	CL5	beginning of working hours
77	1	GCLDDET2	DS	CL5	end of working hours
78	1	GCLDDEFI	DS	CL1	filler
79	1	GCLDDEH	DS	H	number of assigned holidays
80	1	GCLDDES	DS	H	number of assigned SYMDATS
81	1	GCLDDYE#	EQU	*-GCLDDEDT	

4.6.24 *SHLOHOL Output list of holidays

The *SHLOHOL function is used to output a list of holidays from the specified calendar file. The layout of the output area is created with XPAND=LOHOL.

Format

Macro	Operands
CALENDR	<pre> MF = C / D / L / M / E ,FUNCT = *SHLOHOL / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,OUTPUT = (address, length) address: <var: pointer> / (<reg: pointer>) length: <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000> ,NAME1 = *FIRST / <c-string 1..30: name 1..30> <var: char:30: date 1..30> / <reg: A(char:30: date 1..30)> ,NAME2 = *LAST / <c-string 1..30: name 1..30> <var: char:30: date 1..30> / <reg: A(char:30: date 1..30)> ,NUMB = *NO / <integer -4096..4096> / <var: int:4: integer -4096..4096> / <reg: int:4: integer -4096..4096> </pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

***SHLOHOL**

Output list of holidays.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file from which output of a list of holidays is requested. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

OUTPUT=

Output area for the list of holidays. For a detailed description, see page 81.

address: <var: pointer> / (<reg: pointer>)

Operand value	Meaning
<var: pointer>:	Alternatively, the following two specifications are possible: <ul style="list-style-type: none"> - address of the output area as a direct address specification in the format A(field) - name of the field containing the address of the output area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the output area

length : <integer 0..1000000> / <var: int:4: integer 0..1000000> /
 <reg: int:4: integer 0..1000000>

Operand value	Meaning
<integer 0..1000000>:	Length in bytes of the output area as a direct specification
<var: int:4: integer 0..1000000>:	Name of the field containing the number of bytes as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1000000>:	Register containing the length specification for the output area; only permitted in conjunction with MF=M

NAME1=

First holiday to be output.

***FIRST**

The first holiday in the calendar is the first holiday to be output.

<c-string 1..30: name 1..30>

Name of the first holiday to be output as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the first holiday to be output; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the name of the holiday; only permitted in conjunction with MF=M.

NAME2=

Last holiday to be output.

If the last holiday to be output is defined by means of the NUMB operand, this operand is not evaluated.

***LAST**

The last holiday in the calendar is the last holiday to be output.

<c-string 1..30: name 1..30>

Name of the last holiday to be output as a direct specification.

<var: char:30: name 1..30>

Name of the field containing the last holiday to be output; only permitted in conjunction with MF=M.

<reg: A(char:30: name 1..30)>

Register with the address of the field containing the name of the holiday; only permitted in conjunction with MF=M.

NUMB=

Number of holidays to be output, beginning with the holiday specified for NAME1.

If your entry \neq *NO, the NAME2 operand is not evaluated. For a detailed description, see page 80.

***NO**

Output is restricted by the specification in NAME2.

<integer -4096..4096>

Number of calendar days to be output as a direct specification.

<var: int:4: integer -4096..4096>

Name of the field containing the number of calendar days to be output as an integer; field length = 4 bytes.

<reg: int:4: integer -4096..4096>

Register containing the number of holidays to be output; only permitted in conjunction with MF=M.

Required layout areas:**List of holidays (XPAND = LOHOL)**

268	1	*	List-of-holidays		
269	1	GCLDLHVS	DS	H	version of interface
270	1	GCLDLHAH	DS	H	number of holidays in calendar
271	1	GCLDLHSH	DS	H	number of selected holidays
272	1	GCLDLHGH	DS	H	number of given holidays
273	1	GCLDLHA	DS	1024CL34	area for holidays entry is
274	1	*			described with XPAND=ENTHOL
275	1	GCLDLHA#	EQU	1024	
276	1	GCLDOLH#	EQU	*-GCLDLHVS	

Holiday (XPAND = ENTHOL)

114	1	*	holiday-entry		
115	1	GCLDHEHN	DS	CL30	holiday name
116	1	GCLDHEHT	DS	CL1	holiday type (C=cyclic, N=non-cyclic)
117	1	*			
118	1	GCLDHEAS	DS	CL1	holiday is assigned (Y=yes, N=no)
119	1	*			
120	1	GCLDHEAD	DS	H	number of assigned days
121	1	GCLDHLE#	EQU	*-GCLDHEHN	

4.6.25 *SHLOSYM Output list of SYMDATs

The *SHLOSYM function is used to request output of a list of SYMDATs from the specified calendar file.

The layout of the output area is created using XPAND=LOSYM.

Format

Macro	Operands
CALENDR	<pre>MF = C / D / L / M / E ,FUNCT = *SHLOSYM / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,OUTPUT = (address, length) address: <var: pointer> / (<reg: pointer>) length: <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000> ,NAME1 = *FIRST / <c-string 1..20: name 1..20> <var: char:20: date 1..20> / <reg: A(char:20: date 1..20)> ,NAME2 = *LAST / <c-string 1..20: name 1..20> <var: char:20: date 1..20> / <reg: A(char:20: date 1..20)> ,NUMB = *NO / <integer -4096..4096> / <var: int:4: integer -4096..4096> / <reg: int:4: integer -4096..4096></pre>

Operands

MF=

The MF operand controls the form of macro expansion; see page 201ff.

FUNCT=

Function code with which the desired calendar function is specified.

***SHLOSYM**

Output list of SYMDATs (symbolic dates).

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file from which output of the list of SYMDATs is requested. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

OUTPUT=

Output area for the list of SYMDATs. For a detailed description, see page 81.

address: <var: pointer> / (<reg: pointer>)

Operand value	Meaning
<var: pointer>:	Alternatively, the following two specifications are possible: - address of the output area as a direct address specification in the format A(field) - name of the field containing the address of the output area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the output area

length : <integer 0..1000000> / <var: int:4: integer 0..1000000>
 <reg: int:4: integer 0..1000000>

<integer 0..1000000>:	Length in bytes of the output area as a direct specification
<var: int:4: integer 0..1000000>:	Name of the field containing the number of bytes as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1000000>:	Register containing the length specification for the output area; only permitted in conjunction with MF=M

NAME1=

First SYMDAT to be output.

***FIRST**

The first SYMDAT in the calendar is the first SYMDAT to be output.

<c-string 1..20: name 1..20>

Name of the first SYMDAT to be output as a direct specification.

<var: char:20: name 1..20>

Name of the field containing the first SYMDAT to be output; only permitted in conjunction with MF=M.

<reg: A(char:20: name 1..20)>

Register with the address of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

A detailed description of the NAME1 operand can be found on page [80](#).

NAME2=

Last SYMDAT to be output.

If the last SYMDAT to be output is defined by means of the NUMB operand, NAME2 is not evaluated.

***LAST**

The last SYMDAT in the calendar is the last SYMDAT to be output.

<c-string 1..20: name 1..20>

Name of the last SYMDAT to be output as a direct specification.

<var: char:20: name 1..20>

Name of the field containing the last SYMDAT to be output; only permitted in conjunction with MF=M.

<reg: A(char:20: name 1..20)>

Register with the address of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

NUMB=

Number of SYMDATs to be output, beginning with the SYMDAT specified for NAME1. If your entry \neq *NO, the NAME2 operand is not evaluated. For a detailed description, see page 80.

***NO**

Output is restricted by the specification in NAME2.

<integer -4096..4096>

Number of SYMDATs to be output as a direct specification.

<var: int:4: integer -4096..4096>

Name of the field containing the number of SYMDATs to be output as an integer; field length = 4 bytes.

<reg: int:4: integer -4096..4096>

Register containing the number of SYMDATs to be output; only permitted in conjunction with MF=M.

Required layout areas:**List of SYMDATs (XPAND = LOSYM)**

228	1	*	List-of-SYMDAT		
229	1	GCLDLSVS	DS	H	version of interface
230	1	GCLDLSAS	DS	H	number of SYMDATs in calendar
231	1	GCLDLSSS	DS	H	number of selected SYMDATs
232	1	GCLDLSGS	DS	H	number of given SYMDATs
233	1	GCLDLSA	DS	4096CL56	area for SYMDAT entry is
234	1	*			described with XPAND=ENTSYM
235	1	GCLDLSA#	EQU	4096	
236	1	GCLDOLS#	EQU	*-GCLDLSVS	

SYMDAT entry (XPAND = ENTSYM)

86	1	*	SYMDAT-entry		
87	1	GCLDSESN	DS	CL20	SYMDAT name
88	1	GCLDSEST	DS	CL1	SYMDAT type (C=cyclic, N=non-cyclic)
89	1	*			
90	1	GCLDSET	DS	CL8	time
91	1	GCLDSECT	DS	CL1	kind of cycle (D=day, W=week, O=workday, M=month)
92	1	*			
93	1	GCLDSECV	DS	H	cycle value
94	1	GCLDSEAL	DS	CL1	alternative for free days
95	1	*			(B=before, A=after, S=skip, O=on)
96	1	*			
97	1	GCLDSEF1	DS	CL1	filler
98	1	GCLDSED1	DS	CL10	start date for cyclic SYMDAT
99	1	GCLDSED2	DS	CL10	end date for cyclic SYMDAT
100	1	GCLDSEAD	DS	H	number of assigned days
101	1	GCLDSYE#	EQU	*-GCLDSESN	

4.6.26 *SHSYMIN Output SYMDAT with assigned calendar days

The *SHSYMIN function is used to request output of a SYMDAT with the assigned calendar days from the specified calendar file.

The layout of the output area is created with XPAND=SYMINF.

Format

Macro	Operands
CALENDR	<pre> MF = C / D / L / M / E ,FUNCT = *SHSYMIN / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,NAME = <c-string 1..20: name 1..20> / <var: char:20: name 1..20> / <reg: A(char:20: name 1..20)> ,OUTPUT = (address, length) address: <var: pointer> / (<reg: pointer>) length: <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000> ,DATE1 = <c-string 10..10: date 10..10> / *FIRST / *NEXT / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,DATE2 = *LAST / <c-string 10..10: date 10..10> / <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,NUMB = *NO / <integer -4096..4096> / <var: int:4: integer -4096..4096> / <reg: int:4: integer -4096..4096> </pre>

Operands**MF=**

The MF operand controls the form of macro expansion; see page [201ff](#).

FUNCT=

Function code with which the desired calendar function is specified.

***SHSYMIN**

Output SYMDAT information.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form; see page 195. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file from which output of the SYMDAT and assigned calendar days is requested. For a detailed description, see page 75.

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

NAME=

Name of the SYMDAT to be output. For a detailed description, see page 77.

<c-string 1..20: name 1..20>

Name of the SYMDAT as a direct specification.

<var: char:20: name 1..20>

Name of the field containing the SYMDAT; only permitted in conjunction with MF=M.

