CHICAGO BOOTH

CRSP®
Center for Research in Security Prices

CRSP Fortran 95 API Guide

CRSP US Stock, US Indexes, and CCM Databases

CHICAGO BOOTH

CRSP®
Center for Research in Security Prices

105 West Adams, Suite 1700
Chicago, IL 60603
Tel: 312.263.6400
Fax: 312.263.6430
Email: Support@crsp.ChicagoBooth.edu

Contents

Introduction

The supplied suite of CRSP utilities allows full-featured access to CRSP databases and is intended to cover a variety of the most typical queries and uses of CRSP data. The features of CRSP tools can often save end-users the whole effort of programming their own reporting utilities. In other cases the user-program flow can be significantly simplified by the use of output files produced from CRSP tools in a number of widely accepted formats.

For situations when a user-program requires direct access to CRSP datasets, a CRSP API library is provided in Fortran 95 (F95). The CRSP API comprises the platform-specific object library and a set of F95 precompiled modules and include-files. Additionally, a set of sample program sources is supplied along with respective make files and test data to build and test the samples.

Item-Based Access

The CRSP Fortran 95 API introduces item-based access which generally supersedes the older set-based method of programming access to CRSP databases.

Conceptually, the item-based access allows uniform access to the whole array of CRSP datasets by presenting them as collections of items that are instances of a generic **item object**. Each specific CRSP dataset is represented as an instance of a generic **access-handle object**.

In item-access context users are shielded from the internal mechanics of access to the specific structures of CRSP datasets. This adds more uniformity to your code, especially in cases of accessing a mixed set of CRSP data, thus simplifying code development and maintenance.

In addition to its dynamic and extensible nature, item-access also introduces a set of **derived items** that are defined as parameterized functional combinations of underlying regular items. These items significantly improve code consistency and simplicity by internally performing all of the necessary calculations that previously had to be coded explicitly in user programs.

Legacy Set-Based Access

Existing Fortran 95 user-programs that do not access CRSP Compustat dataset should not require migration to item-based access. The supplied Fortran 95 CRSP API contains the needed underlying functions for set-based access.

However, mixing of the legacy set-based access and item-access contexts in the same program is not recommended, so any new user-programs that access CRSP data should be implemented in the item-access context. CRSP supplies a set of sample programs that demonstrate the flow of programs using item-access.

Building and Executing Programs

The CRSP Fortran 95 API includes a variety of sample programs illustrating item-based access to CRSP databases from Fortran 95 programs. This section describes the sample programs and shows you how to build and execute them on the Windows XP, Sun Solaris, and Linux platforms.

Before creating your own programs, it is a good idea to first build and execute one or more of these sample programs. Besides illustrating CRSP API programming techniques, the sample programs have been tested and debugged by CRSP. If you can successfully run them, then you know your programming environment is correctly configured.

The following table lists the supplied sample programs, organized by database.

CRSP Database	Fortran 95 Sample Programs	Make File/ Platform .ext
CRSP/Compustat Merged Database	**ccmitm_fsamp1.f90** –Sequential item-access to CRSP Compustat dataset **ccmitm_fsamp2.f90** –Direct item-access to CRSP Compustat dataset by GVKEY list **ccmitm_fsamp3.f90** –Direct item-access to CRSP Compustat securities data by GVKEY.IID **ccmitm_fsamp4.f90** –Use of CRSP Link for Compustat, item-access by CRSP permno.	**Windows / SunOS /Linux Lahey Fortran 95 / Linux G95** `f95_samp_ccm.mak / .mk/ .mk5 / .mkg5`
CRSP US Stock Database	**stkitm_fsamp1.f90** –sequential item-access to CRSP Stock dataset. **stkitm_fsamp2.f90** –direct item-access to CRSP Stock dataset by CRSP permno. **stkitm_fsamp4.f90** –use of regular and derived data-items for CRSP Stock dataset.	**Windows / SunOS /Linux Lahey Fortran 95 / Linux G95** `f95_samp_stkitm.mak / .mk/ .mk5 / .mkg5`
CRSP US Index Database	**inditm_fsamp1.f90** –sequential item-access to CRSP Stock & Index Database. **inditm_fsamp2.f90** –direct item-access to CRSP Stock Ind dataset by CRSP indno.	**Windows / SunOS /Linux Lahey Fortran 95 / Linux G95** `f95_samp_inditm.mak / .mk/ .mk5 / .mkg5`

API Environment

Environment Variables

The CUPL installation process sets a number of environment variables pointing to the locations of modules, include and library files, as well as sample programs. The values of these environment variables are given in the following table, broken down by platform.

PLATFORM	F95 MODULES (*.mod)	F95 INCLUDES (*.inc)	F95 LIBRARY	F95 SAMPLES (*.f90)
Windows 32-bit **Windows 64-bit**	%CRSP_INCLUDE%\mod	%CRSP_INCLUDE%	%CRSP_LIB%	%CRSP_SAMPLE%
SunOS sparc	$CRSP_INCLUDE/mod	$CRSP_INCLUDE	$CRSP_LIB	$CRSP_SAMPLE
SunOS i86pc	$CRSP_INCLUDE/mod	$CRSP_INCLUDE	$CRSP_LIB	$CRSP_SAMPLE
Linux 32-bit **Linux 64-bit**	**Lahey Fortran 95 compiler (32-bit):** $CRSP_INCLUDE/mod **G95 complier:** $CRSP_INCLUDE/mod_g95	$CRSP_INCLUDE	$CRSP_LIB	$CRSP_SAMPLE

Compiler Options

Platform-specific Fortran 95 compiler options used with Fortran 95 CRSP API are listed in the table below. Refer to the CRSPAccess Release Notes for specific versions of the supported Fortran 95 compilers.

PLATFORM	FORTRAN 95 COMPILER OPTIONS
Windows 32-bit **Windows 64-bit**	**Intel VisualFortran:** `ifort /Qvec- /I %CRSP_INCLUDE% /I %CRSP_INCLUDE%\mod`
SunOS sparc	**Sun Pro Fortran 95:** `f95 -w -xarch=v9  -ext_names=plain -I$CRSP_INCLUDE -M$CRSP_INCLUDE/mod -KPIC`
SunOS i86pc	**Sun Pro Fortran 95:** `f95 -w -xtarget=generic64 -ext_names=plain -I$CRSP_INCLUDE -M$CRSP_INCLUDE/mod   -KPIC`

Linux 32-bit
 	
Linux 64-bit
	Lahey Fortran 95 (32-bit) :
	
	`lf95 -w -I$CRSP_INCLUDE -I$CRSP_INCLUDE/mod  -Am`
	
	G95:
	
	`g95 -w -I$CRSP_INCLUDE -I$CRSP_INCLUDE/mod_g95`

Fortran 95 Libraries

Platform-specific Fortran 95 libraries and options for linking with Fortran 95 CRSP API are listed in the table below:

PLATFORM	Fortran 95 LINK LIBRARIES
Windows 32-bit **Windows 64-bit**	**Intel VisualFortran:** `%CRSP_LIB%\crsp_lib_f95.lib  %CRSP_LIB%\crsp_lib.lib`
SunOS sparc	**Sun Pro Fortran 95:** `$CRSP_LIB/crsplib_f95.a  $CRSP_LIB/crsplib.a -lm -lnsl`
SunOS i86pc	**Sun Pro Fortran 95:** `$CRSP_LIB/crsplib_f95.a  $CRSP_LIB/crsplib.a -lm –lnsl`

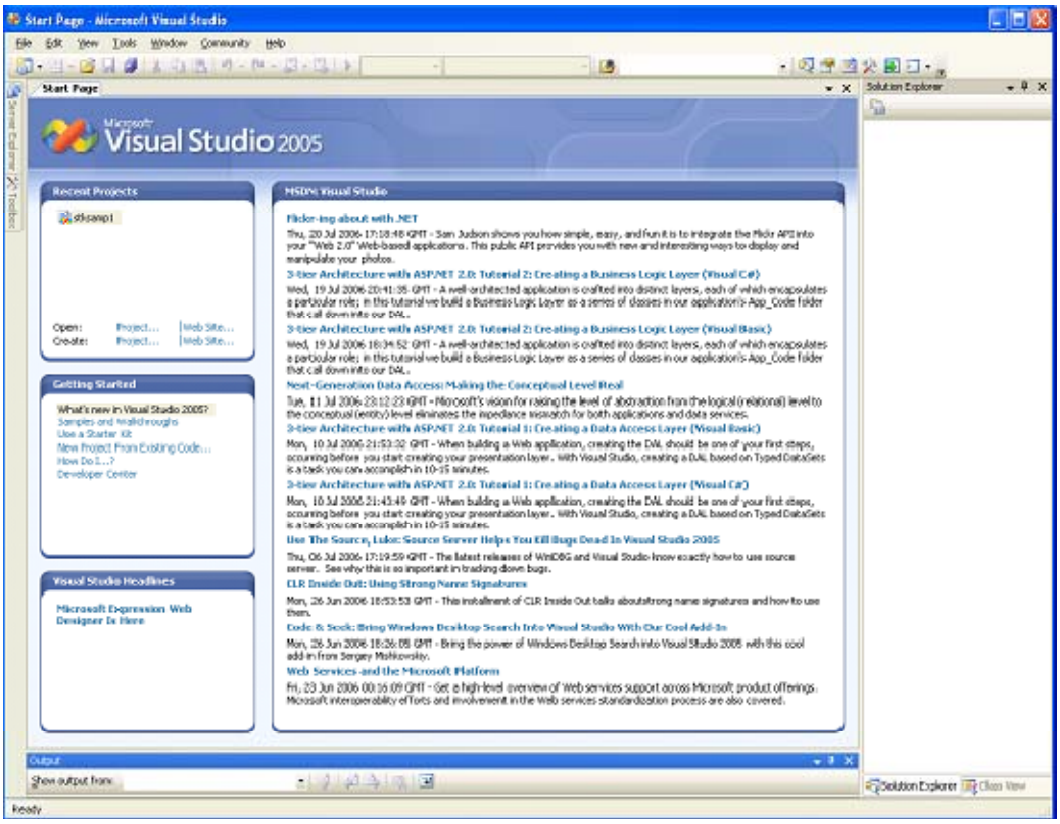
Linux 32-bit
 	
Linux 64-bit
	Lahey Fortran 95 (32 bit):
	
	`$CRSP_LIB/crsplib_f95.a  $CRSP_LIB/crsplib.a –lm`
	
	G95:
	
	`$CRSP_LIB/crsplib_g95.a  $CRSP_LIB/crsplib.a -lm`

Microsoft Windows

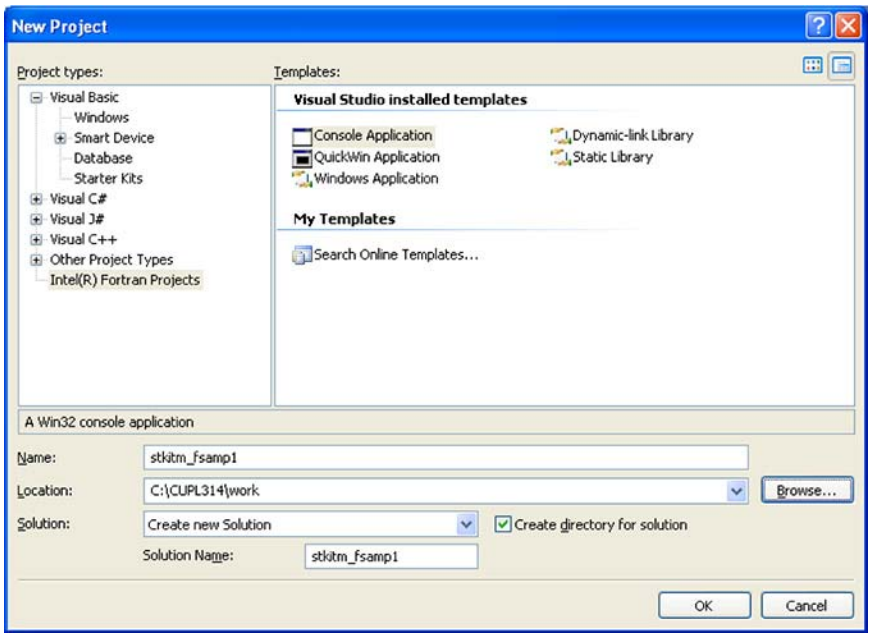
Microsoft Visual Studio 2005

The following walks you through the steps to build and run the `stkitm_fsamp1.f90` sample program using Microsoft Visual Studio 2005. This sample program is located in the Sample folder in the CRSP root directory where you have installed the CRSPAccess software.

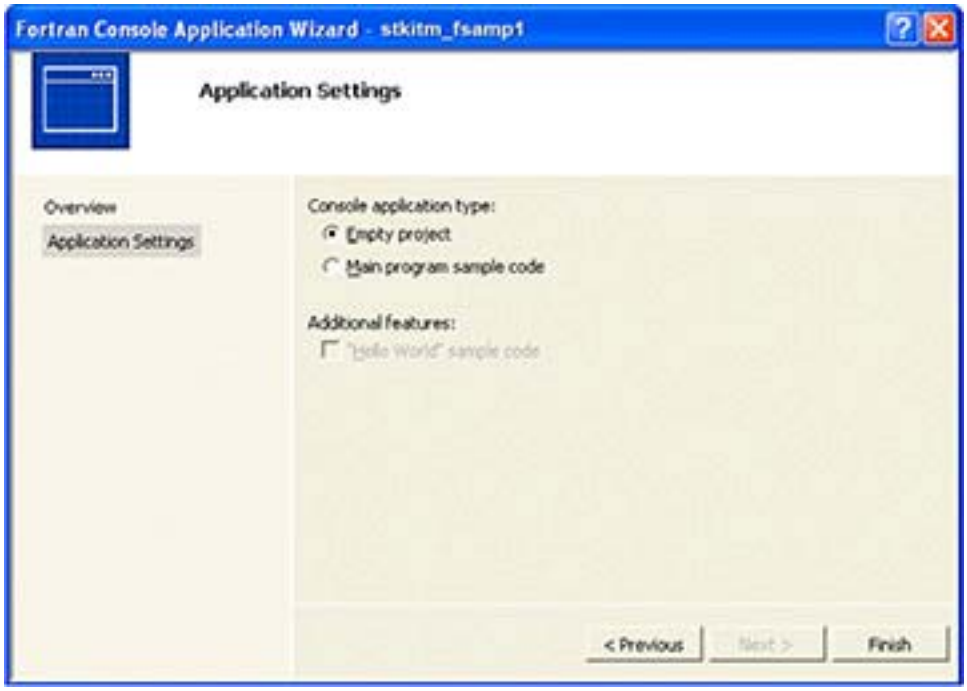
1. Open the Microsoft Visual Studio 2005 development environment start page. **Start→Programs→Microsoft Visual Studio 2005** opens the screen below. Click on the **Create: Project** button at the upper left of your screen, or from the Menu bar select **File→New→Project**.



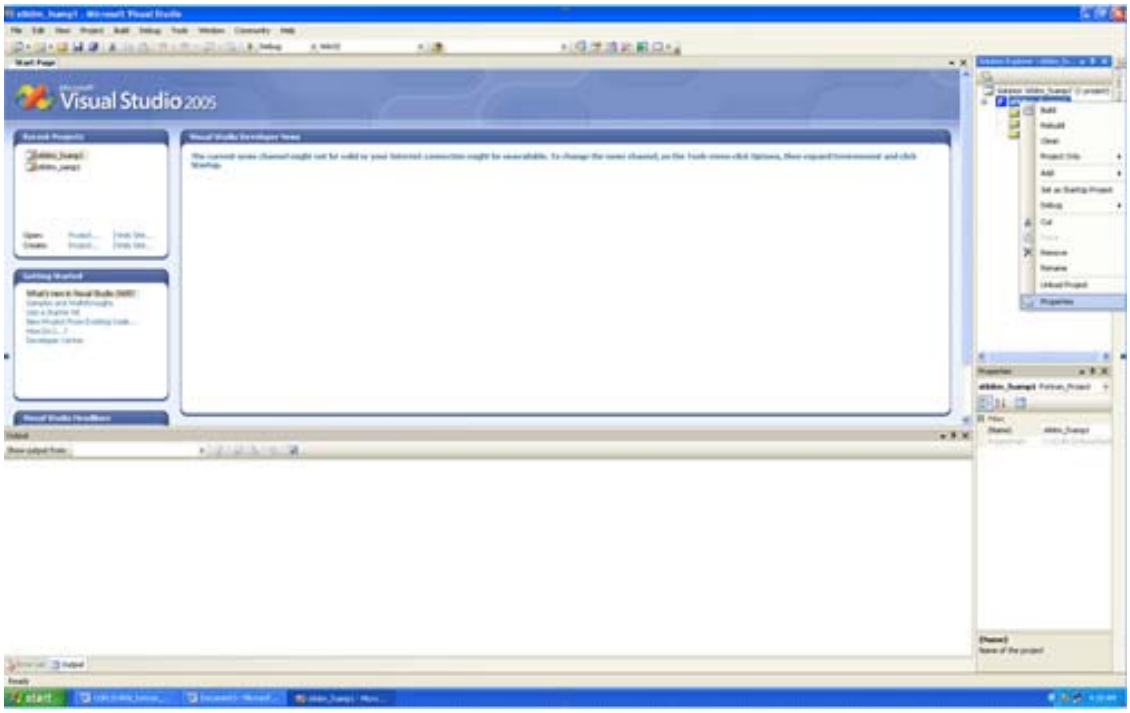
2. To create a new project, highlight the Intel(R) Fortran Projects folder in the Project Types box on the left and Console Application in the Templates box on the right. Enter the name of the project you are creating in the Name box below as "`stkitm_fsamp1`". Move the cursor to the Location and overwrite as `C:\CUPL314\work\`, or the directory in which you wish to work. Click OK.