<reg: A(char:20: name 1..20)>

Register with the address of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

OUTPUT=

Output area for the SYMDAT and assigned calendar days. For a detailed description, see page 81.

address: <var: pointer> / (<reg: pointer>)

Operand value	Meaning
<var: pointer>:	Alternatively, the following two specifications are possible: <ul style="list-style-type: none"> - address of the output area as a direct address specification in the format A(field) - name of the field containing the address of the output area; only permitted in conjunction with MF=M
(<reg: pointer>):	Register with the address of the output area

**length : <integer 0..1000000> / <var: int:4: integer 0..1000000>
<reg: int:4: integer 0..1000000>**

Operand value	Meaning
<integer 0..1000000>:	Length in bytes of the output area as a direct specification
<var: int:4: integer 0..1000000>:	Name of the field containing the number of bytes as an integer; field length = 4 bytes; only permitted in conjunction with MF=M
<reg: int:4: integer 0..1000000>:	Register containing the length specification for the output area; only permitted in conjunction with MF=M

DATE1=

First assigned calendar day to be output. For a detailed description, see page [79](#).

<c-string 10..10: date 10..10>

Date of the first calendar day to be output as a direct specification.

***FIRST**

The output of assigned calendar days begins with the first day in this calendar.

***NEXT**

The output of assigned calendar days begins with the day following the current date.

<var: char:10: date 10..10>

Name of the field containing the date of the first calendar day to be output; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date of the calendar day; only permitted in conjunction with MF=M.

DATE2=

Last assigned calendar day to be output.

If the last assigned calendar day is defined by means of the NUMB operand, DATE2 is not evaluated. For a detailed description, see page [79](#).

***LAST**

The last calendar day in the calendar is the last calendar day to be output.

<c-string 1..10: name 1..10>

Date of the last calendar day to be output.

<var: char:10: name 1..10>

Name of the field containing the date of the last calendar day to be output; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date of the calendar day; only permitted in conjunction with MF=M.

NUMB=

Number of calendar days to be output, beginning with the calendar day specified for DATE1. If your entry ≠ *NO, the DATE2 operand is not evaluated. For a detailed description, see page [80](#).

***NO**

Output is restricted by the specification in DATE2.

<integer -4096..4096>

Number of calendar days to be output as a direct specification.

<var: int:4: integer -4096..4096>

Name of the field containing the number of calendar days to be output as an integer; field length = 4 bytes.

<reg: int:4: integer -4096..4096>

Register containing the number of calendar days to be output; only permitted in conjunction with MF=M.

Required layout areas:**SYMDAT with assigned calendar days (XPAND = SYMINF)**

241	1 *	SYMDAT_information		
242	1	GCLDSIVS DS	H	version of interface
243	1	GCLDSIAD DS	H	number of assigned days
244	1	GCLDSISD DS	H	number of selected days
245	1	GCLDSIGD DS	H	number of given days
246	1	GCLDSISN DS	CL20	SYMDAT name
247	1	GCLDSIST DS	CL1	SYMDAT type (C=cyclic, N=non-cyclic)
248	1 *			
249	1	GCLDSIT DS	CL8	time
250	1	GCLDSICT DS	CL1	kind of cycle (D=day, W=week, O=workday, M=month)
251	1 *			
252	1	GCLDSICV DS	H	cycle value
253	1	GCLDSIAL DS	CL1	alternative for free days
254	1 *			(B=before, A=after, S=skip, O=on)
255	1 *			
256	1	GCLDSID1 DS	CL10	start date for cyclic SYMDAT
257	1	GCLDSID2 DS	CL10	end date for cyclic SYMDAT
258	1	GCLDSIF1 DS	CL1	filler1
259	1	GCLDSIA DS	1860CL10	area for assigned days entry
260	1 *			is described with
261	1 *			XPAND=ENTDATE
262	1	GCLDSIA# EQU	1860	
263	1	GCLDOSI# EQU	*-GCLDSIVS	

Date entry (XPAND = ENTDATE)

300	1 *	Date-entry		
301	1	GCLDDTY DS	CL4	year
302	1	GCLDDTH1 DS	CL1	hyphen
303	1	GCLDDTMO DS	CL2	month
304	1	GCLDDTH2 DS	CL1	hyphen
305	1	GCLDDTDY DS	CL2	day
306	1	GCLDDTE# EQU	*-GCLDDTY	

4.6.27 *SHNESTM Output next time for a set of SYMDATs

The *SHNESTM function is used to request the next time after a particular start time for a set of SYMDATs out of the specified calendar file. The output area layout is created with XPAND=NESTM.

Format

Makro	Operanden
CALENDR	<pre> MF = C / D / L / M / E ,FUNCT = *SHNESTM / <var: enum-of _funct_s:1> / <reg: enum-of _funct_s:1> ,VERSION = 1 ,CALLER = <u>USER</u> / SYSTEM ,CCB = <var: pointer> / (<reg: pointer>) ,OUTPUT = (adresse, länge) adresse: <var: pointer> / (<reg: pointer>) länge: <integer 0..1000000> / <var: int:4: integer 0..1000000> / <reg: int:4: integer 0..1000000> ,NAME1 = *FIRST / <c-string 1..20: name 1..20> / <var: char:20: name 1..20> / <reg: A(char:20: name 1..20)> ,NAME2 = *LAST / <c-string 1..20: name 1..20> / <var: char:20: name 1..20> / <reg: A(char:20: name 1..20)> ,DATE = <c-string 10..10: date 10..10> <var: char:10: date 10..10> / <reg: A(char:10: date 10..10)> ,TIME = <c-string 8..8: time 8..8> / <var: char:8: time 8..8> / <reg: A(char:8: time 8..8)> </pre>

Operandenbeschreibung

MF=

The MF operand controls the form of the macro resolution, see [page 201](#).

FUNCT=

Function code, with which the desired calendar function is specified.

***SHNESTM**

Output next time for a set of SYMDATs.

<var: enum-of _funct_s:1>

Name of the field with the value of the equate. The equates are defined in the C or D form. This can be specified only in conjunction with MF=M.

<reg: enum-of _funct_s:1>

Register containing the value of the equate in right-justified format.

VERSION = 1

Version 1. This is the only version supported at present.

CALLER = USER / SYSTEM

Caller of the interface: user (TU) or system (TPR).

CCB=

Control block of the calendar file from which output of the SYMDAT and assigned calendar days is requested. For a detailed description, see [page 75](#).

<var: pointer>

Alternatively, the following two specifications are possible:

- address of the control block of this calendar file as a direct address specification in the format A(field)
- name of the field containing the address of the control block; only permitted in conjunction with MF=M

(<reg: pointer>)

Register with the address of the control block.

OUTPUT=

Output area for the SYMDAT and assigned calendar days. For a detailed description, see [page 81](#).

adresse: <var: pointer> / (<reg: pointer>)

Operandenwert	Bedeutung
<var: pointer>:	Alternatively, the following two specifications are possible: - address of the output area as a direct address specification in the format A(field) - name of the field containing the address of the output area; only permitted in conjunction with MF=M.
(<reg: pointer>):	Register with the address of the output area.

länge : <integer 0..1000000> / <var: int:4: integer 0..1000000>
 <reg: int:4: integer 0..1000000>

Operandenwert	Bedeutung
<integer 0..1000000>:	Length in bytes of the output area as a direct specification.
<var: int:4: integer 0..1000000>:	Name of the field containing the number of bytes as an integer; field length = 4 bytes; only permitted in conjunction with MF=M.
<reg: int:4: integer 0..1000000>:	Register containing the length specification for the output area; only permitted in conjunction with MF=M.

NAME1=

Name of the first SYMDAT from which times are to be considered. A detailed description of the NAME1 operand can be found on [page 80](#).

***FIRST**

The first SYMDAT in the calendar is the lower limit.

<c-string 1..20: name 1..20>

Name of the first SYMDAT as a direct specification.

<var: char:20: name 1..20>

Name of the field containing the first SYMDAT; only permitted in conjunction with MF=M.

<reg: A(char:20: name 1..20)>

Register with the address of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

<reg: A(char:20: name 1..20)>

Register with the address of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

NAME2=

Name of the last SYMDAT from which times are to be considered. A detailed description of the NAME1 operand can be found on [page 80](#).

***LAST**

The last SYMDAT in the calendar is the last SYMDAT to be output.

<c-string 1..20: name 1..20>

Name of the last SYMDAT as a direct specification.

<var: char:20: name 1..20>

Name of the field containing the last SYMDAT; only permitted in conjunction with MF=M.

<reg: A(char:20: name 1..20)>

Register with the address of the field containing the name of the SYMDAT; only permitted in conjunction with MF=M.

DATE=

Date as of which (together with the TIME) the next time is to be determined. For a detailed description, see [page 77](#).

<c-string 10..10: date 10..10>

Date of the calendar day as a direct specification.

<var: char:10: date 10..10>

Name of the field containing the date of the calendar day; only permitted in conjunction with MF=M.

<reg: A(char:10: date 10..10)>

Register with the address of the field containing the date of the calendar day; only permitted in conjunction with MF=M.

TIME=

Time as of which (together with the DATE) the next time is to be determined. This is specified in the format hh:mm:ss (hh=hours, mm=minutes, ss=seconds). For a detailed description [page 78](#).

<c-string 8..8: time 8..8>

Time specification as a direct specification.

<var: char:8: time 8..8>

Name of the field containing the time specification; only permitted in conjunction with MF=M.

<reg: A(char:8: time 8..8)>

Register with the address of the field containing the time specification; only permitted in conjunction with MF=M.

Required layout areas:

SYMDAT with assigned calendar days ((XPAND = NESTM))

264	1	*	List-of-SYMDAT		
265	1	GCLDNSVS	DS	H	version of interface
266	1	GCLDNSTD	DS	CL10	next date
267	1	GCLDNSTT	DS	CL8	next time
268	1	GCLDNSNM	DS	CL20	SYMDAT name
269	1	*			
269	1	GCLDONS#	EQU	*-GCLDNSVS	

4.7 Parameter list

```

    PARLIST CALENDR MF=D,VERSION=1,XPAND=PARAM
1 PARLIST MFTST MF=D,PREFIX=G,MACID=CLD,ALIGN=F,          *
1          DMACID=CLD,SUPPORT=(E,D,C,M,L),DNAME=CLDPMDL
2 PARLIST DSECT ,
2          *,##### PREFIX=G, MACID=CLD #####
1 *   parameterarea description
1 GCLDHDR FHDR MF=(C,GCLD),EQUATES=NO      Standard-Header
2 GCLDHDR DS      0A
2 GCLDFHE DS      OXL8                    0   GENERAL PARAMETER AREA HEADER
2 *
2 GCLDIFID DS      0A                    0   INTERFACE IDENTIFIER
2 GCLDFCTU DS      AL2                    0   FUNCTION UNIT NUMBER
2 *
2 *                                     BIT 15   HEADER FLAG BIT,
2 *                                     MUST BE RESET UNTIL FURTHER NOTICE
2 *                                     BIT 14-12 UNUSED, MUST BE RESET
2 *                                     BIT 11-0   REAL FUNCTION UNIT NUMBER
2 GCLDFCT DS      AL1                    2   FUNCTION NUMBER
2 GCLDFCTV DS      AL1                    3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 GCLDRET DS      0A                    4   GENERAL RETURN CODE
2 GCLDSRET DS      0AL2                  4   SUB RETURN CODE
2 GCLDSR2 DS      AL1                    4   SUB RETURN CODE 2
2 GCLDSR1 DS      AL1                    5   SUB RETURN CODE 1
2 GCLDMRET DS      0AL2                  6   MAIN RETURN CODE
2 GCLDMR2 DS      AL1                    6   MAIN RETURN CODE 2
2 GCLDMR1 DS      AL1                    7   MAIN RETURN CODE 1
2 GCLDFHL EQU      8                    8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *   main return codes
1 GCLDRSUC EQU      0                    no error
1 GCLDRINT EQU      1                    parameter error
1 GCLDRCEX EQU      2                    calendar already exists
1 GCLDRCNE EQU      3                    calendar does not exist
1 GCLDRDEC EQU      4                    DMS error on calendar file
1 GCLDRNCL EQU      5                    file is not a calendar
1 GCLDRCIN EQU      6                    calendar is inconsistent
1 GCLDRNUP EQU      7                    update of calendar not
1 *                                     possible at the moment
1 GCLDRHFN EQU      8                    holiday file does not exist
1 GCLDRDHF EQU      9                    DMS error on holiday file
1 GCLDRTMD EQU     10                   too many days between calendar
1 *                                     limits
1 GCLDRELS EQU     11                   end date before start date
1 GCLDRNME EQU     12                   no more memory available
1 GCLDRCBI EQU     13                   control block invalid