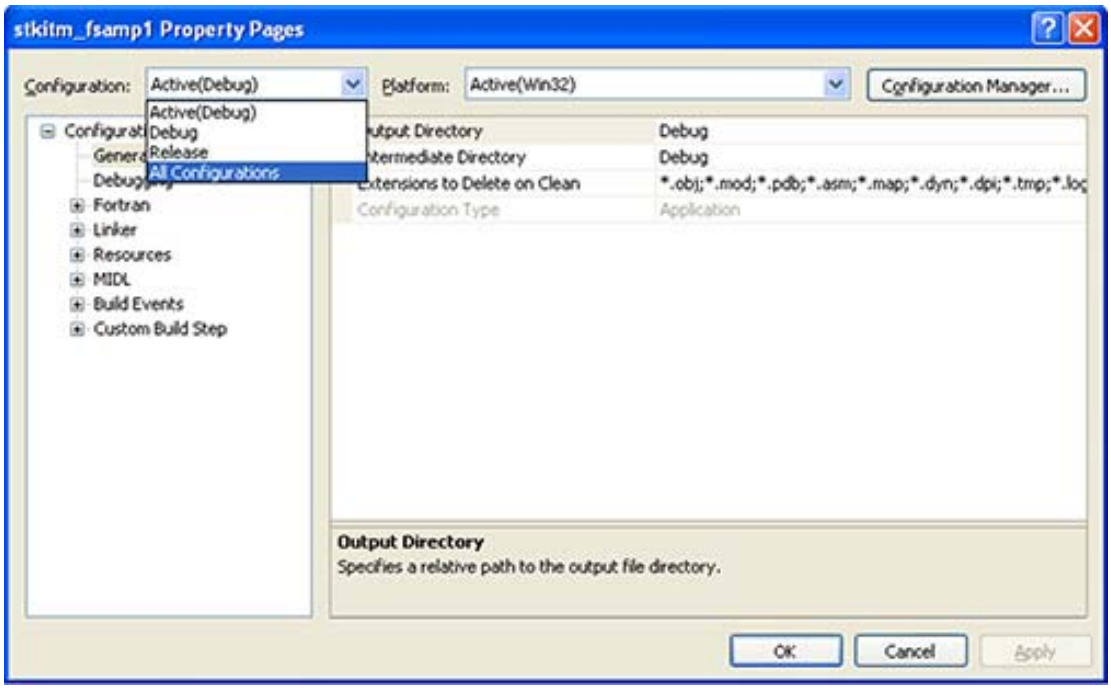
3. You have now opened the Fortran Console Application Wizard. Click on **Application Settings→Empty project**. Click **Finish**.



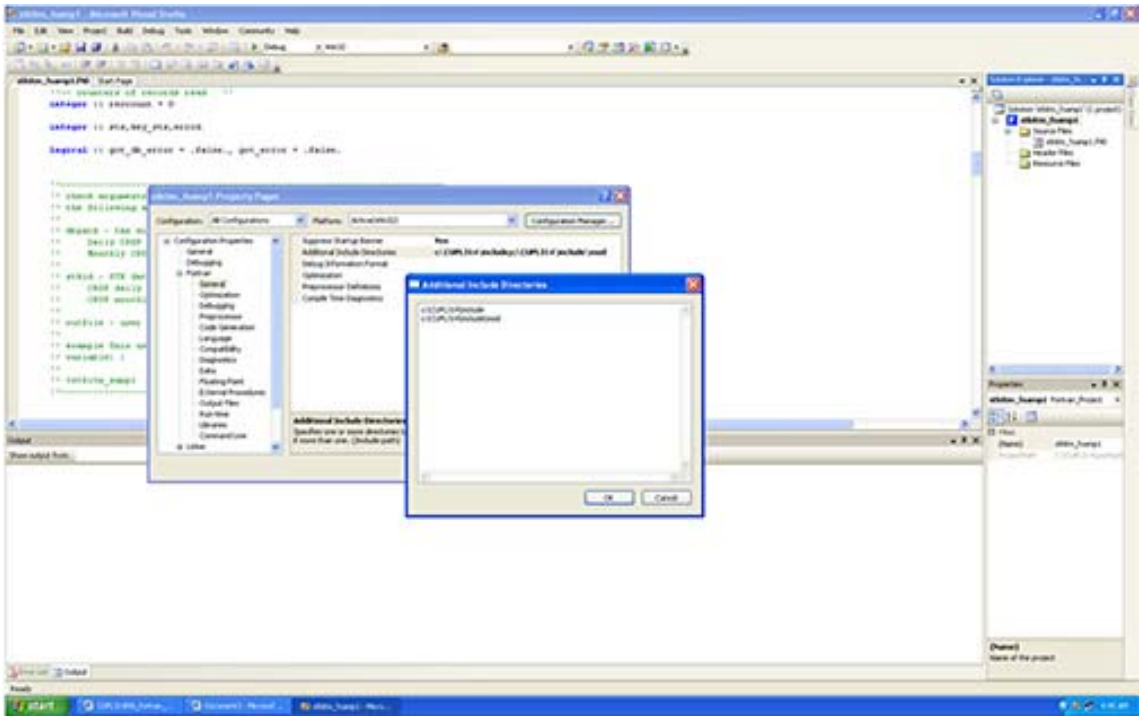
4. Once your new project is added, right-click on `stkitm_fsamp1` and scroll down the menu panel to select Properties. You will advance to the Properties Pages. Changes to project solutions are reflected in the following three steps.



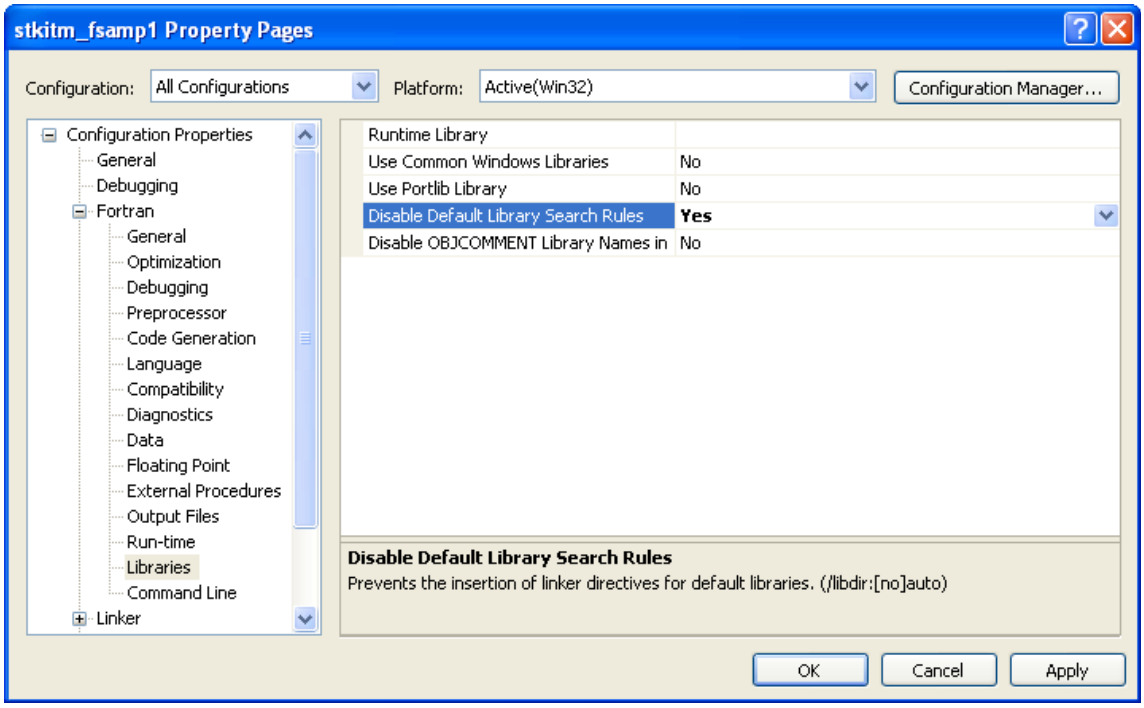
5. In order to be able to run the sample program in either Debug or Release mode, Click on the **Configuration** drop-down and select **All Configurations**.



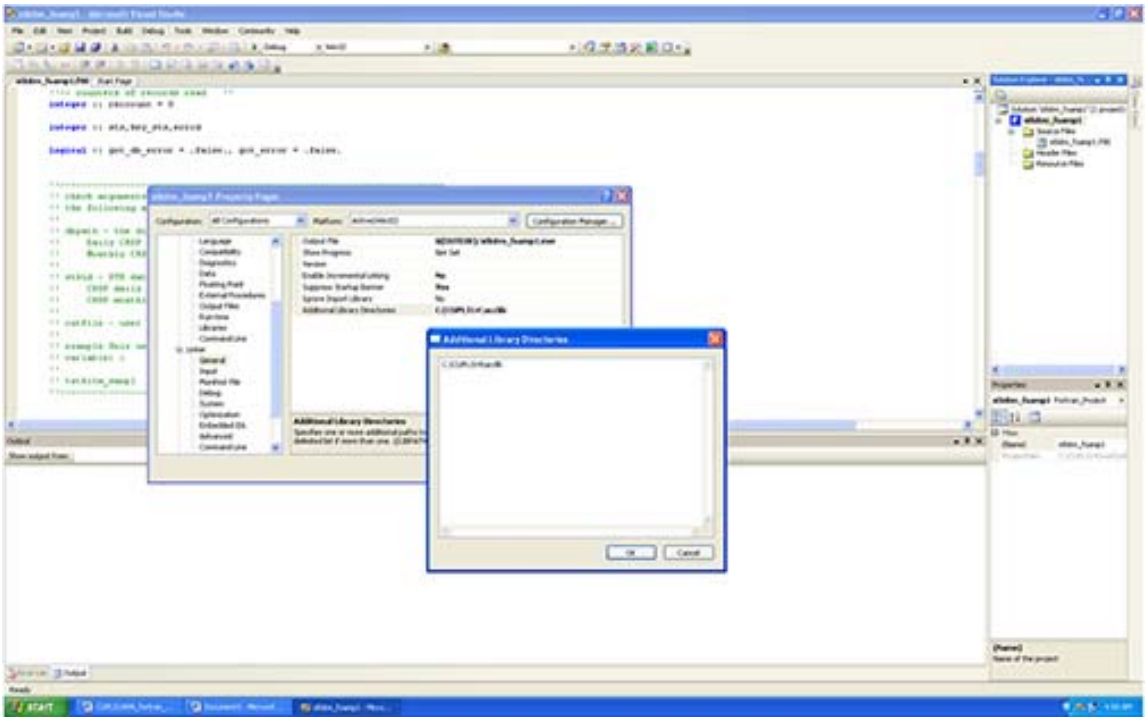
6. Highlight **Fortran** and select **General**. Click on **Additional Include Directories** and add `C:\CUPL314\include and C:\CUPL314\include\mod`



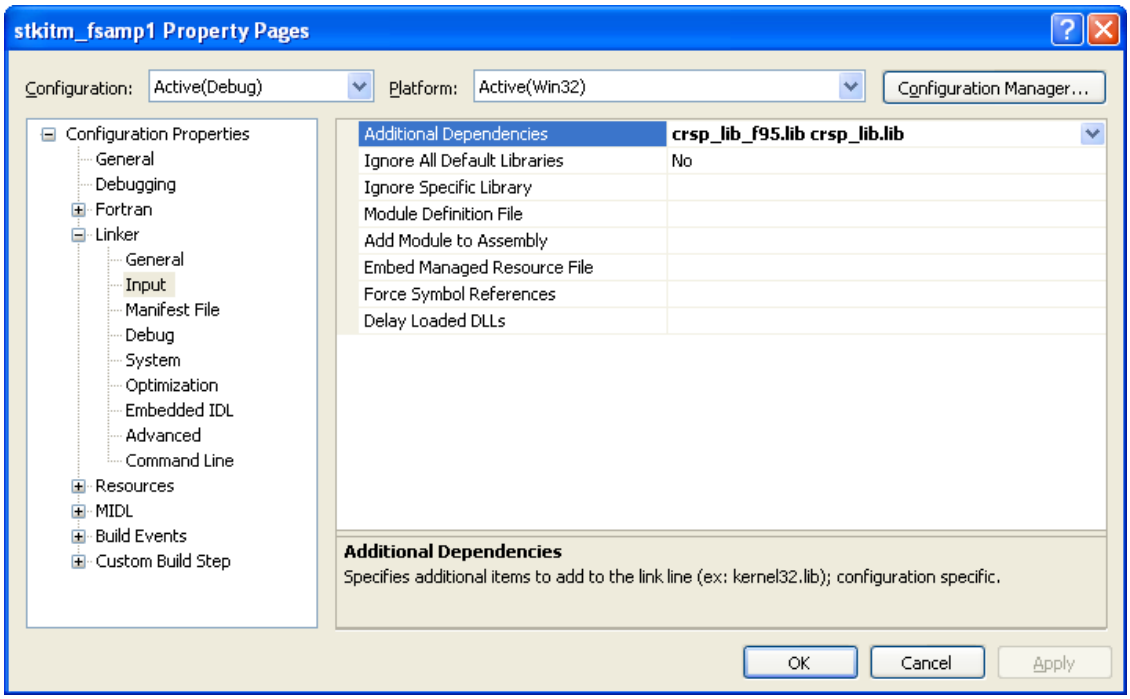
7. Select **Libraries**, highlight **Disable Default Library Search Rules**, and select **Yes**.



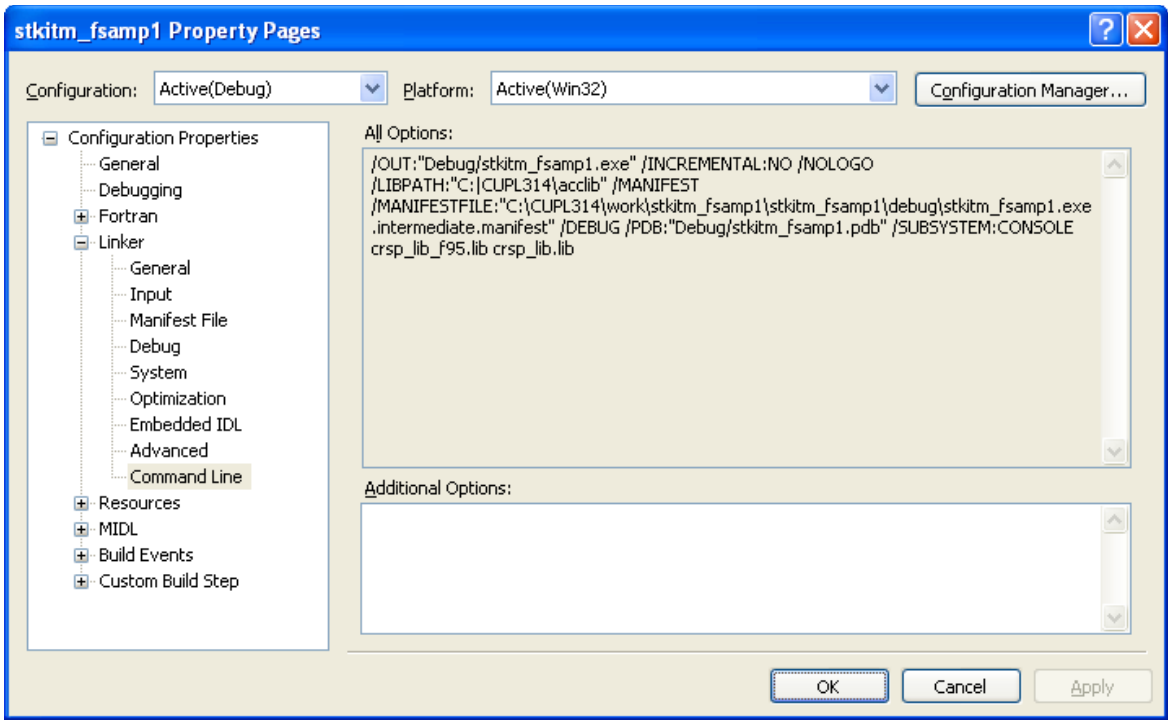
8. Select **Linker→General, Additional Library Directories** and enter `C:\CUPL314\acclib`.



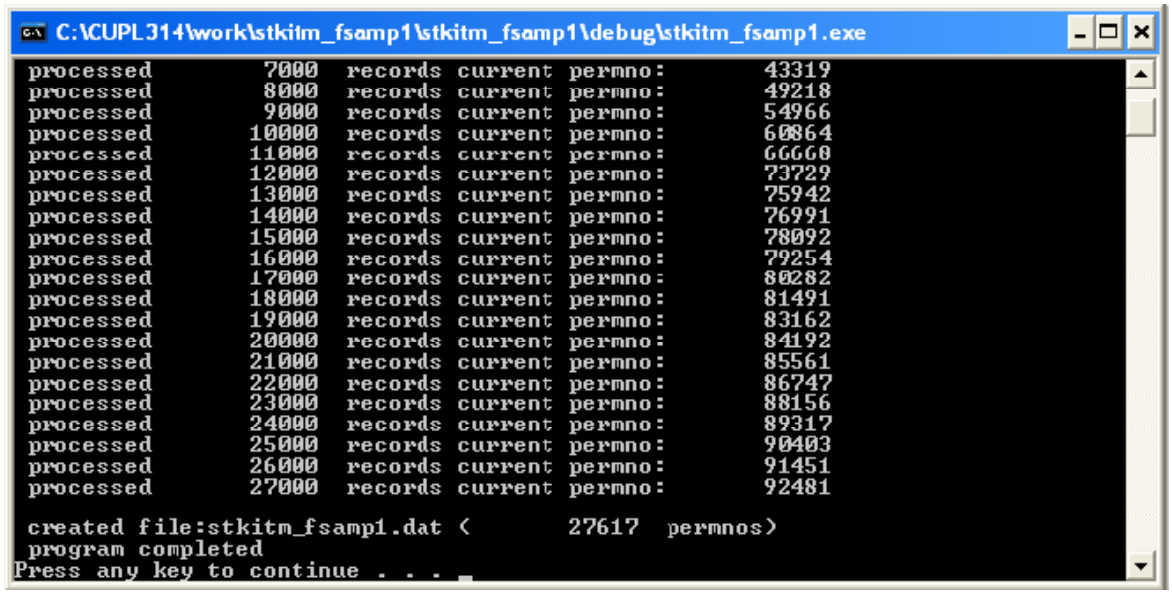
9. Still in **Linker**, select **Input**, and in **Additional Dependencies**, type `crsp_lib_f95.lib crsp_lib.lib`.



10. Select Command Line and click on Apply to set the properties of your project.

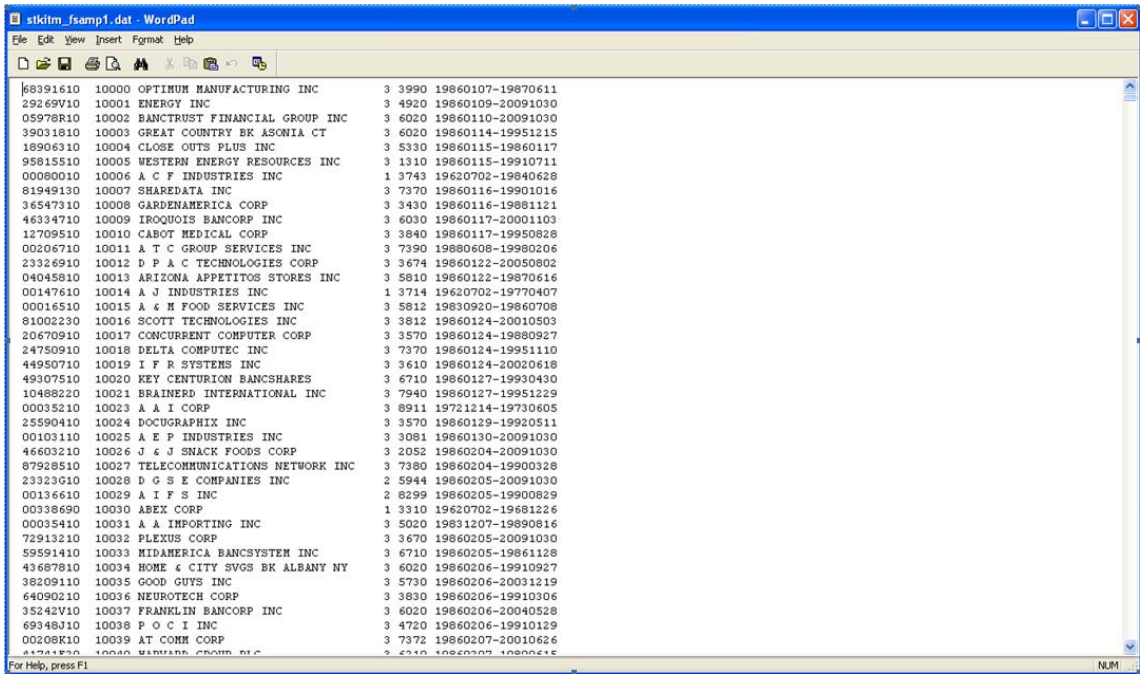


11. To run the program you have just built, from the menu bar, click on: **Debug→Start without Debugging**. The following screen will appear and indicate that the program is complete:



12. The output that you created by running the `stkitm_fsamp1` program is stored in the folder where you initially created the project: `c:\CUPL314\work\stk_samp1`. The output file is a

text file called “`dcnames.dat`”. Note that your output may differ depending on the end date of the database that you are using.

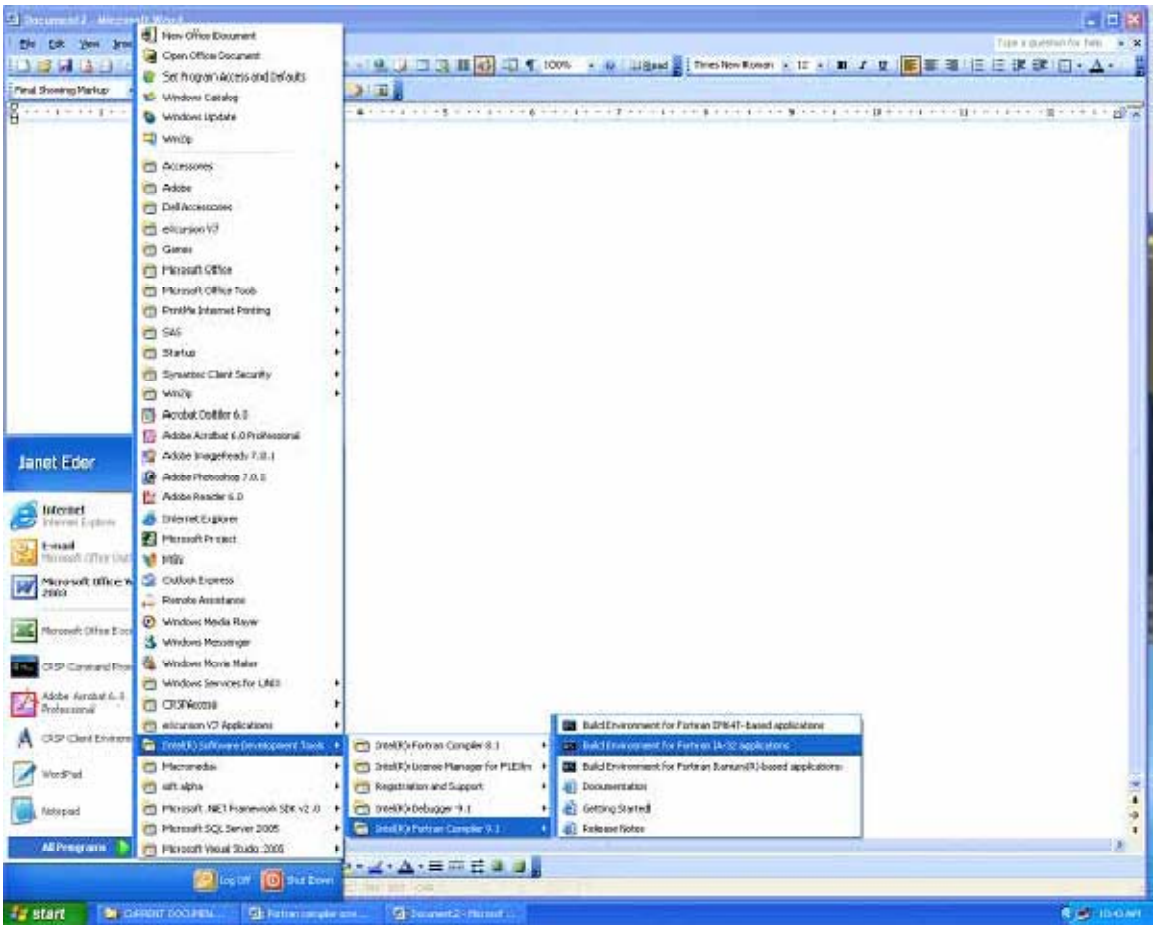


Windows Command Prompt

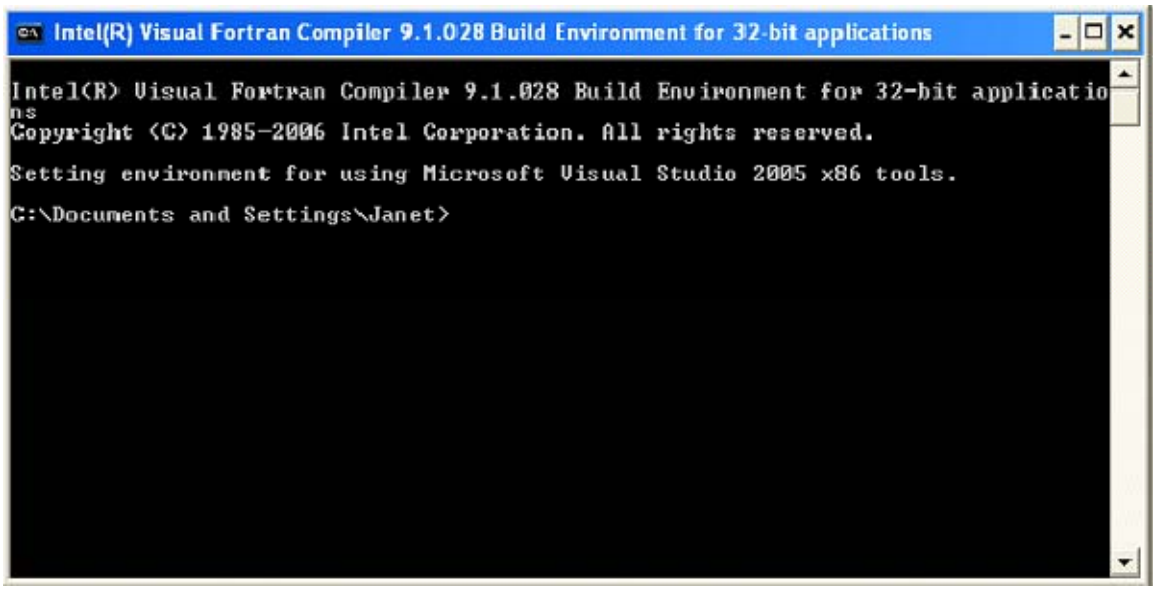
The sample programs can also be compiled and run from a command prompt window. In order to do so, the environment must be set for Intel Fortran to run.

To set the Windows 32-bit environment to Intel(R) Fortran, click on **Start→All Programs→Intel(R) Software Development Tools→Intel(R) Fortran Compiler 9.1→Build Environment for Fortran IA-32 applications**.

To set the Windows 64-bit environment to Intel(R) Fortran, click on **Start→All Programs→Intel Parallel Studio XE 2011→Command Prompt →Parallel Studio XE with Intel Compiler→Intel 64 Visual Studio 2008 mode.**



A DOS window will open ready for you to run your Fortran 95 programs.



Compiling from the Command Prompt

To compile a sample program from the command prompt using the Intel Fortran Compiler, copy it to your program directory and invoke the invoke the `ifort` command as shown below.

```
> copy %crsp_sample%\stkitm_fsamp1.f90 .
```

```
> ifort /I%crsp_include% stkitm_fsamp1.f90 %crsp_lib%\crsp_lib.lib
%crsp.lib%\crsp_lib_F95.lib
```

To run the program:

```
> .\stkitm_fsamp1
```

Using a Make File

Sample programs can also be compiled and linked at the command prompt using the `nmake` utility. A sample description file, `f95_samp.mak`, exists in the `%crsp_sample%` directory. To use the sample description file with your own program, copy it to your program directory and modify it to include your program instead of the sample.

```
> copy  %crsp_sample%\f95_samp.mak  .
```

To compile a specific sample program:

```
> nmake /f f95_samp.mak stkitm_fsamp1.exe
```

To compile all sample programs:

```
> nmake /f f95_samp.mak
```

To run the program:

```
> .\stkitm_fsamp1
```

Sun Solaris

CRSP currently supports Sun Sparc Solaris 2.9/5.9 with the Forte Developer 7.0, Fortran 95 7.0, and Sun x86 Solaris 2.9/5.9.

Fortran was compiled and tested using the above compiler. Fortran library functions interface with C functions in the CRSP object library. Ordinary sample Fortran usage links to the object library, but does not require compiling C programs.

CRSP access depends on environment variables set during installation. Environment variables can be used on Unix with the name preceded by the $ symbol. All file names and environment variable names are case sensitive on Unix systems. The env command can be used in a terminal window to find available environment variables.

Important CRSP files or directories can be found with the following names:

$CRSP_BIN	Directory containing executable programs and shell scripts files. This directory is in the PATH so programs can be run from any directory. Executable versions of the sample programs can be found in this directory.
$CRSP_LIB	Directory containing CRSP object library and internal files.
$CRSP_LIB/crsplib.a	CRSP C object library.
$CRSP_LIB/crsplib_f95.a	CRSP F95 object library.
$CRSP_INCLUDE	Directory containing CRSP FORTRAN header files referred to by INCLUDE statements.
$CRSP_SAMPLE	Directory containing CRSP sample programs.
$CRSP_MSTK	Directory containing monthly CRSP stock and index databases.
$CRSP_DSTK	Directory containing daily CRSP stock and index databases.
$CRSP_CST	Directory containing CRSP Link and COMPUSTAT database.
$CRSP_WORK	Directory created to hold user-generated files

Sun Fortran 95 8.2

Command line

```
> cp $CRSP_SAMPLE/stkitm_fsamp1.f90 .

> chmod 660 stkitm_fsamp1.f90
```

In Sun Sparc Solaris 2.9/5.9:

```
> f95 –w –xarch=v9 –ext_names=plain –I$CRSP_INCLUDE –KPIC -o stkitm_fsamp1
stkitm_fsamp1.f90 $CRSP_LIB/crsplib_f95.a  $CRSP_LIB/crsplib.a
```

In Sun x86 Solaris 2.9/5.9:

```
> f95 –w –xtarget=generic64 –ext_names=plain -I$CRSP_INCLUDE –KPIC -o stkitm_fsamp1
stkitm_fsamp1.f90  $CRSP_LIB/crsplib_f95.a $CRSP_LIB/crsplib.a
```

To run the program:

```
> ./stkitm_fsamp1
```

Sample programs can also be compiled and linked using the `make` utility. The sample program directory `$CRSP_SAMPLE` contains sample make description files for Sun Solaris in `f95_samp.mak`. To use make, copy the relevant description file to your program directory, edit it to support the program(s) of interest and create local executables.

Make file:

To compile a specific sample program:

```
> make –f f95_samp.mk stkitm_fsamp1
```

To compile all sample programs:

```
> make –f f95_samp.mk
```

To run the program:

```
> ./stkitm_fsamp1
```

Linux

CRSP currently supports Linux, Red Hat 7.2 (32-bit) and RHEL5 (64-bit) on Intel x86. FORTRAN was compiled and tested using the Lahey Fortran 95 Version 6.2 (32-bit) and the g95 Version 0.091 32- and 64-bit compilers. Fortran library functions interface with C functions in the CRSP object library. Ordinary sample Fortran usage links to the object library, but does not require compiling C programs.

CRSP access depends on environment variables set during installation. Environment variables can be used on Linux with the name preceded by the $ symbol. All file names and environment variable names are case sensitive on Linux systems. The env command can be used in a terminal window to find available environment variables.

Important CRSP files or directories can be found with the following names:

$CRSP_BIN	Directory containing executable programs and shell scripts files. This directory is in the PATH so programs can be run from any directory. Executable versions of the sample programs can be found in this directory.
$CRSP_LIB	Directory containing CRSP object library and internal files.
$CRSP_LIB/crsplib.a	CRSP object library.
$CRSP_LIB/crsplib_f95.a	CRSP F95 object library.
$CRSP_INCLUDE	Directory containing CRSP Fortran header files referred to by INCLUDE statements.
$CRSP_SAMPLE	Directory containing CRSP sample programs.
$CRSP_MSTK	Directory containing monthly CRSP stock and index databases.
$CRSP_DSTK	Directory containing daily CRSP stock and index databases.
$CRSP_CST	Directory containing CRSP Link and COMPUSTAT database.
$CRSP_WORK	Directory created to hold user-generated files

Following is an example of modifying and running a sample FORTRAN program:

Lahey Fortran 95 Ver. 6.2 (32-bit)

Command line

```
> cp $CRSP_SAMPLE/stkitm_fsamp1.f90 .

> chmod 660 stkitm_fsamp1.f90

> lf95 –w –I$CRSP_INCLUDE stkitm_fsamp1.f90 –o stkitm_fsamp1 $CRSP_LIB/crsplib.a
$CRSP_LIB/crsplib_f95.a –lm
```

To run the program:

```
> ./stkitm_fsamp1
```

Using a Make File

Sample programs can also be compiled and linked using the `make` utility. The sample program directory `$CRSP_SAMPLE` contains sample make description files for Linux in `f95_samp.mak` for the Lahey compiler. To use the make file, copy the relevant description file to your program directory, and edit it to support the program(s) of interest and create local executables.

To compile a specific sample program:

```
> make –f f95_samp.mk stkitm_fsamp1
```

To compile all sample programs:

```
> make –f f95_samp.mk
```

To run the program:

```
> ./stkitm_fsamp1
```

G95 Ver. 0.91 32- and 64-bit

Command line:

```
> cp $CRSP_SAMPLE/stkitm_fsamp1.f90 .

> chmod 660 stkitm_fsamp1.f90

> g95 –o stkitm_fsamp1 –w stkitm_fsamp1.f90 –I$CRSP_INCLUDE $CRSP_LIB/crsplib.a
$CRSP_LIB/crsplib_f95.a `find /usr/local -name libf95.a 2>&1 | grep libf95\.a –lm
```

To run the program:

```
> ./stkitm_fsamp1
```

Using a Make File:

The sample program directory $CRSP_SAMPLE contains sample make description files for Linux in f95_samp.mkg5 for the g95 compiler. To use the make file, copy the relevant description file to your program directory, and edit it to support the program(s) of interest and create local executables.

To compile specific sample program

```
> make –f f95_samp.mkg5 stkitm_fsamp1
```

To compile all sample programs:

```
> make –f f95_samp.mkg5
```

To run the program:

```
> ./stkitm_fsamp1
```

Using the CRSP Fortran 95 API

When you have ascertained that you can successfully compile and execute the provided sample programs, you are ready to begin creating your own programs. This section illustrates the general flow of CRSP API programs and discusses the data objects you will use when accessing CRSP Databases from Fortran 95.

Sample API Program Flow

CRSP Fortran 95 API client applications are structured according to the following steps. The code snippets shown below are excerpted from the stkitm_fsamp4.f90 sample program included with the API.

1. Include a 'use' statement for the `crsp_f_itm_lib` module. This step makes API functions and constants available from within your program.