```

1	GCLDRDTE	EQU	14	date outside of calendar
1	*			limits
1	GCLDRSYE	EQU	15	SYMDAT already exists
1	GCLDRSYN	EQU	16	SYMDAT does not exist
1	GCLDRCSY	EQU	17	SYMDAT is a cyclic SYMDAT
1	GCLDRNSY	EQU	18	SYMDAT is a non-cyclic SYMDAT
1	GCLDRDTE	EQU	19	assignment to date already
1	*			exists
1	GCLDRDTN	EQU	20	assignment to date does not
1	*			exist
1	GCLDRDTI	EQU	21	invalid date
1	GCLDRHLE	EQU	22	holiday already exists
1	GCLDRHLN	EQU	23	holiday does not exist
1	GCLDRCHL	EQU	24	holiday is a cyclic holiday
1	GCLDRNHL	EQU	25	holiday is a non-cyclic
1	*			holiday
1	GCLDRHAC	EQU	26	holiday already activated
1	GCLDRHNA	EQU	27	holiday is not activated
1	GCLDRNIF	EQU	28	output area too small: no
1	*			information available
1	GCLDRSIF	EQU	29	output area too small: some
1	*			information available
1	GCLDRWSY	EQU	30	definition of SYMDAT area
1	*			invalid
1	GCLDRWHL	EQU	31	definition of holiday area
1	*			invalid
1	GCLDRSWI	EQU	32	area for standard week is
1	*			incorrect
1	GCLDRHFI	EQU	33	entry of holiday file is
1	*			incorrect
1	GCLDRILI	EQU	34	LIMIT1 < 1970 or LIMIT2 > 2030
1	*			or new LIMIT1 > old LIMIT2
1	GCLDRSDO	EQU	35	start date of cyclic SYMDAT
1	*			outside of calendar limits
1	*			(warning)
1	GCLDRFNA	EQU	36	function not allowed with
1	*			OPENMOD = *READ or *READALL
1	GCLDRSDD	EQU	37	start date for cyclic SYMDAT
1	*			with CYCLTYP=WORKDAY has been
1	*			deleted as it is outside
1	*			calendar limits
1	GCLDRPRE	EQU	38	internal error in program
1	GCLDRCVI	EQU	39	version of calendar is not
1	*			supported by version of
1	*			interface
1	GCLDRTMS	EQU	40	more than 4096 SYMDATs have
1	*			been created
1	GCLDRTMH	EQU	41	more than 1024 holidays have

1 *			been created
1 GCLDRFNS	EQU	65535	function not supported
1 GCLDRVNA	EQU	65535	version not supported
1 GCLDRAER	EQU	65535	alignment error
1 *			
1 GCLDPFCT	DS	FL1	funct =
1 *	function	codes	
1 GCLDFCRC	EQU	1	create calendar
1 GCLDFOPC	EQU	2	open calendar
1 GCLDFCLC	EQU	3	close calendar
1 GCLDFSVC	EQU	4	save calendar
1 GCLDFMDB	EQU	5	modify basic information
1 GCLDFMDY	EQU	6	modify day
1 GCLDFCNS	EQU	7	create non-cyclic symdat
1 GCLDFCCS	EQU	8	create cyclic symdat
1 GCLDFDES	EQU	9	delete symdat
1 GCLDFMNS	EQU	10	modify non-cyclic symdat
1 GCLDFMCS	EQU	11	modify cyclic symdat
1 GCLDFCNH	EQU	12	create non-cyclic holiday
1 GCLDFCCH	EQU	13	create cyclic holiday
1 GCLDFDEH	EQU	14	delete holiday
1 GCLDFMNH	EQU	15	modify non-cyclic holiday
1 GCLDFMCH	EQU	16	modify cyclic holiday
1 GCLDFAKH	EQU	17	activate holiday
1 GCLDFDAH	EQU	18	deactivate holiday
1 GCLDFSBA	EQU	19	show basic information
1 GCLDFS LD	EQU	20	show list of days
1 GCLDFS DI	EQU	21	show day information
1 GCLDFS DH	EQU	22	show day with holidays
1 GCLDFS LS	EQU	23	show list of symdats
1 GCLDFS SI	EQU	24	show symdat information
1 GCLDFS LH	EQU	25	show list of holidays
1 GCLDFS HI	EQU	26	show holiday information
1 *			
1 GCLDPOPM	DS	FL1	openmod =
1 *	open	modus	
1 GCLDREAD	EQU	1	read
1 GCLDRDAL	EQU	2	read complete calendar
1 GCLDUPDT	EQU	3	update
1 *			
1 GCLDPATR	DS	FL1	attribute =
1 *	attribute	of day	
1 GCLDATUC	EQU	0	unchanged
1 GCLDFREE	EQU	1	free day
1 GCLDWORK	EQU	2	workday
1 GCLDSTWK	EQU	3	standard week
1 *			
1 GCLDPTYP	DS	FL1	cycltyp =

1 *	kind of cycle		
1	GCLDCYUC	EQU 0	unchanged
1	GCLDDAY	EQU 1	cycle = day
1	GCLDWKD	EQU 2	cycle = workday
1	GCLDWEK	EQU 3	cycle = week
1	GCLDMON	EQU 4	cycle = month
1 *			
1	GCLDPALT	DS FL1	alternative =
1 *	alternative		
1	GCLDALUC	EQU 0	unchanged
1	GCLDBEFR	EQU 1	insert before free day
1	GCLDAFT	EQU 2	insert after free day
1	GCLDSKIP	EQU 3	don't insert
1	GCLDON	EQU 4	insert on free day
1 *			
1	GCLDPSAV	DS FL1	save =
1 *	save		
1	GCLDYES	EQU 1	save calendar
1	GCLDNO	EQU 2	don't save calendar
1 *			
1	GCLDPSOR	DS FL1	sort =
1 *	sort		
1	GCLDSSYM	EQU 1	sort by symdat
1	GCLDSTIM	EQU 2	sort by time
1 *			
1	GCLDPFL1	DS AL1	indicator
1	GCLDPRS1	EQU X'FF'	reserved
1	GCLDPFL2	DS AL1	indicator
1	GCLDPRS2	EQU X'FF'	reserved
1	GCLDPFI1	DS CL3	filler
1	GCLDPNUM	DS F	number of days or SYMDATs
1	GCLDPCVL	DS F	number of days between two
1 *			entries
1	GCLDPDMS	DS F	DMS return code
1	GCLDPCCB	DS A	address of control block
1	GCLDPWKA	DS A	address of standard week
1	GCLDPADD	DS A	address of days to add
1	GCLDPAD#	DS F	number of days in area
1	GCLDPDEL	DS A	address of days to delete
1	GCLDPDE#	DS F	number of days in area
1	GCLDPOUT	DS A	address of output area
1	GCLDPOTL	DS F	length of output area
1	GCLDPCLN	DS CL54	calendar name
1	GCLDPNAM	DS CL30	name of SYMDAT or holiday
1	GCLDPNM1	DS CL30	first name (*FIRST = first
1 *			SYMDAT or holiday in
1 *			calendar)
1	GCLDPNM2	DS CL30	last name (*LAST = last SYMDAT

1 *			or holiday in calendar)
1 GCLDPDAT DS	CL10		date
1 GCLDPDT1 DS	CL10		start date (*UNCH = start-date
1 *			of SYMDAT is unchanged,
1 *			*FIRST = first date in
1 *			calendar)
1 GCLDPDT2 DS	CL10		end date (*UNCH = end date of
1 *			SYMDAT is unchanged, *UNDEF =
1 *			end date of SYMDAT is
1 *			undefined, *LAST = last date
1 *			in calendar)
1 GCLDPLM1 DS	CL10		first date in calendar (*UNCH
1 *			= calendar limit is
1 *			unchanged)
1 GCLDPLM2 DS	CL10		last date in calendar (*UNCH =
1 *			calendar limit is unchanged)
1 GCLDPTIM DS	CL8		time (*UNCH = time is
1 *			unchanged, *STD = standard
1 *			time)
1 GCLDPWT1 DS	CL5		beg. of working hours (*UNCH =
1 *			working hours are unchanged,
1 *			*STD = standard working hours)
1 GCLDPWT2 DS	CL5		end of working hours (*UNCH =
1 *			working hours are unchanged,
1 *			*STD = standard working hours)
1 GCLD#	EQU	*-GCLDHDR	
	END		

4.8 Macro syntax

4.8.1 Macro call format

The macro call format consists of two columns, the first of which contains the macro name while the second column contains the possible operands.

Macro name	Operands
<macroname>	<operand ₁ > ,<operand ₂ >

When issuing a macro you must separate the macro name from the first operand by at least one blank column. If you specify more than one operand, you must separate the operands with commas.

Certain characters (metacharacters) are used in the macro call format. These characters are explained in the tables below.

4.8.2 Elements of the metasyntax

Formal notation	Meaning	Examples
UPPERCASE LETTERS	Uppercase letters denote keywords or constants that the user must enter exactly as shown. Keywords must start with * if keywords and, as an alternative, the names of constants or variables can be specified.	DIB FORCED=*YES
lowercase letters	Lowercase letters denote the data type of values or variables that can be specified by the user.	DIB = <var: pointer>
< >	Angle brackets denote variables whose set of values is described by the data types indicated.	<var: pointer>
<u>Underscoring</u>	Underscoring indicates the default value for an operand. If an operand does not have a default value, an operand value must be specified.	FORCED = <u>*NO</u> / *YES
=	The equals sign links an operand name with the corresponding operand values.	DATA = <var: pointer>
/	The slash separates alternative operand values.	FORCED = <u>*NO</u> / *YES
list-poss(n)	A list can be created from the operand values that follow list-poss. n is the maximum number of elements in a list. If a list contains more than one element, it must be enclosed in parentheses.	FLAG=list-poss(3): *SLI / *SKIP / *DC Specify: FLAG=*SKIP FLAG=(*SLI,*DC)

An operand is assigned an operand value using an equals sign. The operand value is one of a set of possible values.

The set of possible operand values is determined by a data type. The table in section 4.8.3 contains the data types of the operand values.