   ```
   use crsp_f_itm_lib
   ```

2. Declare the item-access handle:

   ```
   type(CRSP_ITM_HNDL_T) :: hndl
   ```

3. Declare pointers to item objects to be used in your program:

   ```
   type(CRSP_ITM_T),pointer :: stkhdr_itm => null(),  &
                               prc_itm => null(),         &
                               adjprc_itm => null(),      &
                               adjshr_itm => null(),      &
                               adjvol_itm => null()
   ```

4. Connect to a CRSP database for item-access. Specify database root and the available item-set app_id. :

```
if (crsp_f_itm_init(hndl,dbpath,stkappid,'stk1') == CRSP_FAIL) then
       !!--error
       print *,'Error - failed to connect to db:',TRIM(dbpath)
       stop
endif
```

5. Select the wanted items to be loaded into active item set. Multiple calls to crsp_f_itm_load are allowed and have expanding effect on the item selection, while without the selected item duplication.

```
sts=crsp_f_itm_load(hndl,'STKHDR_ALL',CRSP_MATCH_IGNORE)
if ( sts == CRSP_FAIL .or. sts == CRSP_NOT_FOUND) then
       !!--error
       print *,'Error - failed to load the requested data items (DSTK:1)'
       stop
endif

sts=crsp_f_itm_load(hndl,'DSTK_TS;ADJPRC;ADJSHR;ADJVOL',CRSP_MATCH_IGNORE)
if ( sts == CRSP_FAIL .or. sts == CRSP_NOT_FOUND) then
       !!--error
       print *,'Error - failed to load the requested data items (DSTK:2)'
       stop
endif
```

6. Configure the item-access if needed by setting any of the access configuration codes in the access handle, eg:

```
hndl%fiscal_disp_cd = 'F'
```

See *CRSP Fortran 95 API Data Objects* below for the details of access handle properties.

7. Open the item-access to the selected CRSP dataset:

```
if (crsp_f_itm_open(hndl) == CRSP_FAIL) then
       !!--error
       print *,'Error - failed to open db for access'
       stop
endif
```

8. Attach the user-declared item pointers to the loaded items. Specify the item name and keyset:

```
if (crsp_f_itm_find(hndl,'HEADER',0,stkhdr_itm) == CRSP_FAIL          &
       .or. crsp_f_itm_find(hndl,'PRC',0,prc_itm) == CRSP_FAIL        &
       .or. crsp_f_itm_find(hndl,'ADJPRC',0,adjprc_itm) == CRSP_FAIL  &
       .or. crsp_f_itm_find(hndl,'ADJSHR',0,adjshr_itm) == CRSP_FAIL  &
       .or. crsp_f_itm_find(hndl,'ADJVOL',0,adjvol_itm) == CRSP_FAIL  &
       .or.  .not. associated(stkhdr_itm)  &
       .or.  .not. associated(prc_itm)     &
       .or.  .not. associated(adjprc_itm)  &
       .or.  .not. associated(adjshr_itm)  &
       .or.  .not. associated(adjvol_itm)) then
       !!--error
       print *,'Error - invalid item/keyset specified'
       stop
endif
```

Note: Only previously loaded items can be found. If an item is not being found, first make sure the requested item has been loaded in the requested keyset explicitly or implicitly through a collective item group.

9. Load the primary access key if different from a defined default.

```
if (crsp_f_itm_load_key(hndl,'permno') == CRSP_FAIL) then
       !!--error
       print *,'Error - failed to load index for key:','permno'
       stop
endif
```

10. In case of direct access, set the corresponding key item to the target value.

```
if (crsp_f_itm_set_key(hndl,'KYPERMNO',key) == CRSP_FAIL) then
       !!-- error
       print *,'Error - failed to set key:',key
       stop
endif
```

11. Read the database. Specify the key matching mode when exact key value is not found. In case of sequential access set key flag to CRSP_NEXT. For composite primary key the non-zero key_status signals access on detail key:

```
sts = crsp_f_itm_read(hndl,CRSP_EXACT, key_sts)
if ( sts == CRSP_FAIL) then
       got_db_error = .true.
       print *,'Error - failed to read db for key:',key
       exit
endif
```

12. On successful read-status the data is loaded and item data containers are ready for access. The item data can be accessed through the attached item pointers. This step is where your application-specific logic comes into play.

```
do i=prc_itm%obj%ts%beg, prc_itm%obj%ts%end
      write(ofunit,601)                          &
            stkhdr_itm%arr%header_val%permno,    &
            stkhdr_itm%arr%header_val%hcomnam,   &
            prc_itm%obj%ts%cal%caldt(i),         &
            prc_itm%arr%flt_arr(i),              &
            adjprc_itm%arr%flt_arr(i),           &
            adjshr_itm%arr%int_arr(i),           &
            adjvol_itm%arr%dbl_arr(i)

601         format(I6,1X,A32,1X,I8,1X,F12.5,1X,F12.5,1X,I9,1X,F13.1)
enddo
```

13. Close the item-access and disconnect from the selected CRSP dataset. This also releases the internally allocated storage for this item-handle instance and invalidates any user-declared item pointers attached to the handle.

```
if (crsp_f_itm_close(hndl) == CRSP_FAIL) then
      !!--error
      print *, 'Error-- failed to close db:',TRIM(dbpath)
      stop
endif
```

For more detailed examples of item-access to supported CRSP database products, you are encouraged to refer to the supplied set of sample programs.

CRSP Fortran 95 API Data Objects

Access to CRSP databases is achieved through two principal objects: the **access handle** – of type `CRSP_ITM_HNDL_T`, and the **item** – of type `CRSP_ITM_T`. These two important object types can be seen in steps 2 and 3 of the sample program flow above.

CRSP_ITM_HNDL_T

The item-access handle object—`type(CRSP_ITM_HNDL_T)` —encapsulates the information required to establish and maintain a single item-access session to a given CRSP database. Additional access sessions (either to the same or to another CRSP database), concurrent in the same program, require a separate access handle object. All of the item objects available in the active session are grouped within the respective access handle.

The main properties of the access handle object are listed on the following table:

NAME	Fortran 95 TYPE	DESCRIPTION
keytype	character(LEN=CRSP_NAME_LEN)	Determines the keys used to select data in read functions. Supported keytypes for the application are included in the reference data. A default will be set.
keyset_disp_cd	character(LEN=CRSP_TYPE_LEN)	Determines whether keyset items are labeled by the keyset number (NUM), the keyset tag (TAG), or the expanded list of all items comprising the keyset (EXP). The default display is TAG.
fiscal_disp_cd	character(LEN=CRSP_TYPE_LEN)	Determines whether fiscal-based time series items are reported on a calendar basis (C) or a fiscal basis (F). The default it C.
curr_disp_cd	character(LEN=CRSP_TYPE_LEN)	Determines whether monetary values are reported in the currency reported by Compustat (REP) or in US Dollars (USD). The default currency display code is REP.
grp_fill_cd	character(LEN=CRSP_TYPE_LEN)	Determines whether group item lists are filled so that every selected item is included for every selected keyset (Y or N). The default is Yes (Y).
dataset	CRSP_ITM_SET_T	Pointer to descriptor of currently attached CRSP dataset; includes root info for the CRSP dataset.

In a user-program the access handle objects are normally declared and allocated directly then passed to Fortran 95 itm-API functions as a parameter. The function `crsp_f_itm_init` initializes the contents of the access handle and connects it to the specified CRSP database.

CRSP_ITM_T

The item object—`type(CRSP_ITM_T)`—represents a generic container for a single data item defined in a given CRSP database. It unifies the data types defined for each of the supported CRSP databases and allows uniform access to the associated CRSP data containers from your programs.

The main properties of the item object are listed in the following table:

NAME	Fortran 95 TYPE	DESCRIPTION
itm_name	character(LEN=CRSP_NAME_LEN)	name of the item from a CRSP dataset.
keyset	integer	number of the keyset defined in a CRSP dataset.
itm_info	CRSP_ITM_INFO_T	item metadata; includes description, default keyset, and stored data type.
obj	CRSP_ITM_OBJ_T	describes the underlying CRSP data-object.
arr	CRSP_ITM_OBJARR_T	describes the Fortran 95 container associated with the defined CRSP data-object.
itmkeyset	CRSP_ITM_KEYSET_T	describes the details of the keyset (when non-zero and loaded), including its number, name, tag, and array of composing items of same CRSP_ITM_T type.
itmcal	CRSP_ITM_CAL_T	for calendar-bound items, describes the details of the attached calendar, including its id, keyset ,frequency, and attached calendar object of CRSP_CAL_T type. When requested, the calendar may be 'shifted', based on the currently loaded company's FYE to attribute properly the item's period data.

Item objects are normally declared as Fortran 95 pointers and then attached to the actual defined item objects by calling the `crsp_f_itm_find` function for the given access handle and the specified item name and keyset.

CRSP_ITM_OBJ_T

Item data is accessed from the data-object `itm%obj` and associated to a Fortran 95 container `itm%arr`. The item data container object, `type(CRSP_ITM_OBJ_T)`, describes an instance of a CRSP data-object (time-series, array, row) that is defined for the specific item. Only a single data-object can be defined for a given item, which is identified by the `objtype` property.

Properties of the item data-object are listed in the following table:

NAME	Fortran 95 TYPE	DESCRIPTION
objtype	integer	type of the defined and allocated object: • CRSP_TS_OTID: CRSP time-series • CRSP_ARRAY_OTID: CRSP array • CRSP_ROW_OTID: CRSP row
ts	CRSP_TS_T	pointer to allocated CRSP time-series data-object.
arr	CRSP_ARRAY_T	pointer to allocated CRSP array data-object.
row	CRSP_ROW_T	pointer to allocated CRSP row data-object.
is_empty	logical	indicates whether the allocated CRSP data-object contains no data.

The item data-object normally has an associated Fortran 95 container, which is either Fortran 95 array or scalar of the data type corresponding to the actual stored data, as identified by `arrtype` property. Details of the CRSP container objects types are listed in the reference section *CRSP Container Objects*.

CRSP_ITM_OBJARR_T

The item data array, `type(CRSP_ITM_OBJARR_T)`, describes the associated Fortran 95 container object. The Fortran 95 container is allocated based on the object's type (`objtype`) and contained data type (`arrtype`). The respective scalar member has suffix `_val` to its name, and `_arr` for the array type. Time-series and array data are stored in array type, while row data is kept in scalar type:

- `itm%arr%arrtype`:

  `CRSP_TS_OTID: itm%arr%<arrtype_name>_arr` - time-series object data array

  `CRSP_ARARY_OTID: itm%arr%<arrtype_name>_arr` - array object data array

  `CRSP_ROW_OTID: itm%arr%<arrtype_name>_val` - row object data scalar.

NOTE: throughout the implementation of the CRSP Fortran 95 API, the Fortran 95 array indexing is **0-based**, thus the first element of an array is `data_arr(0)`.

Properties of the item data array for the item data types that are common to all of the supported CRSP datasets are listed in the following table:

NAME	Fortran 95 TYPE	DESCRIPTION
arrtype	integer	data type of the defined and allocated Fortran 95 container. Common data types: • CRSP_INTEGER_TID: integer • CRSP_FLOAT_TID: real • CRSP_DOUBLE_TID: double precision • CRSP_CHAR_TID: character(1) • CRSP_CHARACTER_TID: CRSP_VARSTRING_T type
int_val / arr	integer / dimension (:)	pointer to allocated scalar / array of integer type
flt_val / arr	real / dimension (:)	pointer to allocated scalar / array of real type
dbl_val / arr	double precision / dimension (:)	pointer to allocated scalar / array of double precision type
char_val / arr	character(LEN=1) / dimension (:)	pointer to allocated scalar / array of single-character type
vstr_val / arr	CRSP_VARSTRING_T / dimension (:)	pointer to allocated scalar / array of variable-length string type
structured types specific to CRSP datasets	Refer to the description of data types for the specific CRSP dataset.	

In a user-program the item container data is usually accessed directly as defined by item's data type, eg:

```
print *, sale_itm%arr%dbl_arr(i)
```

The item data container is normally accessed in association with its item data-object.

NOTE: If an incorrect data container is referenced, an access violation error should occur on the referenced null-pointer. In such situations the recommended action is to verify that the appropriate containers are being accessed for the selected items.

Data for items of CRSP array type is accessed in the valid `[0..num-1]` index range, as defined in the corresponding `arr` data-object. For example:

```
itm%arr%dbl_arr(i), i=0..itm%obj%arr%num-1
```

Data for items of CRSP time-series type is accessed in the valid `[beg,end]` index range, as defined in the corresponding `ts` data-object, e.g.:

```
itm%arr%dbl_arr(i), i=itm%obj%ts%beg..itm%obj%ts%end
```

Data for items of CRSP row type is not indexed and is accessed directly from the value as defined by the corresponding scalar/structured type, e.g.:

```
itm%arr%master_val%ccmid
```

To verify if an element of an array item contains a missing value, call the function `crsp_f_itm_is_miss_arrval`.

Supporting Information

Various supporting information about CRSP databases, items, keysets and other item-access objects is stored in the following derived types:

Fortran 95 TYPE NAME	DESCRIPTION	ACCESS VIA TYPE NAME	USAGE
CRSP_ITM_INFO_T	Item information; includes item's full name, description, display format, data type and size information. Also includes the default keyset number associated with this item.	CRSP_ITM_T	itm%itm_info
CRSP_ITM_KEYSET_T	Keyset descriptor; includes keyset information and the array of items composing the keyset.	CRSP_ITM_T	itm%itmkeyset
CRSP_ITM_CAL_T	Calendar descriptor; includes calendar's id, associated keyset number, base calendar name, and calendar's frequency, also the base calendar object. Additionally, for fiscal calendars indicates whether the calendar has been shifted based on the currently loaded company's FYE.	CRSP_ITM_T	itm%itmcal
CRSP_KEYSET_T	Keyset information; includes keyset's number, name, tag, and description. Indicates whether the keyset has been loaded and associated with any of the requested items.	CRSP_ITM_KEYSET_T	itmkeyset%keyset_info
CRSP_ITM_SET_T	CRSP data set descriptor; includes the set's path, name, id, and database root information.	CRSP_ITM_HNDL_T	hndl%dataset
CRSP_ROOT_INFO_T	CRSP data set root information; includes internal service information about the currently loaded database such as creation/modification date, product code and name, and descriptors of available calendars. Mainly intended for internal use.	CRSP_ITM_SET_T	dataset%root_info

NOTE: While selected supported information is populated on initiating of the connection to a CRSP data set (on return from call to `crsp_f_itm_init`), the listed supported information becomes available only on opening of the CRSP data set (on return from call to `crsp_f_itm_open`).

The relevant details of the derived types shown above are listed in the *Supporting Types* reference section.

Generic Data Types

All CRSP databases contain data items of both simple Fortran 95 data types and of database-specific structured data types. Moreover, each composing field of the structured data type can instead be requested as an individual data item of the simple Fortran 95 data type.

The vast majority of the data items defined in CRSP datasets are of CRSP time-series container object type, with the stored values commonly of generic Fortran 95 data types. A limited set of items is stored in CRSP array and CRSP row container objects; these items are mostly of structured data types and are listed in the following sections regarding particular CRSP database products.

The following table lists the supported generic data types and ways to access data from the item-associated container:

ITEM OBJECT TYPE	Fortran 95 TYPE NAME	INTERNAL STORAGE	ACCESS VIA CRSP_ITM_T
time-series	**CRSP_TS_T**		**itm%obj%ts**
	integer	int(4)	Itm%arr%int_arr(i)
	real	float(4)	Itm%arr%flt_arr(i)
	double precision	double(8)	Itm%arr%dbl_arr(i)
	character(LEN=1)	char (1)	Itm%arr%char_arr(i)
	CRSP_VARSTRING_T	char(n)	Itm%arr%vstr_arr(i)
array	**CRSP_ARRAY_T**		**itm%obj%arr**
	integer	int(4)	Itm%arr%int_arr(i)
	real	float(4)	Itm%arr%flt_arr(i)
	double precision	double(8)	Itm%arr%dbl_arr(i)
	character(LEN=1)	char (1)	Itm%arr%char_arr(i)
	CRSP_VARSTRING_T	char(n)	Itm%arr%vstr_arr(i)
row	**CRSP_ROW_T**		**itm%obj%row**
	integer	int(4)	Itm%arr%int_val
	real	float(4)	Itm%arr%flt_val
	double precision	double(8)	Itm%arr%dbl_val
	character(LEN=1)	char (1)	Itm%arr%char_val
	CRSP_VARSTRING_T	char(n)	Itm%arr%vstr_val

NOTE: The derived type CRSP_VARSTRING_T accommodates varying-length character strings and is used in the context of the CRSP Fortran 95 API to store data of individual character items that are composing fields of a structured data item. For example, the CCM structured item COMPANY has a field for company name, which can be referenced indirectly as `company_itm%arr%company_val%conm` as a fixed-length character string. Alternatively, this field can be requested individually as the CONM item and then referenced as `conm_itm%arr%vstr_val` as varying-length string.

See the description of the CRSP_VARSTRING_T type at the end of this guide for usage information.

Accessing CRSP Databases

The following sections describe the details of accessing CRSP databases supported by the API. Supported databases are the CRSP US Stock Database, the CRSP US Index Database, and the CRSP/Compustat Merged Database. Each section presents database connection information, available access keys, as well how to access a database's data groups and items from your programs.

CRSP US Stock Database

To connect to the specific CRSP Stock database instance the path to its database root should be specified. When installed on your system, CRSP Stock data set will be assigned an environment variable pointing to the CRSP Stock database root.

Additionally, an application ID should be specified on the call to `crsp_f_itm_init` to indicate the item-universe to be loaded for the session and describes the available items and item groups, eg:

```
sts = crsp_f_itm_init (hndl,'CRSP_DSTK',app_id,'stk1')
```

User-programs should access the CRSP Stock data set with the `app_id` as listed in the following table:

STK ROOT/ APP ID	**Fortran 95 TYPE**	**DESCRIPTION**
CRSP_DSTK		**CRSP Daily Stock data set**
CRSP_DSTKITM_ID	integer	CRSP Daily Stock data items and groups
CRSP_MSTK		**CRSP Monthly Stock data set**
CRSP_MSTKITM_ID	integer	CRSP Monthly Stock data items and groups

The details on included items and item groups can be found starting on page 37.

Access Keys

CRSP Stock data set contains various data on companies and securities. Access key is composed of access key items the values of which can be retrieved or set from the user-program to control the direct access to STK data.

Default access key is loaded automatically on opening the access session to the CRSP STK data set

Additionally, a set of alternative access keys (and associated key items) is defined to facilitate access to the data by CRSP PERMNO, CUSIP, and other keys.

The current key universe can be retrieved by requesting header information and sequentially traversing the whole data set on the selected access key. The function `crsp_f_itm_get_key` can also be used to retrieve the value of the access key items for the currently read record.

To switch to access by an alternative key, a user calls `crsp_f_itm_load_key` to set the access key index, followed by calls to `crsp_itm_set_key` to set the value of the key items used on subsequent reading of the database.

The defined STK access keys and associated key items are listed in the following table:

CCM ACCESS KEY/ KEY ITEMS	**Fortran 95 TYPE**	**DESCRIPTION**	**NOTES**
PERMNO		**CRSP historical PERMNO**	**default**
KYPERMNO	Integer	CRSP company issue's PERMNO	primary key item
PERMCO		**CRSP historical PERMCO**	
KYPERMCO	Integer	CRSP company's PERMCO	primary key item
CUSIP		**CRSP Stock CUSIP**	
KYCUSIP	char(CRSP_CUSIP_LEN)	CRSP Stock issue's CUSIP	primary key item
HCUSIP		**CRSP Stock Hisorical CUSIP**	
KYHCUSIP	char(CRSP_CUSIP_LEN)	CRSP issue's Historical CUSIP	primary key item
Ticker		**CRSP Stock ticker**	
KYTICKER	char(CRSP_STK_TIC_LEN)	CRSP issue's ticker	primary key item
SICCD		**CRSP Stock SIC code**	
KYSIC	Integer	CRSP Stock security's SIC	primary key item

Data Types

Generally, individual CRSP Stock database data items are of common simple Fortran 95 data types and stored data can be accessed through `itm%arr` and corresponding scalar or array member.

Additionally, selected CRSP supplemental STK data groups can be accessed by the entire group as a defined structured type rather than as a stand-alone item. These data groups and their elements can both be accessed by `itm_name`, but recommended programming access is through the `itm_name` of the structure. To access the structured type and its fields, load the structured type `itm_name` during initialization, create a `CRSP_ITM_T` pointer matching the `itm_name`, attach it to the data, and access the structured type and its fields through the pointer:

```
sts = crsp_f_itm_load(hndl,'HEADER',match_flag)

sts = crsp_f_itm_find(hndl,'HEADER',0,header_itm)

permno = header_itm%arr%header_val%permno

...
```

Structured Types for CRSP US Stock Database Access

The tables below show the data groups available as STK structured types and their usage through the CRSP_ITM_T type.

STK MNEMONIC		DESCRIPTION	Fortran 95 TYPE NAME	OBJECT TYPE	OBJECT ACCESS VIA CRSP_ITM_T
DAILY	**MONTHLY**				
HEADER	MHEADER	Issue header information	CRSP_STK_HEADER_T	row	header_itm%obj%row
NAMES	MNAMES	Name information snapshot	CRSP_STK_NAME_T	array	names_itm%obj%arr
DISTS	MDISTS	Information for a distribution event	CRSP_STK_DIST_T	array	dists_itm%obj%arr
SHARES	MSHARES	Shares outstanding snapshot	CRSP_STK_SHARE_T	array	shares_itm%obj%arr
DELIST	MDELIST	Delisting information	CRSP_STK_DELIST_T	array	delist_itm%obj%arr
NASDIN	MNASDIN	Snapshot of Nasdaq information	CRSP_STK_NASDIN_T	array	nasdin_itm%obj%arr
PORTF	MPORTF	Portfolio statistic and assignment snapshot	CRSP_STK_PORT_T	array	portf_itm%obj%arr
GROUP	MGROUP	Group statistic and assignment snapshot	CRSP_STK_GROUP_T	array	group_itm%obj%arr

(M)HEADER

STK MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DAILY	**MONTHLY**				
HEADER	**MHEADER**	**Issue header information**	**CRSP_STK_HEADER_T**		**header_itm%arr%header_val**
PERMNO	MPERMNO	PERMNO	int(4)	I6	header_val%permno
PERMCO	MPERMCO	PERMCO	int(4)	I6	header_val%permco
COMPNO	MCOMPNO	NASDAQ Company Number	int(4)	I8	header_val%compno
ISSUNO	MISSUNO	NASDAQ Issue Number	int(4)	I8	header_val%issuno
HEXCD	MHEXCD	Exchange Code - Header	int(4)	I2	header_val%hexcd
HSHRCD	MHSHRCD	Share Code - Header	int(4)	I3	header_val%hshrcd
HNAMECD	MHNAMECD	Name Code -Header	int(4)	I4	header_val%hnamecd
HSICCD	MHSICCD	Standard Industrial Classification (SIC) Code - Header	int(4)	I2	header_val%hsiccd
BEGDT	MBEGDT	Begin of Stock Data	int(4)	I8	header_val%begdt
ENDDT	MENDDT	End of Stock Data	int(4)	I8	header_val%enddt

STK MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
HDLSTCD	MHDLSTCD	Delisting Code -Header	int(4)	I3	header_val%dlstcd
CUSIP	MCUSIP	CUSIP - Header	char(16)	A8	header_val%hcusip
HTICK	MHTICK	Ticker Symbol - Header	char(16)	A6	header_val%htick
HNAICS	MHNAICS	North American Industry Classification System (NAICS) - Header	char(8)	A7	header_val%hnaics
HCOMNAM	MHCOMNAM	Company Name - Header	char(36)	A36	header_val%hcomnam
HTSYMBOL	MHTSYMBOL	Trading Ticker Symbol - Header	char(12)	A12	header_val%htsymbol
HCNTRYCD	MHCNTRYCD	Country Code - Header	char(4)	A3	header_val%hcntrycd
HPRIMEXCH	MHPRIMEXCH	Primary Exchange - Header	char(1)	A1	header_val%hprimexch
HSUBEXCH	MHSUBEXCH	Sub-Exchange - Header	char(1)	A1	header_val%hsubexch
HTRDSTAT	MHTRDSTAT	Trading Status - Header	char(1)	A1	header_val%htrdstat
HSECSTAT	MHSECSTAT	Security Status - Header	char(1)	A1	header_val%hsecstat
HSHRTYPE	MHSHRTYPE	Share Type - Header	char(1)	A1	header_val%hshrtype
HISSUERCD	MHISSUERCD	Issuer Code -Header	char(1)	A1	header_val%hissuercd
HINCCD	MHINCCD	Incorporation Code -Header	char(1)	A1	header_val%hinccd
HITS	MHITS	Intermarket Trading System Indicator - Header	char(1)	A1	header_val%hits
HDENOM	MHDENOM	Trading denomination	char(1)	A1	header_arr(i)%hdenom
HELIGCD	MHELIGCD	Eligibility code	char(1)	A1	header_arr(i)%heligcd
HCONVCD	MHCONVCD	Convertible code	char(1)	A1	header_arr(i)%hconvcd
HNAMEFLAG	MHNAMEFLAG	Name flag	char(1)	A1	header_arr(i)%hnameflag
HNAMEDESC	MHNAMEDESC	Name description	char(24)	A15	header_arr(i)%hnamedesc
HRATING	MHRATING	Rating (if applicable) or strike price	float(4)	F9.4	header_arr(i)%hrating
HEXPDT	MHEXPDT	Expiration date	int(4)	I8	header_arr(i)%hexpdt

(M)NAMES

STK MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DAILY	**MONTHLY**				
NAMES	**MNAMES**	**Name information snapshot**	**CRSP_STK_NAME_T**		**names_itm%arr%names_arr**
UOT	MUOT	Unit of Trade, End of Period - actual	int(4)	I6	names_arr(i)%uot
NAMEDT	MNAMEDT	Names Information Begin Date	int(4)	I8	names_arr(i)%namedt
NAMEENDDT	MNAMEENDDT	Names Information End Date	int(4)	I8	names_arr(i)%nameenddt
SHRCD	MSHRCD	Share Code	int(4)	I2	names_arr(i)%shrcd
NAMECD	MNAMECD	Name Code, End of Period - actual	int(4)	I3	names_arr(i)%namecd
EXCHCD	MEXCHCD	Exchange Code	int(4)	I2	names_arr(i)%exchcd
SICCD	MSICCD	Standard Industrial Classification (SIC) Code	int(4)	I4	names_arr(i)%siccd
NCUSIP	MNCUSIP	CUSIP	char(16)	A8	names_arr(i)%ncusip
TICKER	MTICKER	Ticker Symbol	char(8)	A5	names_arr(i)%ticker
SNAICS	MSNAICS	North American Industry Classification System (NAICS)	char(8)	A7	names_arr(i)%snaics
COMNAM	MCOMNAM	Company Name	char(36)	A32	names_arr(i)%comnam
TSYMBOL	MTSYMBOL	Trading Ticker Symbol	char(12)	A10	names_arr(i)%tsymbol
CNTRYCD	MCNTRYCD	Country Code, End of Period - actual	char(4)	A3	names_arr(i)%cntrycd
PRIMEXCH	MPRIMEXCH	Primary Exchange	char(1)	A1	names_arr(i)%primexch
SUBEXCH	MSUBEXCH	Sub-Exchange	char(1)	A1	names_arr(i)%subexch
TRDSTAT	MTRDSTAT	Trading Status	char(1)	A1	names_arr(i)%trdstat
SECSTAT	MSECSTAT	Security Status	char(1)	A1	names_arr(i)%secstat
SHRTYPE	MSHRTYPE	Share Type, End of Period - actual	char(1)	A1	names_arr(i)%shrtype
ISSUERCD	MISSUERCD	Issuer Code, End of Period - actual	char(1)	A1	names_arr(i)%issuercd
INCCD	MINCCD	Incorporation Code, End of Period - actual	char(1)	A1	names_arr(i)%inccd
ITS	MITS	Intermarket Trading System, End of Period - actual	char(1)	A1	names_arr(i)%its
DENOM	MDENOM	Trading Denomination, End of Period - actual	char(1)	A1	names_arr(i)%denom
ELIGCD	MELIGCD	Eligibility Code, End of Period - actual	char(1)	A1	names_arr(i)%eligcd
CONVCD	MCONVCD	Convertible Code, End of Period - actual	char(1)	A1	names_arr(i)%convcd
NAMEFLAG	MNAMEFLAG	Name Flag, End of Period - actual	char(1)	A1	names_arr(i)%nameflag
SHRCLS	MSHRCLS	Share Class	char(4)	A1	names_arr(i)%shrcls

STK MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
NAMEDESC	MNAMEDESC	Name Description, End of Period - actual	char(24)	A15	names_arr(i)%namedesc
RATING	MRATING	Interest Rate, End of Period - actual	float(4)	F9.4	names_arr(i)%rating
EXPDT	MEXPDT	Expiration Date, End of Period - actual	int(4)	I8	names_arr(i)%expdt

(M)DISTS

STK MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DAILY	**MONTHLY**				
DISTS	**MDISTS**	**Information for a distribution event**	**CRSP_STK_DIST_T**		**dists_itm%arr%dists_arr**
DISTCD	MDISTCD	Distribution Code	int(4)	I4	dists_arr(i)%distcd
DIVAMT	MDIVAMT	Dividend Amount	float(4)	F9.5	dists_arr(i)%divamt
FACPR	MFACPR	Factor to Adjust Price in Period	float(4)	F8.4	dists_arr(i)%facpr
FACSHR	MFACSHR	Factor to Adjust Shares Outstanding	float(4)	F8.4	dists_arr(i)%facshr
DCLRDT	MDCLRDT	Distribution Declaration Date	int(4)	I8	dists_arr(i)%dclrdt
EXDT	MEXDT	Ex-Distribution Date	int(4)	I8	dists_arr(i)%exdt
RCRDDT	MRCRDDT	Record Date	int(4)	I8	dists_arr(i)%rcrddt
PAYDT	MPAYDT	Payment Date	int(4)	I8	dists_arr(i)%paydt
ACPERM	MACPERM	Acquiring PERMNO	int(4)	I5	dists_arr(i)%acperm
ACCOMP	MACCOMP	Acquiring PERMCO	int(4)	I5	dists_arr(i)%accomp

(M)SHARES

STK MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DAILY	**MONTHLY**				
SHARES	**MSHARES**	**Shares outstanding snapshot**	**CRSP_STK_SHARE_T**		**shares_itm%arr%shares_arr**
SHROUT	MSHROUT	Shares Outstanding	int(4)	I10	shares_arr(i)%shrout
SHRSDT	MSHRSDT	Shares Outstanding Observation Date	int(4)	I8	shares_arr(i)%shrsdt
SHRSENDDT	MSHRSENDDT	Shares Outstanding Observation End Date	int(4)	I8	shares_arr(i)%shrsenddt
SHRFLG	MSHRFLG	Shares Outstanding Observation Flag	int(4)	I4	shares_arr(i)%shrflg

(M)DELIST

STK MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DAILY	**MONTHLY**				
DELIST	**MDELIST**	**Delisting information**	**CRSP_STK_DELIST_T**		**delist_itm%arr%delist_arr**
DLSTDT	MDLSTDT	Delisting Date	int(4)	I8	delist_arr(i)%dlstdt
DLSTCD	MDLSTCD	Delisting Code	int(4)	I3	delist_arr(i)%dlstcd
NWPERM	MNWPERM	Linked PERMNO After Delisting	int(4)	I8	delist_arr(i)%nwperm
NWCOMP	MNWCOMP	Linked PERMCO After Delisting	int(4)	I8	delist_arr(i)%nwcomp
NEXTDT	MNEXTDT	Date of Next Available Information	int(4)	I8	delist_arr(i)%nextdt
DLAMT	MDLAMT	Total Amount Used in Delisting Return	float(4)	F13.5	delist_arr(i)%dlamt
DLRETX	MDLRETX	Delisting Return without Dividends	float(4)	F11.6	delist_arr(i)%dlretx
DLPRC	MDLPRC	Delisting Price	float(4)	F13.5	delist_arr(i)%dlprc
DLPDT	MDLPDT	Delisting Payment Date	int(4)	I8	delist_arr(i)%dlpdt
DLRET	MDLRET	Delisting Return	float(4)	F11.6	delist_arr(i)%dlret

(M)NASDIN

STK MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DAILY	**MONTHLY**				
NASDIN	**MNASDIN**	**Snapshot of Nasdaq information**	**CRSP_STK_NASDIN_T**		**nasdin_itm%arr%nasdin_arr**
TRTSCD	MTRTSCD	NASDAQ Status Code, End of Period	int(4)	I2	nasdin_arr(i)%trtscd
TRTSDT	MTRTSDT	Beginning Effective Date of Traits	int(4)	I8	nasdin_arr(i)%trtsdt
TRTSENDDT	MTRTSENDDT	Last Effective Date of Traits	int(4)	I8	nasdin_arr(i)%trtsenddt
NMSIND	MNMSIND	NASDAQ National Market Indicator	int(4)	I2	nasdin_arr(i)%nmsind
MMCNT	MMMCNT	NASDAQ Market Makers Count	int(4)	I4	nasdin_arr(i)%mmcnt
NSDINX	MNSDINX	NASDAQ Index Code	int(4)	I2	nasdin_arr(i)%nsdinx

(M)PORTF

STK MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DAILY	**MONTHLY**				
PORTF	**MPORTF**	**Portfolio statistic and assignment snapshot**	**CRSP_STK_PORT_T**		**port_itm%arr%port_arr**
PORT	MPORT	Portfolio Assignment	int(4)	I4	portf_arr(i)%port
STAT	MSTAT	Portfolio Statistic Value	double(8)	F16.5	portf_arr(i)%stat

(M)GROUP

STK MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DAILY	**MONTHLY**				
GROUP	**MGROUP**	**Group statistic and assignment snapshot**	**CRSP_STK_GROUP_T**		**group_itm%arr%group_arr**
GRPDT	MGRPDT	Group Beginning Date	int(4)	I8	group_arr(i)%grpdt
GRPENDDT	MGRPENDDT	Group Ending Date	int(4)	I8	group_arr(i)%grpenddt
GRPFLAG	MGRPFLAG	Group Flag	int(4)	I4	group_arr(i)%grpflag
GRPSUBFLAG	MGRPSUBFLAG	Group Subflag	int(4)	I4	group_arr(i)%grpsubflag

CRSP US Index Database

To connect to the specific CRSP Index database instance the path to its database root should be specified. When installed on your system, CRSP Index data sets will be assigned an environment variable pointing to the CRSP Index database root.

Additionally, an application ID should be specified on the call to `crsp_f_itm_init` to indicate the item-universe to be loaded for the session and describes the available items and item groups, eg:

```
sts = crsp_f_itm_init (hndl,'CRSP_DSTK',app_id,'ind1')
```

User-programs should access the CRSP Index data sets with the `app_id` as listed in the following table:

STK ROOT/ APP ID	**Fortran 95 TYPE**	**DESCRIPTION**
CRSP_DSTK		**CRSP Daily Stock and Index data sets**
CRSP_DINDITEMS_ID	integer	CRSP Daily Index series data items and groups
CRSP_DINDGITEMS_ID	integer	CRSP Daily Index group data items and groups
CRSP_MSTK		**CRSP Monthly Stock and Index data sets**
CRSP_MINDITEMS_ID	integer	CRSP Monthly Index series data items and groups
CRSP_MINDGITEMS_ID	integer	CRSP Monthly Index group data items and groups

Access Keys

CRSP Index data sets includes various data on market indexes. Access key is composed of access key items the values of which can be retrieved or set from the user-program to control the direct access to IND data.

Default access key is loaded automatically on opening the access session to the CRSP IND data set

The current key universe can be retrieved by requesting header information and sequentially traversing the whole data set on the selected access key. The function `crsp_f_itm_get_key` can also be used to retrieve the value of the access key items for the currently read record.

The defined IND access keys and associated key items are listed in the following table:

CCM ACCESS KEY/ KEY ITEMS	**Fortran 95 TYPE**	**DESCRIPTION**	**NOTES**
indno		**CRSP Index's INDNO**	**default**
KYINDNO	integer	CRSP index's INDNO	primary key item

Data Types

Generally, individual CRSP Index database data items are of common simple Fortran 95 data types and stored data can be accessed through `itm%arr` and corresponding scalar or array member.