4.8.3 Data types of the operand values

Data type	Character set	Comments
c-string	EBCDIC characters	must be enclosed in quotes
integer	[+-] 0..2147483647	is a decimal number
var:	Indicates the start of a variable specification. The variable type is indicated after the colon (see table in section 4.8.5)	<var:var-type>
reg:	Register 0..15	Specify: (<reg:var-type>)

4.8.4 Suffixes for data types

Suffix	Meaning
n..m	For the data type integer, n..m indicates a range of values; n: minimum value m: maximum value
	For the data type c-string, n..m is a length specification in bytes; n: minimum length m: maximum length where $n < m$
n	For the data type c-string, n is a length specification in bytes; n must be strictly adhered to.

The operand values can be specified directly as a string or integer (see data types “c-string” and “integer”) or indirectly by means of a variable (see data type “var:”). The table in section 4.8.5 lists the data types that are possible for variables.

4.8.5 Data types of variables

Data type	Description	Definition in program
char:n	The variable is a string with n characters. If the length specification is omitted, n=1 is used.	CLn
int:n	The variable is an integer that occupies n bytes. If n is not specified, n=1 is used. Prerequisite: $n \leq 4$	FLn
enum-of E:n	The variable is the enumeration E, which occupies n bytes. If the length specification is omitted, n=1 is used. ($n \leq 4$)	XLn
pointer	The variable is an address or an address value.	A

4.8.6 MF operand

The MF operand determines the type of macro generation.

Based on the operand values for MF, a distinction is made between five types (forms) of macro call:

Macro name	Operands
<macroname>	MF=L [,operand ₁ ,...,operand _n]
	MF=M ,operand ₁ ,...,operand _n [,PREFIX=p][,MACID=mac]
	MF=D [,PREFIX=p]
	MF=C [,PREFIX=p][,MACID=mac]
	MF=E [,PARAM=<adr> / (<reg>)]

MF=C [,PREFIX=p][,MACID=mac]

Only the data area is generated. Operand values are not evaluated. Each field has a field name and explanatory equates, if required. The data area ends with a length equate. Generally, the standard header (see page 84) must be initialized by the user. The PREFIX operand enables the user to define the first character of the field names and equates. p = 1 letter.

The MACID operand enables the user to define the second to fourth characters (inclusive) of the field names and equates. mac = 1..3 characters.

All other operands are not evaluated in the C form; this does not apply to VERSION = 1 and XPAND.

MF=D [,PREFIX=p]

A DSECT is generated. Each field has a field name and explanatory equates, if required. The DSECT ends with a length equate. There is no switchover to the initial address level.

The **DSECT** describes the structure of a storage area without occupying any memory itself. The symbolic name specified for DSECT is entered in an ESD record. The address level is set to zero.

The p prefix enables the user to define the first character of the field names and equates. p = 1 letter.

XPAND is evaluated and defines which DSECT is created.

MF=E [,PARAM=<addr> / (<reg>)]

Only the commands required to call the function module are generated. If CALLER=USER is specified, the command part ends with an SVC. In the macro call, the address of the data area must be specified using the operand values.

The PARAM operand identifies the address of the data area. addr = address of the data area.

reg = register containing the address of the data area. Before the macro is called, the register must be loaded with this address value.

If PARAM is not specified, register 1 must be supplied with the address of the parameter list before a macro call.

All other operands are not evaluated in the E form.

MF=L [,operand 1, ..., operand n] [PREFIX=p]

Only the data area is generated, taking into account the operand values specified in the macro call. The data area does not contain any field names or any explanatory equates. The standard header (see page 84) is initialized. The macro call is issued in the definition part of the program.

With shared code programming, this call must not be located in the invariable part of the program if it contains variable data. The data area is initialized with constants in the invariable part of the program, copied to a local data area before the E form call, and modified there if necessary. The modification is implemented with the M form, for example, if it is offered for the relevant interface.

MF=M ,operand 1,...,operand n [,PREFIX=p][,MACID=mac]

Commands (e.g. MVCs) are generated which overwrite fields with the operand values specified in the macro call during the program run in a data area already initialized with MF=L, or with shared code programming in a local copy of the data area initialized with MF=L. The M form thus offers a simple option for **dynamically** adapting the operand values with which a macro is called to suit the program run.

If MF=M is specified, no default values are used for function operands, i.e. all operands must be specified explicitly.

As the commands generated with MF=M use the symbolic addresses and equates of the C or D form, you must ensure when using the M form that these names are available for addressing the data area to be modified. In particular, you must make sure that in a macro call with MF=M the PREFIX and MACID operands are specified with the same values as in the associated MF=C or MF=D call.

The PREFIX operand enables the user to define the first character of the field names and equates. p = 1 letter.

The MACID operand enables the user to define the second to fourth characters (inclusive) of the field names and equates. mac = 1..3 characters.

4.9 C header file (calendr.h)

Descriptions in double angle brackets (<<...>>) are inserted comments or explanations.

```

/* main return codes */
/* _mret_code */
#define CALENDRsuccessful 0 /* no error */
#define CALENDRinternal_error 1 /* parameter error */
#define CALENDRcalendar_existing 2 /* calendar already exists */
#define CALENDRcalendar_not_existing 3 /* calendar does not exist */
#define CALENDRDMS_error_calendar 4 /* DMS error on calendar file */
#define CALENDRno_calendar 5 /* file is not a calendar */
#define CALENDRcalendar_inconsistent 6 /* calendar is inconsistent */
#define CALENDRno_update_possible 7 /* update of calendar not
/* possible at the moment */
#define CALENDRholiday_file_not_existing 8
/* holiday file does not
/* exist
#define CALENDRDMS_error_holiday_file 9
/* DMS error on holiday file */
#define CALENDRtoo_many_days 10 /* too many days between
/* calendar limits
#define CALENDREnddate_less_startdate 11
/* end date before start date */
#define CALENDRno_memory 12 /* no more memory available */
#define CALENDRcontrol_block_invalid 13
/* control block invalid */
#define CALENDRdate_outside 14 /* date outside of calendar
/* limits
#define CALENDRSYMDAT_existing 15 /* SYMDAT already exists */
#define CALENDRSYMDAT_not_existing 16 /* SYMDAT does not exist */
#define CALENDRcyclic_SYMDAT 17 /* SYMDAT is a cyclic SYMDAT */
#define CALENDRnon_cyclic_SYMDAT 18 /* SYMDAT is a non-cyclic
/* SYMDAT
#define CALENDRassignement_existing 19 /* assignment to date
/* already exists
#define CALENDRassignement_not_existing 20
/* assignment to date does
/* not exist
#define CALENDRdate_invalid 21 /* invalid date */
#define CALENDRholiday_existing 22 /* holiday already exists */
#define CALENDRholiday_not_existing 23 /* holiday does not exist */
#define CALENDRcyclic_holiday 24 /* holiday is a cyclic
/* holiday

```

```
#define CALENDRnon_cyclic_holiday 25 /* holiday is a non-cyclic */
/* holiday */
#define CALENDRholiday_activated 26 /* holiday already activated */
#define CALENDRholiday_not_activated 27 /* holiday is not activated */
#define CALENDRno_information 28 /* output area too small: no */
/* information available */
#define CALENDRsome_information 29 /* output area too small: some*/
/* information available */
#define CALENDRwrong_SYMDAT_area 30 /* definition of SYMDAT area */
/* invalid */
#define CALENDRwrong_holiday_area 31 /* definition of holiday area */
/* invalid */
#define CALENDRstandardweek_incorrect 32 /* area for standard week is */
/* incorrect */
#define CALENDRholiday_file_incorrect 33 /* entry of holiday file is */
/* incorrect */
#define CALENDRinvalid_limits 34 /* LIMIT1 < 1970 or LIMIT2 > */
/* 2030 or new LIMIT1 > old */
/* LIMIT2 */
#define CALENDRstart_date_outside 35 /* start date of cyclic */
/* SYMDAT outside of */
/* calendar limits (warning) */
#define CALENDRfunction_not_allowed 36 /* function not allowed with */
/* OPENMOD = *READ or */
/* *READALL */
#define CALENDRSYMDAT_start_date_deleted 37 /* start date for cyclic */
/* SYMDAT with */
/* CYCLTYP=WORKDAY has been */
/* deleted as it is outside */
/* calendar limits */
#define CALENDRprogram_error 38 /* internal error in program */
#define CALENDRcalendar_version_not_supported 39 /* version of calendar is */
/* not supported by version */
/* of interface */
#define CALENDRtoo_many_symdat 40 /* more than 4096 SYMDATs have*/
/* been created */
#define CALENDRtoo_many_holiday 41 /* more than 1024 holidays */
/* have been created */
#define CALENDRtoo_many_assigned_days 42 /* more than 1860 days */
/* assigned to a holiday */
#define CALENDRlinkage_error_function 65535 /* function not supported */
```

```
#define CALENDRLinkage_error_version 65535
/* version not supported */
#define CALENDRLinkage_error_alignment 65535
/* alignment error */

/* function codes */
/* ENUM _funct_s */
#define CALENDRCreacal 1 /* create calendar */
#define CALENDROpenacal 2 /* open calendar */
#define CALENDRClseacal 3 /* close calendar */
#define CALENDRSavecal 4 /* save calendar */
#define CALENDRModbas 5 /* modify basic information */
#define CALENDRModday 6 /* modify day */
#define CALENDRCreNSym 7 /* create non-cyclic symdat */
#define CALENDRCreCSym 8 /* create cyclic symdat */
#define CALENDRdelSym 9 /* delete symdat */
#define CALENDRModNSym 10 /* modify non-cyclic symdat */
#define CALENDRModCSym 11 /* modify cyclic symdat */
#define CALENDRCrenhol 12 /* create non-cyclic holiday */
#define CALENDRCrecho1 13 /* create cyclic holiday */
#define CALENDRdelhol 14 /* delete holiday */
#define CALENDRModnhol 15 /* modify non-cyclic holiday */
#define CALENDRModcho1 16 /* modify cyclic holiday */
#define CALENDRAkthol 17 /* activate holiday */
#define CALENDRdeakhol 18 /* deactivate holiday */
#define CALENDRshbas 19 /* show basic information */
#define CALENDRshloday 20 /* show list of days */
#define CALENDRshdayin 21 /* show day information */
#define CALENDRshdayh1 22 /* show day with holidays */
#define CALENDRshlosym 23 /* show list of symdats */
#define CALENDRshsymin 24 /* show symdat information */
#define CALENDRshlohol 25 /* show list of holidays */
#define CALENDRshholin 26 /* show holiday information */
#define CALENDRshnestm 27 /* show next symdat time */

/* open modus */
/* ENUM _open_s */
#define CALENDRread 1 /* read */
#define CALENDRreadall 2 /* read complete calendar */
#define CALENDRupdate 3 /* update */

/* attribute of day */
/* ENUM _attribute_s */
#define CALENDRunchanged 0 /* unchanged */
#define CALENDRfree_day 1 /* free day */
```

```

#define CALENDRwork_day 2          /* workday          */
#define CALENDRstd_week 3         /* standard week    */

/* alternative                      */
/* ENUM _alternative_s              */
#define CALENDRunchanged 0        /* unchanged        */
#define CALENDRbefore 1          /* insert before free day */
#define CALENDRafter 2           /* insert after free day */
#define CALENDRskip 3            /* don't insert     */
#define CALENDRon 4              /* insert on free day */

/* kind of cycle                    */
/* ENUM _cycle_s                    */
#define CALENDRunchanged 0        /* unchanged        */
#define CALENDRday 1             /* cycle = day      */
#define CALENDRworkday 2         /* cycle = workday  */
#define CALENDRweek 3           /* cycle = week     */
#define CALENDRmonth 4          /* cycle = month    */

/* save                              */
/* ENUM _save_s                     */
#define CALENDRyes 1             /* save calendar    */
#define CALENDRno 2             /* don't save calendar */

/* sort                              */
/* ENUM _sort_s                      */
#define CALENDRsymdat 1          /* sort by symdat   */
#define CALENDRtime 2           /* sort by time     */

/* parameterarea description        */
struct CALENDR_md1 {
    /* Standardheader                */
    struct ESMFHDR hdr;
    unsigned char function_code;    /* funct =          */
    unsigned char open_modus;      /* openmod =        */
    unsigned char attribute;       /* attribute =      */
    unsigned char cycle_typ;       /* cycltyp =       */
    unsigned char alternative;     /* alternative =    */
    unsigned char save;            /* save =          */
    unsigned char sort;           /* sort =          */

    /* indicator                      */
    struct /* flag_1 */ {
        char reserved_1: 8;        /* reserved        */
    } flag_1;
};