Additionally, selected CRSP supplemental IND data groups can be accessed by the entire group as a defined structured type rather than as a stand-alone item. These data groups and their elements can both be accessed by itm_name, but recommended programming access is through the itm_name of the structure. To access the structured type and its fields, load the structured type itm_name during initialization, create a CRSP_ITM_T pointer matching the itm_name, attach it to the data, and access the structured type and its fields through the pointer:

```
sts = crsp_f_itm_load(hndl,'INDHDR',match_flag)

sts = crsp_f_itm_find(hndl,'INDHDR',0,indhdr_itm)

indno = indhdr_itm%arr%indhdr_val%indno

...
```

Structured Types for CRSP US Index Database Access

The tables below show the data groups available as structured types and their usage through the CRSP_ITM_T type.

IND MNEMONIC		DESCRIPTION	Fortran 95 TYPE NAME	OBJECT TYPE	OBJECT ACCESS VIA CRSP_ITM_T
DAILY	**MONTHLY**				
INDHDR	MINDHDR	Index header information	CRSP_IND_HEADER_T	row	indhdr_itm%obj%row
REBAL	MREBAL	Index rebalancing period summary	CRSP_IND_REBAL_T	array	rebal_itm%obj%arr
LIST	MLIST	Issue list information	CRSP_IND_LIST_T	array	list_itm%obj%arr

(M)INDHDR

IND MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DAILY	**MONTHLY**				
INDHDR	**MINDHDR**	**Index header information**	**CRSP_IND_HEADER_T**		**indhdr_itm%arr%indhdr_val**
INDNO	MINDNO	specific index series	int(4)	I7	indhdr_val%indno
INDCO	MINDCO	major index series group	int(4)	I7	indhdr_val%indco
PRIMFLAG	MPRIMFLAG	0 if master or permno of master	int(4)	I6	indhdr_val%primflag
PORTNUM	MPORTNUM	portfolio number in master if subset	int(4)	I6	indhdr_val%portnum
INDNAME	MINDNAME	index name	char(80)	A79	indhdr_val%indname
GROUPNAME	MGROUPNAME	index group name	char(80)	A79	indhdr_val%groupname
METHCODE	MMETHCODE	code of possible methodology types	int(4)	I6	indhdr_val%method%methcode
PRIMTYPE	MPRIMTYPE	fractile, selected, rulebased, market	int(4)	I6	indhdr_val%method%primtype
SUBTYPE	MSUBTYPE	index type subcode	int(4)	I6	indhdr_val%method%subtype
WGTTYPE	MWGTTYPE	reweighting type flag	int(4)	I6	indhdr_val%method%wgttype
WGTFLAG	MWGTFLAG	reweighting timing flag	int(4)	I6	indhdr_val%method%wgtflag
FLAGCODE	MFLAGCODE	code of possible exception types	int(4)	I6	indhdr_val%flags%flagcode
ADDFLAG	MADDFLAG	handling new issues	int(4)	I6	indhdr_val%flags %addflag
DELFLAG	MDELFLAG	handling issues becoming ineligible	int(4)	I6	indhdr_val%flags%delflag
DELRETFLAG	MDELRETFLAG	measuring return of delisted issues	int(4)	I6	indhdr_val%flags%delretflag

IND MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
MISSFLAG	MMISSFLAG	handling missing prices	int(4)	I6	indhdr_val%flags%missflag
UUNIVCODE	MUUNIVCODE	code of possible universe/subset types (index universe)	int(4)	I6	indhdr_val%induniv%univcode
UBEGDT	MUBEGDT	beginning date of valid data (index universe)	int(4)	I8	indhdr_val%induniv%begdt
UENDDT	MUENDDT	beginning date of valid data (index universe)	int(4)	I8	indhdr_val%induniv%enddt
UWANTEXCH	MUWANTEXCH	valid exchange code (index universe)	int(4)	I6	indhdr_val%induniv%wantexch
UWANTNMS	MUWANTNMS	Nasdaq National Market classification (index universe)	int(4)	I6	indhdr_val%induniv%wantnms
UWANTWI	MUWANTWI	when-issued code (index universe)	int(4)	I6	indhdr_val%induniv%wantwi
UWANTINC	MUWANTINC	valid incorporation (index universe)	int(4)	I6	indhdr_val%induniv%wantinc
USCCODE	MUSCCODE	code of possible restriction types (index universe)	int(4)	I6	indhdr_val%induniv%sccode
UFSTDIG	MUFSTDIG	bitmap of first share code digit (index universe)	int(4)	I6	indhdr_val%induniv%fstdig
USECDIG	MUSECDIG	bitmap of second share code digit (index universe)	int(4)	I6	indhdr_val%induniv%secdig
PUNIVCODE	MPUNIVCODE	code of possible universe/subset types (partition universe)	int(4)	I6	indhdr_val%partuniv%univcode
PBEGDT	MPBEGDT	beginning date of valid data (partition universe)	int(4)	I8	indhdr_val%partuniv%begdt
PENDDT	MPENDDT	beginning date of valid data (partition universe)	int(4)	I8	indhdr_val%partuniv%enddt
PWANTEXCH	MPWANTEXCH	valid exchange code (partition universe)	int(4)	I6	indhdr_val%partuniv%wantexch
PWANTNMS	MPWANTNMS	Nasdaq National Market classification (partition universe)	int(4)	I6	indhdr_val%partuniv%wantnms
PWANTWI	MPWANTWI	when-issued code (partition universe)	int(4)	I6	indhdr_val%partuniv%wantwi
PWANTINC	MPWANTINC	valid incorporation (partition universe)	int(4)	I6	indhdr_val%partuniv%wantinc
PSCCODE	MPSCCODE	code of possible restriction types (partition universe)	int(4)	I6	indhdr_val%partuniv%sccode
PFSTDIG	MPFSTDIG	bitmap of first share code digit (partition universe)	int(4)	I6	indhdr_val%partuniv%fstdig
PSECDIG	MPSECDIG	bitmap of second share code digit (partition universe)	int(4)	I6	indhdr_val%partuniv%secdig
RULECODE	MRULECODE	code of possible assignment rule types	int(4)	I6	indhdr_val%rules%rulecode
BUYFNCT	MBUYFNCT	function code for buy rules	int(4)	I6	indhdr_val%rules%buyfnct
SELLFNCT	MSELLFNCT	function code for sell rules	int(4)	I6	indhdr_val%rulesl%sellfnct
STATFNCT	MSTATFNCT	function code for calculating statistic	int(4)	I6	indhdr_val%rules%statfnct
GROUPFLAG	MGROUPFLAG	how stats are grouped before applying rules	int(4)	I6	indhdr_val%rules%groupflag
ASSIGNCODE	MASSIGNCODE	code of possible assignment types	int(4)	I6	indhdr_val%assign%assigncode
ASPERM	MASPERM	permno of associated index for breakpoints	int(4)	I6	indhdr_val%assign%asperm
ASPORT	MASPORT	portfolio number in associated index	int(4)	I6	indhdr_val%assign%asport

IND MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
REBALCAL	MREBALCAL	calid of rebalancing calendar	int(4)	I6	indhdr_val%assign%rebalcal
ASSIGNCAL	MASSIGNCAL	calid of assignment calendar	int(4)	I6	indhdr_val%assign%assigncal
CALCCAL	MCALCCAL	calid of calculation range calendar	int(4)	I6	indhdr_val%assign%calccal

(M)REBAL

IND MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DAILY	MONTHLY				
REBAL	**MREBAL**	**Index rebalancing period summary**	**CRSP_IND_REBAL_T**		**rebal_itm%arr%rebal_arr**
RBBEGDT	MRBBEGDT	rebalancing beginning date	int(4)	I8	rebal_arr(i)%rbbegdt
RBENDDT	MRBENDDT	rebalancing ending date	int(4)	I8	rebal_arr(i)%rbenddt
RUSDCNT	MRUSDCNT	count used as of rebalancing	int(4)	I8	rebal_arr(i)%rusdcnt
MAXCNT	MMAXCNT	maximum count during period	int(4)	I8	rebal_arr(i)%maxcnt
RTOTCNT	MRTOTCNT	available count as of rebalancing	int(4)	I8	rebal_arr(i)%rtotcnt
ENDCNT	MENDCNT	count at end of period	int(4)	I8	rebal_arr(i)%endcnt
MINID	MMINID	identifier at minimum value	int(4)	I8	rebal_arr(i)%minid
MAXID	MMAXID	identifier at maximum value	int(4)	I8	rebal_arr(i)%maxid
MINSTAT	MMINSTAT	smallest statistic in period	double(8)	F14.3	rebal_arr(i)%minstat
MAXSTAT	MMAXSTAT	largest statistic in period	double(8)	F14.3	rebal_arr(i)%maxstat
MEDSTAT	MMEDSTAT	median statistic in period	double(8)	F14.3	rebal_arr(i)%medstat
AVGSTAT	MAVGSTAT	average statistic in period	double(8)	F14.3	rebal_arr(i)%avgstat

(M)LIST

IND MNEMONIC		FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
DAILY	MONTHLY				
LIST	**MLIST**	**Issue list information**	**CRSP_IND_LIST_T**		**list_itm%arr%list_arr**
TPERMNO	MTPERMNO	issue identifier	int(4)	I8	list_arr(i)%tpermno
LBEGDT	MLBEGDT	first date included	int(4)	I8	list_arr(i)%lbegdt

LENDDT	MLENDDT	last date included	int(4)	I8	list_arr(i)%lenddt
SUBIND	MSUBIND	code for subcategory of list	int(4)	I6	list_arr(i)%subind
LWEIGHT	MLWEIGHT	weight during range	double(8)	F16.5	list_arr(i)%lweight
TPERMNO	MTPERMNO	issue identifier	int(4)	I8	list_arr(i)%tpermno
LBEGDT	MLBEGDT	first date included	int(4)	I8	list_arr(i)%lbegdt
LENDDT	MLENDDT	last date included	int(4)	I8	list_arr(i)%lenddt
SUBIND	MSUBIND	code for subcategory of list	int(4)	I6	list_arr(i)%subind
LWEIGHT	MLWEIGHT	weight during range	double(8)	F16.5	list_arr(i)%lweight

CRSP/Compustat Merged Database

To connect to the specific CRSP CCM database instance the path to its database root should be specified. When installed on your system, CRSP CCM data set will be assigned an environment variable pointing to the CRSP CCM database root.

Additionally, an application ID should be specified on the call to `crsp_f_itm_init` to indicate the item-universe to be loaded for the session and describes the available items and item groups, eg:

```
sts = crsp_f_itm_init (hndl,'CRSP_CCM',app_id,'ccm1')
```

User-programs should access the CRSP CCM data set with the `app_id` as listed in the following table:

CCM ROOT/ APP ID	**Fortran 95 TYPE**	**DESCRIPTION**
CRSP_CCM		**CCM/CRSP Compustat data set**
CRSP_CCMITEMS_ID	integer	Compustat Xpressfeed data items and groups

The details on included items and item groups can be found starting on page 52.

Access Keys

CRSP Compustat Xpressfeed includes various data on companies, securities, and indexes. Access key is composed of access key items the values of which can be retrieved or set from the user-program to control the direct access to the CCM data.

Default access key for CRPS CCM is loaded automatically on opening the access session to the CRSP CCM data set

Additionally, a set of alternative access keys (and associated key items) is defined to facilitate access to the data by CRSP PERMNO,CUSIP and other keys.

The current key universe can be retrieved by requesting header information and sequentially traversing the whole data set on the selected access key. The function `crsp_f_itm_get_key` can also be used to retrieve the value of the access key items for the currently read record.

To switch to access by an alternative key, a user calls `crsp_f_itm_load_key` to set the access key index, followed by calls to `crsp_itm_set_key` to set the value of the key items used on subsequent reading of the database.

The defined CCM access keys and associated key items are listed in the following table:

CCM ACCESS KEY/ KEY ITEMS	Fortran 95 TYPE	DESCRIPTION	NOTES
gvkey		**Compustat GVKEY and IID**	**default**
KYGVKEY	integer	Compustat company's GVKEY	primary key item
KYIID	char(CRSP_CCM_IID_LEN)	Compustat company security's IID	secondary key item
gvkeyx		**Compustat permanent identifier for indexes**	
KYGVKEYX	integer	Compustat index's GVKEYX	primary key item
ccmid		**Compustat permanent identifier - either GVKEY or GVKEYX**	
KYCCMID	integer	CRSP CCMID (GVKEY or GVKEYX as reported in MASTER item)	primary key item
KYIID	char(CRSP_CCM_IID_LEN)	Compustat company security's IID	secondary key item
permco		**CRSP historical PERMCO Link**	
KYPERMCO	integer	CRSP company's PERMCO	primary key item
permno		**CRSP historical PERMNO Link**	
KYPERMNO	integer	CRSP company issue's PERMNO	primary key item
cusip		**Compustat CUSIP**	
KYCUSIP	char(CRSP_CCM_CUSIP_LEN)	Compustat issue's CUSIP	primary key item
ticker		**Compustat reported Issue Trading Symbol selects GVKEY and security**	
KYTICKER	char(CRSP_CCM_TIC_LEN)	Compustat issue's ticker	primary key item
sic		**Compustat -reported SIC code. Security or Company**	
KYSIC	integer	Compustat security's SIC	primary key item
KYIID	char(CRSP_CCM_IID_LEN)	Compustat company security's IID	secondary key item
apermno		**Link-Used PERMNO**	
KYPERMNO	integer	CRSP company issue's PERMNO	primary key item
apermco		**Link-Used PERMCO**	
KYPERMCO	integer	CRSP company issue's PERMCO	primary key item
ppermno		**CRSP PERMNO when security is marked as primary**	
KYPERMNO	integer	CRSP company issue's PERMNO	primary key item

Data Types

Generally, individual Compustat Xpressfeed data items are of common simple Fortran 95 data types and stored data can be accessed through `itm%arr` and corresponding scalar or array member.

Also additional character data types were introduced to store specific classes of Xpressfeed items, as listed in the following table:

ITEM OBJECT TYPE	Fortran 95 TYPE NAME	INTERNAL STORAGE	ACCESS VIA CRSP_ITM_T	NOTES
time-series	**CRSP_TS_T**		**itm%obj%ts**	
	CRSP_CCM_FTNT_T	char(CRSP_CCM_FTNT_LEN)	itm%arr%ftnt_arr(i)%ftnt	Used for CCM footnote items, mainly time-series
	CRSP_CCM_TEXTITEM_T	char(CRSP_CCM_TEXTITEM_LEN)	itm%arr%text_arr(i)%text	Used for various CCM character string items, mainly time-series
array	**CRSP_ARRAY_T**		**itm%obj%arr**	
	CRSP_CCM_FTNT_T	char(CRSP_CCM_FTNT_LEN)	itm%arr%ftnt_arr(i)%ftnt	
	CRSP_CCM_TEXTITEM_T	char(CRSP_CCM_TEXTITEM_LEN)	itm%arr%text_arr(i)%text	
row	**CRSP_ROW_T**		**itm%obj%row**	
	CRSP_CCM_FTNT_T	char(CRSP_CCM_FTNT_LEN)	itm%arr%ftnt_val%ftnt	
	CRSP_CCM_TEXTITEM_T	char(CRSP_CCM_TEXTITEM_LEN)	itm%arr%text_val%text	

Additionally, selected Compustat Xpressfeed primary data groups and CRSP supplemental data groups can be accessed by the entire group as a defined structured type rather than as a stand-alone item. These data groups and their elements can both be accessed by `itm_name`, but recommended programming access is through the `itm_name` of the structure. To access the structured type and its fields, load the structured type `itm_name` during initialization, create a `CRSP_ITM_T` pointer matching the `itm_name`, attach it to the data, and access the structured type and its fields through the pointer:

```
sts = crsp_f_itm_load(hndl,'MASTER',match_flag)

sts = crsp_f_itm_find(hndl,'MASTER',0,mstr_itm)

ccmid = mstr_itm%arr%master_val%ccmid

...
```

Structured Types for CRSP/Compustat Merged Database Access

The tables below show the data groups available as CCM structured types and their usage through the CRSP_ITM_T type.

CCM MNEMONIC	DESCRIPTION	Fortran 95 TYPE NAME	OBJECT TYPE	OBJECT ACCESS VIA CRSP_ITM_T
MASTER	CCM company id and range data	CRSP_CCM_MASTER_T	row	master_itm%obj%row
COMPANY	CCM company header information	CRSP_CCM_COMPANY_T	row	company_itm%obj%row
IDX_INDEX	CCM idx_index header information	CRSP_CCM_IDX_INDEX_T	row	idx_index_itm%obj%row
SPIND	S&P index header (pre-GICS)	CRSP_CCM_SPIND_T	row	spind_itm%obj%row
COMPHIST	CCM company header history	CRSP_CCM_COMPHIST_T	array	comphist_itm%obj%arr
CSTHIST	CST header history	CRSP_CST_NAME_T	array	csthist_itm%obj%arr
LINK	CRSP CCM link history	CRSP_CCM_LINK_T	array	link_itm%obj%arr
LINKUSED	CCM company CRSP link used data	CRSP_CCM_LINKUSED_T	array	linkused_itm%obj%arr
LINKRNG	CCM company CRSP link range data	CRSP_CCM_LINKRNG_T	array	linkused_itm%obj%arr
ADJFACT	CCM company adjustment factor history	CRSP_CCM_ADJFACT_T	array	adjfact_itm%obj%arr
HGIC	CCM company GICS code history	CRSP_CCM_HGIC_T	array	hgic_itm%obj%arr
OFFTITL	CCM company officer title data	CRSP_CCM_OFFTITL_T	array	offtitl_itm%obj%arr
CCM_FILEDATE	CCM company filing date data	CRSP_CCM_FILEDATE_T	array	ccm_filedate_itm%obj%arr
CCM_IPCD	CCM industry presentation code data	CRSP_CCM_IPCD_T	array	ccm_ipcd_itm%obj%arr
SECURITY	CCM security header information	CRSP_CCM_SECURITY_T	row	security_itm%obj%row
SECHIST	CCM security header history	CRSP_CCM_SECHIST_T	array	sechist_itm%obj%arr
SEC_MTHSPT	CCM security monthly split events	CRSP_CCM_SEC_MTHSPT_T	row	sec_mthspt_itm%obj%row
SEC_MSPT_FN	CCM security monthly split event footnotes	CRSP_CCM_SEC_MTH_FN_T	row	sec_mspt_fn_itm%obj%row
SEC_MDIV_FN	CCM security monthly dividend event footnotes	CRSP_CCM_SEC_MTH_FN_T	row	sec_mdiv_fn_itm%obj%row
SEC_SPIND	CCM security S&P information events	CRSP_CCM_SEC_SPIND_T	row	sec_spind_itm%obj%row
IDXCST_HIS	CCM security historical index constituents	CRSP_CCM_IDXCST_HIS_T	array	idxcst_his_itm%obj%arr
SPIDX_CST	CCM security S&P index constituent events	CRSP_CCM_SPIDX_CST_T	array	spidx_cst_itm%obj%arr
CCM_SEGCUR	CCM opseg currency rate data	CRSP_CST_SEGCUR_T	array	ccm_segcur_itm%obj%arr
CCM_SEGSRC	CCM opseg source data	CRSP_CST_SEGSRC_T	array	ccm_segsrc_itm%obj%arr

CCM_SEGPROD	CCM opseg product data	CRSP_CST_SEGPROD_T	array	ccm_segprod_itm%obj%arr
CCM_SEGCUST	CCM opseg customer data	CRSP_CST_SEGCUST_T	array	ccm_segcust_itm%obj%arr
CCM_SEGDTL	CCM opseg detail data	CRSP_CST_SEGDTL_T	array	ccm_segdtl_itm%obj%arr
CCM_SEGITM	CCM opseg item data	CRSP_CST_SEGITM_T	array	ccm_seg_itm%obj%arr
CCM_SEGNAICS	CCM opseg NAICS data	CRSP_CST_SEGNAICS_T	array	ccm_segnaics_itm%obj%arr
CCM_SEGGEO	CCM opseg geographic data	CRSP_CST_SEGGEO_T	array	ccm_seggeo_itm%obj%arr

MASTER

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
MASTER				**master_itm%arr%master_val**
BEGQTR	Quarterly date of earliest data (yyyy.q)	int(4)	I6	master_val%begqtr
BEGYR	Annual date of earliest data (yyyymmdd)	int(4)	I4	master_val%begyr
CBEGDT	First date of Compustat data	int(4)	I8	master_val%cbegdt
CCMID	Permanent record identifier for Compustat company or index data, represents GVKEY for company, GVKEYX for index	int(4)	I6	master_val%ccmid
CCMIDTYPE	Type of key for Compustat data. 1 = company data, 2 = index data	int(4)	I2	master_val%ccmidtype
CENDT	Last date of Compustat data	int(4)	I8	master_val%cendt
ENDQTR	Quarterly date of last data (yyyy.q)	int(4)	I6	master_val%endqtr
ENDYR	Annual date of last data (yyyymmdd)	int(4)	I4	master_val%endyr

COMPANY

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
COMPANY				**company_itm%arr%company_val**
ADD1-4	Address lines 1-4	char(68)	A65	company_val%add#
ADDZIP	Postal code	char(24)	A24	company_val%addzip
BUSDESC	Business description	char(2000)	A2000	company_val%busdesc
CIK	CIK number	char(12)	A10	company_val%cik
CITY	City	char(104)	A104	company_val%city

CONM	Company name	char(256)	A255	company_val%conm
CONML	Company legal name	char(104)	A100	company_val%conml
COSTAT	Postal code	char(24)	A1	company_val%addzip
COUNTY	County code	char(104)	A100	company_val%county
DLDTE	Research company deletion date	int(4)	I8	company_val%dldte
DLRSN	Research company reason for deletion	char(12)	A8	company_val%dlrsn
EIN	Employer identification number	char(12)	A10	company_val%ein
FAX	Fax number	char(24)	A18	company_val%fax
FIC	ISO Country code of incorporation	char(16)	A3	company_val%fic
FYRC	Fiscal year end (current)	int(4)	I2	company_val%fyrc
GGROUP	GICS groups	char(12)	A4	company_val%ggroup
GIND	GICS industries	char(12)	A6	company_val%gind
GSECTOR	GICS sectors	char(12)	A2	company_val%gsector
GSUBIND	GICS sub-industries	char(12)	A8	company_val%gsubind
IDBFLAG	International/Domestic/Both indicator	char(12)	A1	company_val%idbflag
INCORP	State/Province of incorporation code	char(12)	A8	company_val%incorp
IPODATE	Company initial public offering date	int(4)	I8	company_val%ipodate
LOC	ISOCountry code/ headquarters	char(4)	A3	company_val%loc
NAICS	North American Industry Classification Code	char(8)	A6	company_val%naics
PHONE	Phone number	char(24)	A18	company_val%phone
PRICAN	Primary Issue Tag - Canada	char(12)	A8	company_val%prican
PRIROW	Primary Issue Tag – rest of world	char(12)	A8	company_val%prirow
PRIUSA	Primary Issue Tag - USA	char(12)	A8	company_val%priusa
SIC	SIC code	int(4)	I4	company_val%sic
SPCINDCD	S&P industry sector code - reference	int(4)	I4	company_val%spcindcd
SPCSECCD	S&P economic sector code - reference	int(4)	I4	company_val%spcseccd
STATE	State/Province	char(12)	A8	company_val%state
STKO	Stock ownership code	int(4)	I1	company_val%stko
WEBURL	Website address	char(68)	A60	company_val%weburl

IDX_INDEX

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
IDX_INDEX				**idx_index_itm%arr%idx_index_val**
IDX13KEY	13 character key	char(16)	A13	idx_index_val%idx13key
IDXCSTFLG	Index constituent flag	char(4)	A2	idx_index_val%idxcstflg
IDXSTAT	Index Status	char(2)	A1	Idx_index_val%idxstat
INDEXCAT	Index category code	char(12)	A10	idx_index_val%indexcat
INDEXGEO	Index geographical area	char(12)	A10	idx_index_val%indexgeo
INDEXTYPE	Index type	char(12)	A10	idx_index_val%indextype
INDEXVAL	Index value	char(12)	A10	idx_index_val%indexval
SPII	S&P industry index code	int(4)	I4	idx_index_val%spii
SPMI	S&P major index code	int(4)	I4	idx_index_val%spmi
TICI	Issue trading ticker	char(12)	A8	idx_index_val%tici
XCONM	Company Name (Index)	char(256)	A255	idx_index_val%xconm
XINDEXID	Index ID	char(12)	A12	idx_index_val%xindexid
XTIC	Ticker/trading symbol (index)	char(10)	A10	idx_index_val%xtic

SPIND

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SPIND				**spind_itm%arr%spind_val**
SPIID	S&P Industry ID	int(4)	I4	spind_val%spiid
SPIMID	S&P Major Index ID	int(4)	I4	spind_val%spimid
SPITIC	S&P Index ticker	char(12)	A12	spind_val%spitic
SPIDESC	S&P Index industry description /reference	char(256)	A256	spind_val%spidesc

COMPHIST

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
COMPHIST				**comphist_itm%arr%comphist_arr(i)**
HCHGENDDT	Comphist description last effective date	int(4)	I8	comphist_arr(i)%hchgenddt
HDLDTE	Historical research company – deletion date	int(4)	I8	comphist_arr(i)%hdldte
HFYRC	Historical fiscal year end month / current	int(4)	I10	comphist_arr(i)%hfyrc
HIPODATE	Historical company official public offering date	int(4)	I10	comphist_arr(i)%hipodate
HSIC	Historical SIC Code	int(4)	I10	comphist_arr(i)%hsic
HSPCINDCD	Historical S&P Industry code	int(4)	I10	comphist_arr(i)%hspcindcd
HSPCSECCD	Historical S&P Economic sector code	int(4)	I10	comphist_arr(i)%hspcseccd
HSTKO	Historical stock ownership code	int(4)	I10	comphist_arr(i)%hstko
HADD1…4	Historical address lines 1-4	char(68)	A68	comphist_arr(i)%haddl#
HADDZIP	Historical postal code	char(68)	A24	comphist_arr(i)%haddzip
HBUSDESC	Historical business description	char(2000)	A2000	comphist_arr(i)%hbusdesc
HCIK	Historical CIK number	char(12)	A12	comphist_arr(i)%hcik
HCITY	Historical city	char(104)	A104	comphist_arr(i)%hcity
HCONM	Historical company name	char(256)	A256	comphist_arr(i)%hconm
HCONML	Historical legal company name	char(104)	A104	comphist_arr(i)%hconml
HCOSTAT	Historical active/inactive status marker	char(4)	A4	comphist_arr(i)%hcostat
HCOUNTY	Historical county code	char(104)	A1044	comphist_arr(i)%hcounty
HDLRSN	Historical research company reason for deletion	char(12)	A12	comphist_arr(i)%hdlrsn
HEIN	Historical employer identification number	char(12)	A12	comphist_arr(i)%hein
HFAX	Historical fax number	char(16)	A16	comphist_arr(i)%hfax
HFIC	Historical ISO country code / incorporation	char(16)	A16	comphist_arr(i)%hfic
HGGROUP	Historical GICS group	char(12)	A12	comphist_arr(i)%hggroup
HGIND	Historical GICS industries	char(12)	A12	comphist_arr(i)%hgind
HGSECTOR	Historical GICS sector	char(12)	A12	comphist_arr(i)%hgsector
HGSUBIND	Historical GICS sub-industries	char(12)	A12	comphist_arr(i)%hgsubind
HIDBFLAG	Historical international, domestic, both indicator	char(12)	A12	comphist_arr(i)%hidbflag

HINCORP	Historical state/province of incorporation code	char(12)	A12	comphist_arr(i)%hincorp
HLOC	Historic ISO country code/ headquarters	char(4)	A4	comphist_arr(i)%hloc
HNAICS	Historical NAICS codes	char(8)	A8	comphist_arr(i)%hnaics
HPHONE	Historical phone number	char(16)	A16	comphist_arr(i)%hphone
HPRICAN	Historical primary issue tag - Cananda	char(12)	A12	comphist_arr(i)%hprican
HPRIROW	Historical primary issue tag – rest of world	char(12)	A12	comphist_arr(i)%hprirow
HPRIUSA	Historical primary issue tag - US	char(12)	A12	comphist_arr(i)%hpriusa
HSTATE	Historical state/province	char(12)	A12	comphist_arr(i)%hstate
HWEBURL	Historical website url	char(68)	A68	comphist_arr(i)%hweburl

CSTHIST

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CSTHIST				**csthist_itm%arr%csthist_arr(i)**
CST_CHGDT	CST History effective date	int(4)	I8	csthist_arr(i)%cst_chgdt
CST_CHGENDDT	CST History last effective date	int(4)	I8	csthist_arr(i)%cst_chgenddt
CST_DNUM	CST History industry code	int(4)	I4	csthist_arr(i)%cst_dnum
CST_FILE	CST History file identification code	int(4)	I4	csthist_arr(i)%cst_file
CST_ZLIST	CST History exchange listing and S&P Index code	int(4)	I4	csthist_arr(i)%cst_zlist
CST_STATE	CST History state identification code	int(4)	I4	csthist_arr(i)%cst_state
CST_COUNTY	CST History county identification code	int(4)	I4	csthist_arr(i)%cst_county
CST_STINC	CST History state incorporation code	int(4)	I4	csthist_arr(i)%cst_stinc
CST_FINC	CST History foreign incorporation code	int(4)	I4	csthist_arr(i)%cst_finc
CST_XREL	CST History industry index relative code	int(4)	I4	csthist_arr(i)%cst_xrel
CST_STK	CST History stock ownership code	int(4)	I4	csthist_arr(i)%cst_stk
CST_DUP	CST History duplicate file code	int(4)	I4	csthist_arr(i)%cst_dup
CST_CCNDX	CST History current Canadian index code	int(4)	I4	csthist_arr(i)%cst_ccndx
CST_GICS	CST History Global Industry Classification Standard Code	int(4)	I4	csthist_arr(i)%cst_gics
CST_IPODT	CST History IPO date	int(4)	I4	csthist_arr(i)%cst_ipodt
CST_FUNDF1	CST History fundamental file identification code 1	int(4)	I4	csthist_arr(i)%cst_fund1

CST_FUNDF2	CST History fundamental file identification code 2	int(4)	I4	csthist_arr(i)%cst_fundf2
CST_FUNDF3	CST History fundamental file identification code 3	int(4)	I4	csthist_arr(i)%cst_fundf3
CST_NAICS	CST History North American Industry Classification	char(8)	A8	csthist_arr(i)%cst_naics
CST_CPSPIN	CST History primary S&P Index marker	char(4)	A4	csthist_arr(i)%cst_cpspin
CST_CSSPIN	CST History subset S&P Index marker	char(4)	A4	csthist_arr(i)%cst_csspin
CST_CSSPII	CST History secondary S&P Index marker	char(4)	A4	csthist_arr(i)%cst_csspii
CST_SUBDBT	CST History current S&P subordinated debt rating	char(8)	A8	csthist_arr(i)%cst_subdbt
CST_CPAPER	CST History current S&P commercial paper rating	char(4)	A4	csthist_arr(i)%cst_cpaper
CST_SDBT	CST History current S&P senior debt rating	char(4)	A4	csthist_arr(i)%cst_sdbt
CST_SDBTIM	CST History current S&P senior debt rating - footnote	char(4)	A4	csthist_arr(i)%cst_sdbtim
CST_CNUM	CST History CUSIP issuer code	char(16)	A16	csthist_arr(i)%cst_cnum
CST_CIC	CST History issuer number	char(4)	A4	csthist_arr(i)%cst_cic
CST_CONAME	CST History company name	char(64)	A64	csthist_arr(i)%cst_coname
CST_INAME	CST History industry name	char(4)	A4	csthist_arr(i)%cst_iname
CST_SMBL	CST History stock ticker symbol	char(16)	A16	csthist_arr(i)%cst_smbl
CST_EIN	CST History employer identification number	char(16)	A16	csthist_arr(i)%cst_ein
CST_INCORP	CST History incorporation ISO country code	char(4)	A4	csthist_arr(i)%cst_incorp

LINK

CCM MNEMONIC	**FIELD NAME**	**INTERNAL STORAGE**	**DISPLAY FORMAT**	**FIELD USAGE**
LINK				**link_itm%arr%link_arr(i)**
LINKDT	Effective date of the link record	int(4)	I8	link_arr(i)%linkdt
LINKENDDT	Last effective date of the link record	int(4)	I8	link_arr(i)%linkenddt
LPERMNO	CRSP PERMNO link during link period	int(4)	I6	link_arr(i)%lpermno
LPERMCO	CRSP PERMCO link during link period	int(4)	I10	link_arr(i)%lpermco
LIID	Security identifier	char(4)	A3	link_arr(i)%liid
LNKTYPE	Link type code	char(4)	A4	link_arr(i)%lnktype
LINKPRIM	Primary security link marker	char(4)	A1	link_arr(i)%linkprim

LINKUSED

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
LINKUSED				**linkused_itm%arr%linkused_arr(i)**
ULINKDT	Effective date of the link	int(4)	I8	linkused_arr(i)%ulinkdt
ULINKENDDT	Last effective date of the link	int(4)	I8	linkused_arr(i)%ulinkenddt
ULINKID	Linkused row identifier	int(4)	I	linkused_arr(i)%ulinkid
UGVKEY	GVKEY used in the link	int(4)	I6	linkused_arr(i)%ugvkey
UPERMNO	CRSP PERMNO link during link period	int(4)	I6	linkused_arr(i)%upermno
UPERMCO	CRSP PERMCO link during link period	int(4)	I6	linkused_arr(i)%upermco
UIID	Used Security ID	char(4)	A3	linkused_arr(i)%uiid
USEDFLAG	Flag marking whether link is used in building composite record	int(4)	I	linkused_arr(i)%usedflag
ULINKPRIM	Used link primary marker	char(4)	A1	linkused_arr(i)%ulinkprim
ULINKTYPE	Used link type	char(4)	A4	linkused_arr(i)%ulinktype

LINKRNG

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
LINKRNG				**linkrng_itm%arr%linkrng_arr(i)**
RKEYSET	Keyset applicable to range	int(4)	I8	linkrng_arr(i)%rkeyset
RCALID	Calendar applicable to range	int(4)	I8	linkrng_arr(i)%rcalid
RBEGIND	Beginning time series range of link	int(4)	I8	linkrng_arr(i)%rbegind
RENDIND	Ending time series range of link	int(4)	I8	linkrng_arr(i)%rendind
RPREVIND	Time series range immediately preceding the link	int(4)	I8	linkrng_arr(i)%rprevind
RBEGDT	Beginning calendar range of link	int(4)	I8	linkrng_arr(i)%rbegdt
RENDDT	Ending calendar range of link	int(4)	I8	linkrng_arr(i)%renddt
RPREVDT	Ending calendar range preceding the link	int(4)	I8	linkrng_arr(i)%rprevdt
RFISCAL_DATA_FLG	Type of time series, C-calendar or F-fiscal.	char(8)	A8	linkrng_arr(i)%rfiscal_data_flg
EFFDATE	Effective date- company cumulative factor	int(4)	I10	linkrng_arr(i)%effdate
THRUDATE	Thu date – company cumulative factor	int(4)	I10	linkrng_arr(i)%thrudate

ADJFACT

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
ADJFACT				**adjfact_itm%arr%adjfact_arr(i)**
ADJEX	Cumulative adjustment factor by Ex-date	double(8)	F18.4	adjfact_arr(i)%adjex
ADJPAY	Cumulative adjustment factor by Pay-date	double(8)	F18.4	adjfact_arr(i)%adjpay

HGIC

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
HGIC				**hgic_itm%arr%hgic_arr(i)**
INDFROM	Effective from (start) date	int(4)	I8	hgic_arr(i)%indfrom
INDTHRU	Effective through (last) date	int(4)	I8	hgic_arr(i)%indthru
GGROUPH	Industry group name	char(12)	A12	hgic_arr(i)%ggrouph
GINDH	Group industry	char(12)	A12	hgic_arr(i)%gindh
GSECTORH	Group industry sector	char(12)	A12	hgic_arr(i)%gsectorh
GSUBINDH	Group sub-industries	char(12)	A12	hgic_arr(i)%gsubindh

OFFTITL

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
OFFTITL				**offtitl_itm%arr%offtitl_arr(i)**
OFID	Officer ID	int(4)	I9	offtitl_arr(i)%ofid
OFCD	Officer title	char(16)	A8	offtitl_arr(i)%ofcd
OFNM	Officer Name(s)	char(40)	A39	offtitl_arr(i)%ofnm