```

```
/* indicator */
struct /* flag_2 */ {
    char reserved_2: 8; /* reserved */
} flag_2;

char filler1[3]; /* filler */
long number; /* number of days or SYMDATs */
unsigned long cycle_value; /* number of days between two
/* entries */
unsigned long DMS_returncode; /* DMS return code */
void* control_block; /* address of control block */
void* week_address; /* address of standard week */
void* days_to_add; /* address of days to add */
unsigned long days_to_add_num; /* number of days in area */
void* days_to_del; /* address of days to delete */
unsigned long days_to_del_num; /* number of days in area */
void* output_area; /* address of output area */
unsigned long output_area_len; /* length of output area */
char calendar_name[54]; /* calendar name */
char name[30]; /* name of SYMDAT or holiday */
char name1[30]; /* first name (*FIRST = first
/* SYMDAT or holiday in
/* calendar)
char name2[30]; /* last name (*LAST = last
/* SYMDAT or holiday in
/* calendar)
char date[10]; /* date */
char date1[10]; /* start date (*UNCH =
/* start date of SYMDAT is
/* unchanged, *FIRST = first
/* date in calendar)
char date2[10]; /* end date (*UNCH = end date
/* of SYMDAT is unchanged,
/* *UNDEF = end date of
/* SYMDAT is undefined, *LAST
/* = last date in calendar)
char calendar_limit1[10]; /* first date in calendar
/* (*UNCH = calendar limit is
/* unchanged)
char calendar_limit2[10]; /* last date in calendar
/* (*UNCH = calendar limit is
/* unchanged)
char time[8]; /* time (*UNCH = time is
/* unchanged, *STD = standard
/* time)
char work_time1[5]; /* beginning of working hours
/* (*UNCH = working hours are */
```



```

/* unchanged, *STD = standard */
/* working hours) */
char work_time2[5]; /* end of working hours (*UNCH*/
/* = working hours are */
/* unchanged, *STD = standard */
/* working hours) */

};

/* User interface */
#ifdef __SNI_HOST_BS2000
void _SVC(int, void*);
#define GCLDSTU(p) _SVC(193, &p)
#endif

/* System interface */
#ifdef __SNI_HOST_BS2000
#pragma ISL GCLDSTP
void GCLDSTP(struct CALENDR_md1*);
#else
void GCLDSTP(struct CALENDR_md1*);
#endif

/* Standard-week-entry */
struct CALENDR_stdw_entry_md1 {
    << entry of the standard working week; contains the names of the weekdays, the >>
    << attribute for each weekday, and the beginning and end of the standard working hours >>
    char swe_day[3]; /* weekday (MON - SUN) */
    char swe_attribute[1]; /* attribute of day */
    /* (W=workday, F=free day) */
    char swe_time1[5]; /* beginning of working hours */
    char swe_time2[5]; /* end of working hours */
};

/* Date-entry */
struct CALENDR_date_entry_md1 {
    << date entry; contains a four-character year specification, a two-character month >>
    << specification, and a two-character day specification >>
    char dte_year[4]; /* year */
    char dte_hyphen1[1]; /* hyphen */
    char dte_month[2]; /* month */
    char dte_hyphen2[1]; /* hyphen */
    char dte_day[2]; /* day */
};

```

```

/* Day-entry                                                    */
struct CALENDAR_day_entry_md1 {
    << calendar day entry; contains the date, the weekday name, the attribute, the >>
    << attribute for this weekday in the standard working week, the holiday definition, >>
    << the beginning and end of the working hours, the number of assigned holidays >>
    << and SYMDATs >>

    char dye_date[10];                /* date                                */
    char dye_day[3];                  /* weekday (MON - SUN)                 */
    char dye_attribute[1];            /* attribute (W=workday,               */
    /* F=free day)                      */
    char dye_attr_def[1];             /* attribute defined for day           */
    /* (W=workday, F=free day,          */
    /* S=standard)                      */
    char dye_attr_std[1];            /* attribute of day in                 */
    /* standard week (W=workday,        */
    /* F=free day)                      */
    char dye_holiday[1];             /* holiday (Y=yes, N=no)               */
    char dye_time1[5];                /* beginning of working hours          */
    char dye_time2[5];                /* end of working hours                */
    char dye_filler[1];              /* filler                               */
    unsigned short dye_assigned_holidays;
    /* number of assigned                */
    /* holidays                          */
    unsigned short dye_assigned_SYMDAT;
    /* number of assigned SYMDATs       */
};

/* SYMDAT-entry                                                */
struct CALENDAR_SYMDAT_entry_md1 {
    << SYMDAT entry; contains the name, type, time, cycle type, cycle value, alternative >>
    << for free days, start and end date for cyclic SYMDATs, as well as the number of >>
    << assigned calendar days >>

    char sye_SYMDAT_name[20];         /* SYMDAT name                         */
    char sye_SYMDAT_type[1];          /* SYMDAT type (C=cyclic,              */
    /* N=non-cyclic)                    */
    char sye_SYMDAT_time[8];          /* time                                  */
    char sye_cycle_type[1];           /* kind of cycle (D=day,               */
    /* W=week, 0=workday,                */
    /* M=month)                          */
    unsigned short sye_cycle_value;    /* cycle value                          */
    char sye_alternative[1];          /* alternative for free days            */
    /* (B=before, A=after,               */
    /* S=skip, 0=on)                     */
    char sye_filler1[1];              /* filler                               */
};

```

```

        char sye_date1[10];                /* start date for cyclic      */
                                           /* SYMDAT                      */
        char sye_date2[10];                /* end date for cyclic SYMDAT */
        unsigned short sye_assigned_days;  /* number of assigned days    */
};

/* SYMDAT_short-entry */
struct CALENDAR_SYMDAT_short_entry_md1 {
<< short SYMDAT entry; contains the name and assigned time >>
        char sse_SYMDAT_name[20];          /* SYMDAT name                */
        char sse_SYMDAT_time[8];          /* time                        */
};

/* holiday-entry */
struct CALENDAR_holiday_entry_md1 {
<< holiday entry; contains the name, holiday type, activation status, and number of >>
<< assigned days >>
        char hle_holiday_name[30];        /* holiday name                */
        char hle_holiday_type[1];         /* holiday type (C=cyclic,    */
                                           /* N=non-cyclic)              */
        char hle_assigned[1];             /* holiday is assigned         */
                                           /* (Y=yes, N=no)               */
        unsigned short hle_assigned_days; /* number of assigned days    */
};

/* Standard-week */
struct CALENDAR_stdweek_md1 {
<< standard working week (short entry); contains the area for the standard working >>
<< week >>
        char stdweek_area[7][14];         /* area for standard week     */
                                           /* entry is described with    */
                                           /* XPAND=ENTSTDW              */
};

/* control-block */
struct CALENDAR_ccb_md1 {
<< control block; contains the address of an internal data area set up when creating >>
<< a calendar >>
        char cb_text[4];                  /* text                        */
        void* cb_address;                 /* address                     */
        unsigned long cb_check_pattern;    /* check pattern               */
};

```

```

};

/* day-list */
struct CALENR_daylist_md1 {
<< list of calendar days (short entry); contains the area of the calendar day list >>
    char pdy_day_area[1860][10]; /* area for days entry is */
                                /* described with */
                                /* XPAND=ENTDATE */
};

/* Basic-Information */
struct CALENR_out_basinf_md1 {
<< entry for basic information; contains the interface version, the number of calendar >>
<< days, the SYMDATs and holidays in the calendar, the calendar name, the first >>
<< and last date entry in the calendar, the first and last SYMDAT in the calendar, >>
<< the first and last holiday in the calendar, and the area of the standard working >>
<< week >>
    unsigned short bi_version; /* version of interface */
    unsigned short bi_number_day; /* number of days in calendar */
    unsigned short bi_number_SYMDAT; /* number of SYMDATs in */
                                    /* calendar */
    unsigned short bi_number_holiday; /* number of holidays in */
                                        /* calendar */
    char bi_calendar_name[54]; /* calendar name */
    char bi_limit1[10]; /* first date in calendar */
    char bi_limit2[10]; /* last date in calendar */
    char bi_first_SYMDAT[20]; /* first SYMDAT in calendar */
    char bi_last_SYMDAT[20]; /* last SYMDAT in calendar */
    char bi_first_holiday[30]; /* first holiday in calendar */
    char bi_last_holiday[30]; /* last holiday in calendar */
    char bi_stdweek_area[7][14]; /* area for standard week */
                                    /* entry is described with */
                                    /* XPAND=ENTSTDW */
};

/* List-of-days */
struct CALENR_out_loday_md1 {
<< entry for a list of calendar days; contains the interface version, the total number of >>
<< calendar days, the selected days, the days output, and the area of the calendar day >>
<< list >>

```

```

    unsigned short lod_version;      /* version of interface      */
    unsigned short lod_available_days;
                                    /* number of days in calendar */
    unsigned short lod_selected_days;
                                    /* number of selected days   */
    unsigned short lod_given_days;   /* number of given days      */
    char lod_day_area[1860][32];     /* area for days entry is    */
                                    /* described with             */
                                    /* XPAND=ENTDAY              */
};

/* day_information */
struct CALENDR_out_dayinf_md1 {
    << entry for calendar day with assigned SYMDATs; contains the interface version, >>
    << the number of assigned and selected SYMDATs, the number of SYMDATs output, >>
    << the date, the weekday name, the attribute for that day and for the corresponding >>
    << weekday in the standard working week, the holiday definition, the beginning and >>
    << end of working hours, and the area for SYMDAT entries >>

    unsigned short di_version;      /* version of interface      */
    unsigned short di_assigned_SYMDAT;
                                    /* number of assigned SYMDATs */
    unsigned short di_selected_SYMDAT;
                                    /* number of selected SYMDATs */
    unsigned short di_given_SYMDAT; /* number of given SYMDATs  */
    char di_date[10];              /* date                      */
    char di_day[3];                /* weekday (MON - SUN)      */
    char di_attribute[1];          /* attribute (W=workday,    */
                                    /* F=free day)              */
    char di_attr_def[1];           /* attribute defined for day */
                                    /* (W=workday, F=free day,  */
                                    /* S=standard)              */
    char di_attr_std[1];           /* attribute of day in      */
                                    /* standard week (W=workday, */
                                    /* F=free day)              */
    char di_holiday[1];           /* holiday (Y=yes, N=no)    */
    char di_time1[5];              /* beginning of working hours */
    char di_time2[5];              /* end of working hours     */
    char di_filler1[1];           /* filler                    */
    char di_SYMDAT_area[4096][28]; /* area for SYMDAT entry is */
                                    /* described with             */
                                    /* XPAND=ENTSYMS            */
};

/* day_holiday */
struct CALENDR_out_dayhol_md1 {

```

<< entry for calendar day with assigned holidays; contains the interface version, the >>
 << number of assigned holidays, the number of holidays output, the date and the area >>
 << for a holiday entry >>

```

    unsigned short dh_version;          /* version of interface      */
    unsigned short dh_assigned_holiday; /* number of assigned       */
                                        /* holidays                  */
    unsigned short dh_given_holiday;   /* number of given holidays */

    char dh_date[10];                  /* date                      */
    char dh_holiday_area[1024][34];   /* area for holidays entry is */
                                        /* described with           */
                                        /* XPAND=ENTHOL            */
};

```

```

/* List-of-SYMDAT                                     */
struct CALENDAR_out_losym_md1 {

```

<< entry for SYMDAT list; contains the interface version, the number of all SYMDATs, >>
 << the number of selected SYMDATs, the number of SYMDATs output, and the area >>
 << for the SYMDAT list >>

```

    unsigned short los_version;        /* version of interface      */
    unsigned short los_available_SYMDAT; /* number of SYMDATs in     */
                                        /* calendar                  */
    unsigned short los_selected_SYMDAT; /* number of selected SYMDATs */
    unsigned short los_given_SYMDAT;   /* number of given SYMDATs  */

    char los_SYMDAT_AREAC[4096][56];   /* area for SYMDAT entry is */
                                        /* described with           */
                                        /* XPAND=ENTSYM            */
};

```

```

/* SYMDAT_information                                 */
struct CALENDAR_out_syminf_md1 {

```

<< entry for SYMDAT with assigned calendar days; contains the interface version, the >>
 << number of assigned and selected calendar days, the number of calendar days output >>
 << the SYMDAT name, the type, the assigned time, the cycle type, the cycle value, >>
 << the alternative for free days, the start and end dates for a cyclic SYMDAT, and the >>
 << area for the entry of the assigned calendar days >>

```

    unsigned short si_version;         /* version of interface      */
    unsigned short si_assigned_days;   /* number of assigned days   */

```

```

unsigned short si_selected_days;
/* number of selected days */
unsigned short si_given_days; /* number of given days */
char si_SYMDAT_name[20]; /* SYMDAT name */
char si_SYMDAT_type[1]; /* SYMDAT type (C=cyclic,
/* N=non-cyclic) */
char si_SYMDAT_time[8]; /* time */
char si_cycle_type[1]; /* kind of cycle (D=day,
/* W=week, 0=workday,
/* M=month) */

unsigned short si_cycle_value; /* cycle value */
char si_alternative[1]; /* alternative for free days
/* (B=before, A=after,
/* S=skip, 0=on) */

char si_date1[10]; /* start date for cyclic
/* SYMDAT */
char si_date2[10]; /* end date for cyclic SYMDAT */
char si_filler1[1]; /* filler1 */
char si_day_area[1860][10]; /* area for assigned days
/* entry is described with
/* XPAND=ENTDATE */
};

/* List-of-holidays */
struct CALENDR_out_lohol_md1 {
<< entry for list of holidays; contains the interface version, the number of all holidays, >>
<< the number of selected holidays, the number of holidays output, and the area of >>
<< the holiday list >>
    unsigned short loh_version; /* version of interface */
    unsigned short loh_available_holiday; /* number of holidays in
/* calendar */
    unsigned short loh_selected_holiday; /* number of selected
/* holidays */
    unsigned short loh_given_holiday; /* number of given holidays */
    char loh_holiday_area[1024][34]; /* area for holidays entry is
/* described with
/* XPAND=ENTHOL */
};

/* holiday-information */
struct CALENDR_out_holinf_md1 {

```

<< entry for holiday with assigned calendar days; contains the interface version, the >>
 << number of assigned and selected calendar days, the number of calendar days >>
 << output, the holiday name, the type, the activation status, and the area for the entry >>
 << of the assigned calendar days >>

```

    unsigned short hi_version;          /* version of interface      */
    unsigned short hi_assigned_days;    /* number of assigned days   */
    unsigned short hi_selected_days;    /* number of selected days   */
    unsigned short hi_given_days;       /* number of given days      */
    char hi_holiday_name[30];          /* holiday name              */
    char hi_holiday_type[1];           /* holiday type (C=cyclic,   */
                                        /* N=non-cyclic)            */
    char hi_assigned[1];                /* holiday is assigned       */
                                        /* (Y=yes, N=no)            */
    char hi_day_area[1860][10];        /* area for assigned days    */
                                        /* entry is described with   */
                                        /* XPAND=ENTDATE            */
};

/* Next-SYMDAT-time */
struct CALENDR_out_nestm_md1 {

```

<<Entry for the date and time most closely following a specified >>
 <<time in the considered set of SYMDATs and the name of the >>
 <<relevant SYMDAT. >>

```

    unsigned short nst_version;         /* interface version        */
    char nst_date[10];                 /* next date                */
    char nst_time[8];                 /* next time                */
    char nst_name[20];                 /* SYMDAT name             */
};

#endif          /* _CALENDR_H */

```

5 Command interface

SHOW-CALENDAR

Request information from calendar file

Application area: UTILITIES
Privileged status: STD-PROCESSING

Function

The SHOW-CALENDAR command is used to obtain information on the calendar data in the specified calendar file.

By default (`SELECT=*TODAY`), the following information is output for the current day: the name of the calendar file, the current date, the name of the weekday (Monday...Sunday), the attribute of the current day (free day / working day), the number of associated SYMDATs (symbolic dates), the working hours, and if appropriate the holiday name, the names of the associated SYMDATs, and the time specifications assigned to them.

You use `SELECT=*BASIC-INFORMATION` to output the basic information of the calendar file: the name of the calendar, the calendar limits, and definitions of the standard working week. Within the standard working week, the attributes and working hours for the weekdays are defined.

You use `SELECT=*DATE` to output the day information relating to one or more days: the date, the name of the weekday, the attribute, the number of assigned SYMDATs, the working hours, the holiday name if appropriate, and (using a special operand) the names of the associated SYMDATs and the assigned time specifications.

You use `SELECT=*SYMBOLIC-DATE` to output information on SYMDATs. These are symbolic dates under whose names calendar days are combined (see description on page 14ff).

You use `SELECT=*HOLIDAY` to output information on holidays. Output can be routed to `SYSOUT` or `SYSLST`. This command supports structured output in S variables (see description in the manual “Commands, Volume 4, Output in S Variables” [1]).

Format

SHOW-CALENDAR

CALENDAR-NAME = <full-filename 1..54>

,SELECT = *TODAY / *BASIC-INFORMATION / *DATE(...) / *SYMBOLIC-DATE(...) / *HOLIDAY(...)

*DATE(...)

FROM = *TODAY / *FIRST-CALENDAR-DATE / <date 8..10>

,TO = *SAME / *TODAY / *LAST-CALENDAR-DATE / <date 8..10> / *BY-NUMBER-OF-DAYS(...)

 *BY-NUMBER-OF-DAYS(...)

NUMBER-OF-DAYS = <integer 1..1827>

,ASSIGNED-SYM-DATE = *NONE / *ALL(...) /

 <full-filename 1..20 without-cat-user-gen-vers with-wild>(...)

 *ALL(...)

ORDER-WITHIN-DAY = *BY-TIME / *BY-SYMBOLIC-DATE

 <full-filename 1..20 without-cat-user-gen-vers with-wild>(...)

ORDER-WITHIN-DAY = *BY-TIME / *BY-SYMBOLIC-DATE

*SYMBOLIC-DATE(...)

FROM = *FIRST-SYMBOLIC-DATE / <full-filename 1..20 without-cat-user-gen-vers with-wild>

,TO = *SAME / *LAST-SYMBOLIC-DATE /

 <full-filename 1..20 without-cat-user-gen-vers with-wild> /

 *BY-NUMBER-OF-SYMBOLIC-DATES(...)

 *BY-NUMBER-OF-SYMBOLIC-DATES(...)

NUMBER-OF-SYM-DATES = <integer 1..4096>

,ASSIGNED-DATES = *NO / *ALL / *NEXT-DATE / *INTERVAL(...)

 *INTERVAL(...)

FROM = *TODAY / *FIRST-ASSIGNED-DATE / <date 8..10>

,TO = *SAME / *TODAY / *LAST-ASSIGNED-DATE / <date 8..10> /

 *BY-NUMBER-OF-DAYS(...)

 *BY-NUMBER-OF-DAYS(...)

NUMBER-OF-DAYS = <integer 1..1827>

continued ➔

```

*HOLIDAY(...)
  |
  | FROM = *FIRST-HOLIDAY / <full-filename 1..30 without-cat-user-gen-vers with-wild>
  |
  | ,TO = *SAME / <full-filename 1..30 without-cat-user-gen-vers with-wild> /
  |   *LAST-HOLIDAY / *BY-NUMBER-OF-HOLIDAYS(...)
  |
  |   *BY-NUMBER-OF-HOLIDAYS(...)
  |     |
  |     | NUMBER-OF-HOLIDAYS = <integer 1..1024>
  |
  | ,ASSIGNED-DATES = *NQ / *YES
  |
,OUTPUT = *SYSOUT / list-poss(2): *SYSOUT / *SYSLST(...)
  |
  | *SYSLST(...)
  |   |
  |   | SYSLST-NUMBER = *STD / <integer 1..99>
  |

```

Operands

CALENDAR-NAME = <full-filename 1..54>

Name of the calendar file from which information is to be output.

SELECT = ***TODAY** / ***BASIC-INFORMATION** / ***DATE(...)** / ***SYMBOLIC-DATE(...)** / ***HOLIDAY(...)**

Specification indicating which information is to be output from the calendar file.

SELECT = ***TODAY**

Specifies the information for the current day:

- the name of the calendar file
- the current date
- the name of the weekday
- the attribute: W = workday or F = free day
- the number of associated SYMDATs
- the working hours (beginning and end)
- if appropriate, the name of the holiday
- a list containing the names of associated SYMDATs and the time specifications assigned to them (sorted by time)

SELECT = ***BASIC-INFORMATION**

Specifies the basic information of the calendar:

- the name of the calendar file
- the calendar limits
- the days of the standard working week and their attributes (working day / free day)

SELECT = *DATE(...)

Specifies the day information for a range of days. The desired range is selected using the subordinate operands FROM and TO. The following information is output for each selected day:

- the date
- the name of the weekday
- the attribute: W = workday or F = free day
- the number of associated SYMDATs
- the working hours (beginning and end)
- if appropriate, the name of the holiday

You can also use the subordinate operand ASSIGNED-SYM-DATE to request a list of the names of associated SYMDATs and the times assigned to them.

FROM = *TODAY / *FIRST-CALENDAR-DATE / <date 8..10>

Determines the first day at which output of the information is to begin.

FROM = *TODAY

Output begins from the current day.

FROM = *FIRST-CALENDAR-DATE

Output begins with the first day in the calendar file.

FROM = <date 8..10>

Output begins with the specified date.

TO = *SAME / *TODAY / *LAST-CALENDAR-DATE / <date 8..10> / *BY-NUMBER-OF-DAYS(...)

Determines the last day at which output of the information is to end.

TO = *SAME

Information is output for the day specified in FROM only.

TO = *TODAY

Output ends with the current day.

TO = *LAST-CALENDAR-DATE

Output ends with the last day in the calendar file.

TO = *BY-NUMBER-OF-DAYS(...)

Determines the size of the desired range in days, beginning with the day specified in FROM.

NUMBER-OF-DAYS = <integer 1..1827>

Number of days.

ASSIGNED-SYM-DATE = *NONE / *ALL(...) /

<full-filename 1..20 without-cat-user-gen-vers with-wild>(…)

Specifies whether the associated SYMDATs are to be output in addition to the calendar days.

The default value is *NONE, i.e. no additional list is output.

ASSIGNED-SYM-DATE = *ALL(...)

All the names of the SYMDATs assigned to the specified calendar days are output.

ORDER-WITHIN-DAY = *BY-TIME / *BY-SYMBOLIC-DATE

Determines the sort criterion. Output is sorted in accordance with the assigned time (*BY-TIME; default) or alphabetically by SYMDAT name (*BY-SYMBOLIC-DATE).

ASSIGNED-SYM-DATE =

<full-filename 1..20 without-cat-user-gen-vers with-wild>(…)

The specified SYMDAT or the group of SYMDATs identified in the wildcard string is also output. (You can select several SYMDATs with a single specification using a wildcard string, where only the wildcard character * is permitted at the end for any character string.)

ORDER-WITHIN-DAY = *BY-TIME / *BY-SYMBOLIC-DATE

Determines the sort criterion. Output is sorted in accordance with the assigned time (*BY-TIME; default) or alphabetically by SYMDAT name (*BY-SYMBOLIC-DATE).

SELECT = *SYMBOLIC-DATE(...)

Specifies information on the SYMDATs defined in the calendar. The range of output is defined using the subordinate operands FROM and TO. You can also output the assigned days using the subordinate operand ASSIGNED-DATES.

FROM = *FIRST-SYMBOLIC-DATE /

<full-filename 1..20 without-cat-user-gen-vers with-wild>

Specifies the SYMDAT at which output is to begin.

FROM = *FIRST-SYMBOLIC-DATE

Output begins with the first SYMDAT in the alphabetical sequence.

FROM = <full-filename 1..20 without-cat-user-gen-vers with-wild>

Output begins with the specified SYMDAT or with the group of SYMDATs (in alphabetical order) identified in the wildcard string.

(You can select several SYMDATs with a single wildcard string, where only the wildcard character * is permitted at the end for any character string.)

TO = *SAME / *LAST-SYMBOLIC-DATE /
<full-filename 1..20 without-cat-user-gen-vers with-wild> /
***BY-NUMBER-OF-SYMBOLIC-DATES(...)**
 Specifies the SYMDAT at which output is to end.

TO = *SAME
 The specification entered for the FROM operand also applies here.

TO = *LAST-SYMBOLIC-DATE
 Output ends with the last SYMDAT in the alphabetical sequence.

TO = <full-filename 1..20 without-cat-user-gen-vers with-wild>
 Output ends with the specified SYMDAT or with the group of SYMDATs (in alphabetical order) identified in the wildcard string.
 (You can select several SYMDATs with a single wildcard string, where only the wildcard character * is permitted at the end for any character string.)

TO = *BY-NUMBER-OF-SYMBOLIC-DATES(...)
 Determines the number of SYMDATs to be output, beginning with the SYMDAT specified in FROM.

NUMBER-OF-SYM-DATES = <integer 1..4096>
 Number of SYMDATs.

ASSIGNED-DATES = *NO / *ALL / *NEXT-DATE / *INTERVAL(...)
 Specifies whether the calendar days assigned to a selected SYMDAT are also to be output.
 The default value is *NO, i.e. no additional list is output.

ASSIGNED-DATES = *ALL
 All calendar days assigned to the selected SYMDAT are also output.

ASSIGNED-DATES = *NEXT-DATE
 All calendar days assigned to the selected SYMDAT which occur after the current date are also output.

ASSIGNED-DATES = *INTERVAL(...)
 Defines a range of days to be listed.

FROM = *TODAY / *FIRST-ASSIGNED-DATE / <date 8..10>
 First day in the range of days to be listed.

FROM = *TODAY
 The range begins with the current day.

FROM = *FIRST-ASSIGNED-DATE
 The range begins with the next assigned day.

FROM = <date 8..10>
 The range begins with the specified day.

TO = *SAME / *TODAY / *LAST-ASSIGNED-DATE / <date 8..10> / *BY-NUMBER-OF-DAYS(...)

Last day in the range of days to be listed.

TO = *SAME

The specification entered for the FROM operand also applies here.

TO = *TODAY

The range ends with the current day.

TO = *LAST-ASSIGNED-DATE

The range ends with the last assigned day.

TO = <date 8..10>

The range ends with the specified day.

TO = *BY-NUMBER-OF-DAYS(...)

Determines the number of days to be listed, beginning with the day specified in FROM.

NUMBER-OF-DAYS = <integer 1..1827>

Number of days.

SELECT = *HOLIDAY(...)

Specifies information on holidays. This includes the names of the holidays and their properties (type: cyclic / non-cyclic; activation status: yes / no). The range of output is defined using the subordinate operands FROM and TO. You can also request a list of assigned days using the subordinate operand ASSIGNED-DATES.

FROM = *FIRST-HOLIDAY /

<full-filename 1..30 without-cat-user-gen-vers with-wild>

Specifies the holiday at which output is to begin.

FROM = *FIRST-HOLIDAY

Output begins with the first holiday in the calendar file.

FROM = <full-filename 1..30 without-cat-user-gen-vers with-wild>

Output begins with the specified holiday or with the group of holidays (in alphabetical order) identified in the wildcard string.

(You can select several holidays with a single wildcard string, where only the wildcard character * is permitted at the end for any character string.)

TO = *SAME / <full-filename 1..30 without-cat-user-gen-vers with-wild> / *LAST-HOLIDAY / *BY-NUMBER-OF-HOLIDAYS(...)

Specifies the holiday at which output is to end.

TO = *SAME

The specification entered for the FROM operand also applies here.

TO = *LAST-HOLIDAY

Output ends with the last holiday in the calendar file.

TO = <full-filename 1..30 without-cat-user-gen-vers with-wild>

Output ends with the specified holiday or with the group of holidays (in alphabetical order) identified in the wildcard string.

(You can select several holidays with a single wildcard string, where only the wildcard character * is permitted at the end for any character string.)

TO = *BY-NUMBER-OF-HOLIDAYS(...)

Determines the number of holidays to be output, beginning with the holiday specified in FROM.

NUMBER-OF-HOLIDAYS = <integer 1..1024>

Number of days to be output.

ASSIGNED-DATES = *NO / *YES

Specifies whether all calendar days that fall on the selected holidays are also to be listed.

The default value is *NO, i.e. no additional list is output.

OUTPUT = *SYSOUT / list-poss(2): *SYSOUT / *SYSLST(...)

Specifies whether output is to be written to SYSOUT or SYSLST.

The default value is *SYSOUT. Simultaneous output to both SYSLST and SYSOUT is possible (list specification).

OUTPUT = *SYSLST(...)

Output is written to SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>

Number of the SYSLST file.

The default value is STD, i.e. output is written to SYSLST.

When entering a number, you must make sure that a cataloged file is assigned to the corresponding SYSLST file (SYSLST01 through SYSLST99) (see the ASSIGN-SYSLST command).

Command return code

(SC2)	SC1	Maincode	Meaning / guaranteed messages
	0	CMD0001	Command executed without errors

Output formats

You can use the OUTPUT operand to route the desired output to SYSOUT and/or SYSLST. If desired, the user has the option of using a German-language interface. The layouts of the English interface, which are described below, are identical for SYSOUT and SYSLST.

SHOW-CALENDAR werk.kalender

```

%-----
%
%                               INFORMATION ABOUT CURRENT DAY
%-----
% CALENDAR NAME:   :10SN:$QM211.WERK.KALENDER
%
%-----
%   DATE      DAY ATTR #SYM  WORKING HOURS HOLIDAY
%-----
% 1995-03-24  FRI   W     0  08:00-16:30
%
%-----
%

```

SHOW-CALENDAR werk.kalender,SELECT=*BASIC-INFORMATION

```

%-----
%
%                               BASIC INFORMATION
%-----
% CALENDAR NAME:   :10SN:$QM211.WERK.KALENDER
%
%-----
% CALENDAR LIMITS                START : 1994-01-01
%                                END   : 1996-12-31
%-----
% STANDARD WEEK
%                                WORKING HOURS
%                                START  END
%                                (HH:MM) (HH:MM)
%-----
%   MON                W           08:00 - 16:30
%   TUE                W           08:00 - 16:30
%   WED                W           08:00 - 16:30
%   THU                W           08:30 - 16:30
%   FRI                W           08:00 - 16:30
%   SAT                F           00:00 - 00:00
%   SUN                F           00:00 - 00:00
%-----
%

```

SHOW-CALENDAR werk.kalender,SELECT=*DATE(FROM=*TODAY,TO=1995-03-30)

```

%-----
%                               LIST OF DAYS
%-----
% CALENDAR NAME:   :IOSN:$QM211.WERK.KALENDER
%-----
%   DATE      DAY ATTR #SYM  WORKING HOURS HOLIDAY
%-----
% 1995-03-24  FRI   W    0  08:00-16:30
% 1995-03-25  SAT   F    0  00:00-00:00
% 1995-03-26  SUN   F    0  00:00-00:00
% 1995-03-27  MON   W    0  08:00-16:30
% 1995-03-28  TUE   W    0  08:00-16:30
% 1995-03-29  WED   W    0  08:00-16:30
% 1995-03-30  THU   W    1  08:30-16:30
%-----

```

SHOW-CALENDAR werk.kalender,SELECT=*DATE(FROM=1995-03-29,
TO=1995-03-31,ASSIGNED-SYM-DATE=*ALL)

```

%-----
%                               DAY INFORMATION
%-----
% CALENDAR NAME:   :IOSN:$QM211.WERK.KALENDER
%-----
% 1995-03-29  WED   W    0  08:00-16:30
%-----
%   DATE      DAY ATTR #SYM  WORKING HOURS HOLIDAY
%-----
% 1995-03-31  THU   W    1  08:30-16:30
%-----
% SYMBOLIC DATE ----- TIME ----- SYMBOLIC DATE ----- TIME ----
% FRIDAYFORUM                13:30:00
%-----
%   DATE      DAY ATTR #SYM  WORKING HOURS HOLIDAY
%-----
% 1995-03-31  FRI   W    1  08:00-16:30
%-----
% SYMBOLIC DATE ----- TIME ----- SYMBOLIC DATE ----- TIME ----
% ULTIMO                16:00:00
%-----

```

```
SHOW-CALENDAR werk.kalender,SELECT=*SYMBOLIC-DATE
(FROM=*FIRST-SYMBOLIC-DATE,TO=*LAST-SYMBOLIC-DATE)
```

```
%-----
%
%                               LIST OF SYMBOLIC DATES
%-----
% CALENDAR NAME:   :1OSN:$QM211.WERK.KALENDER
%
%-----
% SYMBOLIC DATE           TIME      TYPE    CYCLTYP  CYCLVAL  CYCLALT
%-----
% FRIDAYFORUM            13:30:00   C      WEEK           1  SKIP
% ULTIMO                  16:00:00   C      MONTH           1  BEFORE
%-----
```

```
/SHOW-CALENDAR werk.kalender,SELECT=*SYMBOLIC-DATE
(FROM=*FIRST-SYMBOLIC-DATE,TO=*LAST-SYMBOLIC-DATE,
ASSIGNED-DATES=*INTERVAL(FROM=*TODAY,TO=1995-03-31))
```

```
%-----
%                               SYMBOLIC DATE INFORMATION
%-----
% CALENDAR NAME:   :1OSN:$QM211.WERK.KALENDER
%
%-----
% SYMBOLIC DATE           TIME      TYPE    CYCLTYP  CYCLVAL  CYCLALT
%-----
% FRIDAYFORUM            13:30:00   C      WEEK           1  SKIP
%-----
%                               ASSIGNED DATES
%-----
% 1995-03-10  1995-03-17  1995-03-24  1995-03-31
%-----
% SYMBOLIC DATE           TIME      TYPE    CYCLTYP  CYCLVAL  CYCLALT
%-----
% ULTIMO                  16:00:00   C      MONTH           1  BEFORE
%-----
%                               ASSIGNED DATES
%-----
% 1995-03-31
%-----
```

```
SHOW-CALENDAR werk.kalender, SELECT=*HOLIDAY(FROM=CHRISTMAS,
TO=EASTER)
```

```

%-----
%
%                               LIST OF HOLIDAYS
%-----
% CALENDAR NAME:   :10SN:$QM211.WERK.KALENDER
%
%-----
% HOLIDAY NAME           TYPE    ACTIVE
%-----
% CHRISTMAS              C      Y
% EARLYMAYDAY           C      Y
% EASTER                 N      Y
%-----

```

```
SHOW-CALENDAR werk.kalender, SELECT = *HOLIDAY(FROM=CHRISTMAS,
TO=EASTER, ASSIGNED-DATES=*YES)
```

```

%-----
%                               HOLIDAY INFORMATION
%-----
% CALENDAR NAME:   :10SN:$QM211.WERK.KALENDER
%
%-----
% HOLIDAY NAME           TYPE    ACTIVE    DATE
%-----
% CHRISTMAS              C      Y          ****-12-24
%-----
%                               ASSIGNED DATES
%-----
% 1995-12-24
%-----
% HOLIDAY NAME           TYPE    ACTIVE    DATE
%-----
% EARLYMAYDAY           C      Y          ****-05-01
%-----
%                               ASSIGNED DATES
%-----
% 1995-05-01
%-----
% HOLIDAY NAME           TYPE    ACTIVE    DATE
%-----
% EASTER                 N      Y
%-----
%                               ASSIGNED DATES
%-----
% 1995-04-16
%-----

```

Meanings of the output fields

ACTIVE	Activation status of the holiday: Y = activated, N = not activated
ASSIGNED DATES	Dates of the calendar days assigned to a SYMDAT or a holiday; output in the format yyyy-mm-dd
ATTR	Attribute of the weekday: W = working day, F = free day
CALENDAR LIMITS	Calendar limits:
START	First date in the calendar (lower calendar limit); output in the format yyyy-mm-dd
END	Last date in the calendar (upper calendar limit); output in the format yyyy-mm-dd
CALENDAR NAME:	Name of the calendar file
CYCLALT	Alternative for a case where a calculated SYMDAT entry is a free day; possible values: BEFORE = insert before free day, AFTER = insert after free day, SKIP = skip day, ON = insert on free day
CYCLTYP	Cycle type of the SYMDAT: DAY, WORKDAY, MONTH, WEEK
CYCLVAL	Cycle value
DATE	Date; output in the format yyyy-mm-dd; for cyclic holidays, output in the format: ****-mm-dd
DAY	Name of the weekday: MON (Monday), TUE (Tuesday), WED (Wednesday), THU (Thursday), FRI (Friday), SAT (Saturday), SUN (Sunday)
HOLIDAY	Name of the associated holiday
HOLIDAY NAME	Name of the holiday; max. 30 characters
SYMBOLIC DATE	Name of a SYMDAT; max. 20 characters
STANDARD WEEK	Definitions of the standard working week include the following information: DAY, ATTR and WORKING HOURS (see relevant descriptions)
TIME	Time entry assigned to the SYMDAT; output in the format hh:mm:ss
TYPE	Type of SYMDAT or holiday: C = cyclic, N = non-cyclic

WORKING HOURS	Beginning and end of the working hours; output in the format hh:mm-hh:mm
#SYM	Number of SYMDATs assigned to a particular day

References

The manuals are available as online manuals, see <http://manuals.ts.fujitsu.com>, or in printed form which must be paid and ordered separately at <http://manualshop.ts.fujitsu.com>.

- [1] **BS2000/OSD-BC**
Commands
User Guide
- [2] **JV (BS2000/OSD)**
Job Variables
User Guide

Index

A

- activate holidays 17
- attribute of a weekday 12
- attribute rule 13

B

- basic information 11
 - modify (macro) 123
 - output (macro) 152

C

- C header file 204

calendar

- close (macro) 97
- create (macro) 99
- lower limit 11
- open (macro) 76, 147
- save (macro) 150
- upper limit 11

calendar data 7

calendar day with assigned holidays

- output (macro) 155

calendar day with assigned SYMDATs

- output (macro) 159

calendar days 13

- initial values 12
- output (macro) 169

calendar editor

- interrupt 25
- list holidays 61
- mask H010 61
- mask H020 64
- output holiday information 64
- start 25
- terminate 25

calendar file 19

- file attributes 20
- security mechanisms 19

calendar limits 11

- move 11

calendar lower limit 11

calendar structure 11

calendar upper limit 11

calendar editor 23

- check input 31
- edit day information 49
- evaluate input 31
- function keys 33
- functions 29
- list calendar days 46
- main menu 34
- mask areas 28
- mask B020 42
- mask C000 34
- mask C010 37
- mask D010 46
- mask D020 49
- mask format 28
- mask layout 28
- mask S010 53
- mask S020 56
- open mode 34
- operate 24
- output basic information 42
- output SYMDAT information 56
- output SYMDAT list 53
- overview of functions 23, 24
- overview of masks 27
- page function 30
- position 30

- calendar editor (cont.)
 - select functions 37
 - select object 31
 - classification of return codes (subcode 1) 85
 - close calendar (macro) 97
 - command
 - SHOW-CALENDAR 217
 - command interface 7, 217
 - command return code 224
 - control block (macro) 75
 - create
 - calendar (macro) 99
 - non-cyclic SYMDAT (macro) 113
 - create layouts (macro) 74
 - cycle
 - SYMDAT 15
 - cycle type
 - SYMDAT 15
 - cycle value
 - SYMDAT 15
 - cyclic holiday 17
 - cyclic SYMDAT 14
- D**
- daily cycle
 - SYMDAT 16
 - data transfer 32
 - dates
 - symbolic 7
 - deactivate holiday (macro) 117
 - delete SYMDAT (macro) 121
 - dialog interface 7, 23
 - DSECT 193
- E**
- end time
 - working hours 12
- F**
- file attributes (calendar file) 20
 - free-day rule 15
 - function
 - page 30
 - position 30
 - function keys (overview) 33
 - function overview (macro) 94
- G**
- GOTO specification 30
- H**
- holiday 17
 - activate 17
 - cyclic 17
 - naming conventions 17
 - non-cyclic 17
 - output (macro) 164, 174
 - holiday file 18
 - structure 18
- I**
- information structure 11
 - initial values
 - calendar days 12
 - installation 20
 - interfaces 7
 - interrupt
 - calendar editor 25
- J**
- job variable
 - monitor program 26
- L**
- layout overview 82
 - lower limit
 - calendar 11
- M**
- macro
 - control block 75
 - create layouts 74
 - example 87
 - function overview 94
 - layout overview 82
 - open calendar 76
 - performance 69
 - macro call format 198

macro function call

- close calendar 97
- create calendar 99
- create non-cyclic SYMDAT 113
- deactivate holiday 117
- delete SYMDAT 121
- modify basic information 123
- modify calendar day 129
- modify cyclic holiday 127
- modify cyclic SYMDAT 133
- modify non-cyclic holiday 138
- modify non-cyclic SYMDAT 142
- open calendar 147
- output basic information 152
- output calendar day with assigned holidays 155
- output calendar day with assigned SYMDATs 159
- output holiday with assigned calendar days 164
- output list of calendar days 169
- output list of holidays 174
- output list of SYMDATs 178
- output SYMDAT with assigned calendar days 183
- save calendar 150

macro function output

- next time for a set of SYMDATs 188

macro syntax 198

maincode (return code) 84

mask

- description of areas 28
- exit 29
- format 28
- layout 28

masks, overview

- calendar editor 27

message file 20

metasyntax of macro 199, 200

MF operand 201, 202, 203

modify

- basic information (macro) 123
- calendar day (macro) 129
- cyclic holiday 127

- non-cyclic holiday (macro) 138
- non-cyclic SYMDAT (macro) 142

modify calendar day (macro) 129

modify holiday (macro) 127

monthly cycle

- SYMDAT 15

move calendar limits 11

N

naming conventions

- holiday 17
- SYMDAT 14

non-cyclic holiday 17

- modify (macro) 138

non-cyclic SYMDAT 14

- create (macro) 113
- modify (macro) 142

O

open calendar (macro) 147

output

- basic information (macro) 152
- calendar day with assigned holidays (macro) 155
- calendar day with assigned SYMDATs (macro) 159
- calendar days (macro) 169
- holiday (macro) 164
- list of holidays (macro) 174
- list of SYMDATs (macro) 178
- SYMDAT (macro) 183

P

page function 30

parameter list 193

program

- monitor (job variable) 26

program interface 7, 69

- format overview 71

R

read calendar file (command) 217

read calendar file (SHOW-CALENDAR) 217

README file 9

return code (command) 224
return codes 84
 overview 85
return codes (macro) 84
 classification 85
 exceptions 85
 overview 85

S

S variable 7
save calendar (macro) 150
SDF-P procedure 7
SHOW-CALENDAR command 217
 output formats 225
standard header 84
standard working week 12
start
 calendar editor 25
start date
 SYMDAT 15
start time
 working hours 12
structure of a calendar 11
structure of the holiday file 18
subcode 1 (return code) 84
 meaning 85
subcode 2 (return code) 84
subsystem declaration 20
symbolic date 14
SYMDAT 14
 cycle types 15
 cycle value 15
 cyclic 14
 daily cycle 16
 monthly cycle 15
 naming conventions 14
 non-cyclic 14
 output (macro) 178, 183
 start date 15
 time specification 14
 weekly cycle 16
 workday cycle 16
SYMDAT output (Macro) 188
syntax file 20

T

terminate
 calendar editor 25
time specification (SYMDAT) 14

W

weekday 12
 attribute 12
weekly cycle
 SYMDAT 16
workday cycle
 SYMDAT 16
working hours 12
 end time 12
 start time 12