CCM_FILEDATE

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CCM_FILEDATE				**ccm_filedate_itm%arr%ccm_filedate_arr(i)**
FDATADATE	Company filing data date	int(4)	I8	ccm_filedate_arr(i)%fdatadate
FCONSOL	Company consolidation level filedate	char(2)	A2	ccm_filedate_arr(i)%fconsol
FPOPSRC	Population source filedate	char(2)	A2	ccm_filedate_arr(i)%fpopsrc
SRCTYPE	Document source type filedate	char(12)	A12	ccm_filedate_arr(i)%srctype
FILEDATE	Company filing date	int(4)	I8	ccm_filedate_arr(i)%filedate

CCM_IPCD

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CCM_IPCD				**ccm_ipcd_itm%arr%ccm_ipcd_arr(i)**
IPDATADATE	Industry presentation code data date	int(4)	I8	ccm_ipcd_arr(i)%ipdatadate
IPCONSOL	Level of consolidation (Industry presentation code)	char(2)	A1	ccm_ipcd_arr(i)%ipconsol
IPPOPSRC	Population source (Industry presentation code)	char(2)	A1	ccm_ipcd_arr(i)%ippopsrc
IPCD	Industry presentation code	char(2)	A1	ccm_ipcd_arr(i)%ipcd

SECURITY

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SECURITY				**security_itm%arr%security_val**
EXCHG	Stock exchange	int(4)	I4	security_val%exchg
DLDTEI	Security inactivation date	int(4)	I8	security_val%dltei
IID_SEQ_ NUM	IID sequence number	int(4)	I8	security_val%iid_seq_num
SBEGDT	First date of Compustat data for issue	int(4)	I8	security_val%sbegdt
SENDDT	Last date of Compustat data for issue	int(4)	I8	security_val%senddt
IID	Issue ID	char(4)	A3	security_val%iid
SCUSIP	CUSIP	char(12)	A12	security_val%cusip
DLRSNI	Security inactivation code	char(12)	A8	security_val%dlrsni

DSCI	Security description	char(32)	A28	security_val%dsci
EPF	Earnings participation flag	char(4)	A1	security_val%epf
EXCNTRY	Stock exchange country code	char(4)	A3	security_val%excntry
ISIN	International security identification number	char(16)	A12	security_val%isin
SSECSTAT	Security status marker	char(4)	A4	security_val%ssecstat
SEDOL	SEDOL	char(8)	A7	security_val%sedol
TIC	Ticker/trading symbol	char(12)	A8	security_val%tic
TPCI	Issue type	char(12)	A8	security_val%tpci

SECHIST

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SECHIST				**sechist_itm%arr%sechist_arr(i)**
HSCHGDT	Historical security change date	int(4)	I8	sechist_arr(i)%hschgdt
HSCHGENDDT	Historical security change end date	int(4)	I8	sechist_arr(i)%hschgenddt
HEXCHG	Historical stock exchange	int(4)	I10	sechist_arr(i)%hexchg
HDLDTEI	Historical security inactivation date	int(4)	I8	sechist_arr(i)%hdldtei
HIID_SEQ_NUM	Historical issue ID sequence number	int(4)	I10	sechist_arr(i)%hiid_seq_num
HIID	Historical issue ID	char(4)	A3	sechist_arr(i)%hiid
HSCUSIP	Historical CUSIP	char(12)	A12	sechist_arr(i)%hscusip
HDLRSNI	Historical security inactivation code	char(12)	A12	sechist_arr(i)%hdhdrsni
HDSCI	Historical security description	char(32)	A32	sechist_arr(i)%hdsci
HEPF	Historical earnings participation flag	char(4)	A4	sechist_arr(i)%hepf
HEXCNTRY	Historical stock exchange country code	char(4)	A4	sechist_arr(i)%hexcntry
HISIN	Historical international security identification number	char(16)	A16	sechist_arr(i)%hisin
HSSECSTAT	Historical security status marker	char(4)	A4	sechist_arr(i)%hssecstat
HSEDOL	Historical SEDOL	char(8)	A8	sechist(i)%hsedol
HTIC	Historical ticker/trading symbol	char(12)	A12	sechist_arr(i)%htic
HTPCI	Historical issue type	char(12)	A12	sechist_arr(i)%htpci

SEC_MTHSPT

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SEC_MTHSPT				**sec_mthspt_itm%arr%sec_mthspt_arr(i)**
DATADATEM	Monthly adjustment factor data date	int(4)	I10	sec_mthspt_arr(i)%datadatem
RAWPM	Raw adjustment factor – pay date - monthly	double(8)	F18.4	sec_mthspt_arr(i)%rawpm
RAWXM	Raw adjustment factor – ex date - monthly	double(8)	F18.4	sec_mthspt_arr(i)%rawxm

SEC_MSPT_FN

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SEC_MSPT_FN				**sec_mspt_fn_itm%arr%sec_mspt_fn_arr(i)**
DATAITEMMF	Monthly split footnote dataitem	char(8)	A8	sec_mspt_fn_arr(i)%dataitemmf
RAWPM_FN1..FN5	Raw adjustment factor – pay date – monthly – footnotes 1-5	char(4)	A4	sec_mspt_fn_arr(i)%rawpm_fn1..fn5
RAWXM_FN1..FN5	Raw adjustment factor – ex date – monthly – footnotes 1-5	char(4)	A4	sec_mspt_fn_arr(i)%rawxm_fn1..fn5

SEC_MDIV_FN

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SEC_MDIV_FN				**sec_mdiv_fn_itm%arr%sec_mdiv_fn_arr(i)**
DIVDATADATEMF	Monthly dividend footnote data date	int(4)	I10	sec_mdiv_fn_arr(i)%divdatadatemf
DIVDATAITEMMF	Monthly dividend footnote data item	char(8)	A8	sec_mdiv_fn_arr(i)%divdataitemmf
DVPSPM_FN1..FN5	Dividend per share – pay date – monthly – footnotes 1-5	char(4)	A4	sec_mdiv_fn_arr(i)%dvpspm_fn1..fn5
DVPSXM_FN1..FN5	Dividend per share – ex date – monthly – footnotes 1-5	char(4)	A4	sec_mdiv_fn_arr(i)%dvpsxm_fn1..fn5

SEC_SPIND

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SEC_SPIND				**sec_spind_itm%arr%sec_spind_arr(i)**
SPBEGDATE	S&P Index event beginning date	int(4)	I10	sec_spind_arr(i)%spbegdate
SPENDDATE	S&P Index event ending date	int(4)	I10	sec_spind_arr(i)%spenddate

SPHIID	S&P holdings industry index ID	int(4)	I4	sec_spind_arr(i)%sphiid
SPHMID	S&P holdings major index ID	int(4)	I4	sec_spind_arr(i)%sphmid
SPHSEC	S&P holdings sector code	int(4)	I4	sec_spind_arr(i)%sphsec
SPH100	S&P holdings S&P 100 marker	int(4)	I4	sec_spind_arr(i)%sph100
SPHCUSIP	S&P holdings CUSIP	char(12)	A9	sec_spind_arr(i)%sphcusip
SPHNAME	S&P holdings name	char(36)	A31	sec_spind_arr(i)%sphname
SPHTIC	S&P holdings ticker	char(12)	A8	sec_spind_arr(i)%sphtic
SPHVG	S&P holdings value/growth indicator	char(4)	A1	sec_spind_arr(i)%sphvg

IDXCST_HIS

CCM MNEMONIC	**FIELD NAME**	**INTERNAL STORAGE**	**DISPLAY FORMAT**	**FIELD USAGE**
IDXCST_HIS				**idxcst_his_itm%arr%idxcst_his_arr(i)**
XFROM	S&P constituent from event date	int(4)	I10	idxcst_his_arr(i)%xfrom
IDX13KEY	S&P 13 character key	char(16)	A13	idxcst_his_arr(i)%idx13key
XGVKETX	S&P constituent event index GVKEY	int(4)	I10	idxcst_his_arr(i)%xgvkeyx

SPIDX_CST

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
SPIDX_CST				**spidx_cst_itm%arr%spidx_cst_arr(i)**
SXBEGDATE	S&P constituent event beginning date	int(4)	I10	spidx_cst_arr(i)%sxbegdate
SXENDDATE	S&P constituent event ending date	int(4)	I10	spidx_cst_arr(i)%sxenddate
SPFLOAT	S&P float shares	int(4)	I4	spidx_cst_arr(i)%spfloat
INDEXID	S&P major index ID	char(12)	A10	spidx_cst_arr(i)%indexid
EXCHGX	S&P constituent exchange	char(8)	A4	spidx_cst_arr(i)%exchgx
TICX	S&P holdings ticker	char(12)	A10	spidx_cst_arr(i)%ticx
CUSIPX	S&P constituent CUSIP	char(12)	A9	spidx_cst_arr(i)%cusipx
CONMX	S&P constituent name	char(44)	A40	spidx_cst_arr(i)%conmx
CONTYPE	S&P constituent type	char(12)	A10	spidx_cst_arr(i)%contype
CONVAL	S&P constituent value	char(12)	A10	spidx_cst_arr(i)%conval

CCM_SEGCUR

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CCM_SEGCUR				**ccm_segcur_itm%arr%segcur_arr(i)**
SC_DATAYR	Data year	int(4)	I4	segcur_arr(i)%sc_datayr
SC_DATAFYR	Data fiscal year end month	int(4)	I2	segcur_arr(i)%sc_datafyr
SC_CALYR	Data calendar year	int(4)	I4	segcur_arr(i)%sc_calyr
SC_SRCYFYR	Source fiscal year end month	int(4)	I2	segcur_arr(i)%sc_srcfyr
SC_XRATE	Period end exchange rate	double(8)	F16.8	segcur_arr(i)%sc_xrate
SC_XRATE12	12-month moving exchange rate	double(8)	F16.8	segcur_arr(i)%sc_xrate12
SC_SRCCUR	Source currency code	char(4)	A3	segcur_arr(i)%sc_srccur
SC_CURCD	ISO currency code (USD)	char(4)	A3	segcur_arr(i)%sc_curcd

CCM_SEGSRC

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CCM_SEGSRC				**ccm_segsrc_itm%arr%segsrc_arr(i)**
SS_SRCYR	Source year	int(4)	I4	segsrc_arr(i)%ss_srcyr
SS_SRCFYR	Source fiscal year end month	int(4)	I2	segsrc_arr(i)%ss_srcfyr
SS_CALYR	Data calendar year	int(4)	I4	segsrc_arr(i)%ss_calyr
SS_RCST1	Reserved 1	int(4)	I4	segsrc_arr(i)%ss_rcst1
SS_SSRCE	Source document code	char(4)	A2	segsrc_arr(i)%ss_ssrce
SS_SUCODE	Source update code	char(4)	A2	segsrc_arr(i)%ss_sucode
SS_CURCD	ISO currency code	char(4)	A3	segsrc_arr(i)%ss_curcd
SS_SRCCUR	Source ISO currency code	char(4)	A3	segsrc_arr(i)%ss_srccur
SS_HNAICS	Segment primary historical NAICS	char(8)	A6	segsrc_arr(i)%ss_hnaics

CCM_SEGPROD

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CCM_SEGPROD				**ccm_segprod_itm%arr%segprod_arr(i)**
SP_SRCYR	Source year	int(4)	I4	segprod_arr(i)%sp_srcyr
SP_SRCFYR	Source fiscal year end month	int(4)	I2	segprod_arr(i)%sp_srcfyr
SP_PDID	Product identifier	int(4)	I4	segprod_arr(i)%sp_pdid
SP_PSID	Segment link – segment identifier	int(4)	I4	segprod_arr(i)%sp_psid
SP_PSALE	External revenues	float(4)	F10.4	segprod_arr(i)%sp_psale
SP_RCST1	Reserved 1	float(4)	F10.4	segprod_arr(i)%sp_rcst1
SP_PNAICS	Product NAICS code	char(8)	A6	segprod_arr(i)%sp_pnaics
SP_PSTYPE	Segment link- segment type	char(16)	A83	segprod_arr(i)%sp_pstype
SP_PNAME	Product name	char(64)	A64	segprod_arr(i)%sp_pname

CCM_SEGCUST

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CCM_SEGCUST				**ccm_segcust_itm%arr%segcust_arr(i)**
SC_SRCYR	Source year	int(4)	I4	segcust_arr(i)%sc_srcyr
SC_SRCFYR	Source fiscal year end month	int(4)	I2	segcust_arr(i)%sc_srcfyr
SC_CDID	customer identifier	int(4)	I4	segcust_arr(i)%sc_cdid
SC_CSID	Segment link – segment identifier	int(4)	I4	segcust_arr(i)%sc_csid
SC_CSALE	customer revenues	float(4)	F10.4	segcust_arr(i)%sc_csale
SC_RCST1	Reserved 1	int(4)	I4	segcust_arr(i)%sc_rcst1
SC_CTYPE	Customer type	char(16)	A8	segcust_arr(i)%sc_ctype
SC_CGEOCD	Geographic area code	char(16)	A8	segcust_arr(i)%sc_cgeocd
SC_CGEOAR	Geographic area type	char(16)	A8	segcust_arr(i)%sc_cgeoar
SC_CSTYPE	Segment link – segment type	char(16)	A8	segcust_arr(i)%sc_cstype
SC_CNAME	Customer name data	char(64)	A64	segcust_arr(i)%sc_cname

CCM_SEGDTL

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CCM_SEGDTL				**ccm_segdtl_itm%arr%segdtl_arr(i)**
SD_SRCYR	Source year	int(4)	I4	segdtl_arr(i)%sd_srcyr
SD_SRCFYR	Source fiscal year end month	int(4)	I2	segdtl_arr(i)%sd_srcfyr
SD_SID	Segment identifier	int(4)	I4	segdtl_arr(i)%sd_sid
SD_RCST1	Reserved 1	int(4)	I4	segdtl_arr(i)%sd_rcst1
SD_STYPE	Segment type	char(16)	A8	segdtl_arr(i)%sd_stype
SD_SOPTP1	Operating segment type 1	char(16)	A8	segdtl_arr(i)%sd_ctype
SD_SOPTP2	Operating segment type	char(16)	A8	segdtl_arr(i)%sd_cgeocd
SD_SGEOTP	Geographic segment type	char(16)	A8	segdtl_arr(i)%sd_cgeoar
SD_SNAME	Segment name	char(256)	A64	segdtl_arr(i)%sd_cname

CCM_SEGITM

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CCM_SEGITM				**ccm_segitm_itm%arr%segitm_arr(i)**
SI_DATYR	Data year	int(4)	I4	segitm_arr(i)%si_datyr
SI_FISCYR	Data fiscal year end month	int(4)	I4	segitm_arr(i)%si_fiscyr
SI_SRCYR	Source year	int(4)	I4	segitm_arr(i)%si_srcyr
SI_SRCFYR	Source fiscal year end month	int(4)	I2	segitm_arr(i)%si_crsfyr
SI_CALYR	Data calendar year	int(4)	I4	segitm_arr(i)%si_calyr
SI_SID	Segment identifier	int(4)	I4	segitm_arr(i)%si_sid
SI_EMP	Employees	int(4)	I9	segitm_arr(i)%si_emp
SI_SALE	Net sales	float(4)	F10.4	segitm_arr(i)%si_sale
SI_OIBD	Operating income before depreciations	float(4)	F10.4	segitm_arr(i)%si_oibd
SI_DP	Depreciation & amortization	float(4)	F10.4	segitm_arr(i)%si_dp
SI_OIAD	Operating income after depreciation	float(4)	F10.4	segitm_arr(i)%si_oiad
SI_CAPX	Capital expenditures	float(4)	F10.4	segitm_arr(i)%si_capx
SI_IAT	Identifiable assets	float(4)	F10.4	segitm_arr(i)%si_iat
SI_EQEARN	Equity in earnings	float(4)	F10.4	segitm_arr(i)%si_eqearn
SI_INVEQ	Investments at equity	float(4)	F10.4	segitm_arr(i)%si_inveq
SI_RD	Research & development	float(4)	F10.4	segitm_arr(i)%si_rd
SI_OBKLG	Order backlog	float(4)	F10.4	segitm_arr(i)%si_obklg
SI_EXPORTS	Export sales	float(4)	F10.4	segitm_arr(i)%si_exports
SI_INTSEG	Inter-segment eliminations	int(4)	I4	segitm_arr(i)%si_intseg
SI_OPINC	Operating profit	float(4)	F10.4	segitm_arr(i)%si_opinc
SI_PI	Pretax income	float(4)	F10.4	segitm_arr(i)%si_pi
SI_IB	Income before extraordinary earnings	float(4)	F10.4	segitm_arr(i)%si_ib
SI_NI	Net income (loss)	float(4)	F10.4	segitm_arr(i)%si_ni
SI_RCST1	Reserved 1	float(4)	F10.4	segitm_arr(i)%si_rcst1
SI_RCST2	Reserved 2	float(4)	F10.4	segitm_arr(i)%si_rcst2
SI_RCST3	Reserved 3	float(4)	F10.4	segitm_arr(i)%si_rcst3

SI_SALEF	Footnote 1 - sales	char(16)	A8	segitm_arr(i)%si_salef
SI_OPINCF	Footnote 2 – operating profit	char(16)	A8	segitm_arr(i)%si_opincf
SI_CAPXF	Footnote 3 – capital expenditures	char(16)	A8	segitm_arr(i)%si_capxf
SI_EQEARNF	Footnote 4 – equity in earnings	char(16)	A8	segitm_arr(i)%si_eqearnf
SI_EMPF	Footnote 5 - employees	char(16)	A8	segitm_arr(i)%si_empf
SI_RDF	Footnote 6 – research & development	char(16)	A8	segitm_arr(i)%si_rdf
SI_STYPE	Segment type	char(16)	A8	segitm_arr(i)%si_stype

CCM_SEGNAICS

CCM MNEMONIC	**FIELD NAME**	**INTERNAL STORAGE**	**DISPLAY FORMAT**	**FIELD USAGE**
CCM_SEGNAICS				**ccm_segnaics_itm%arr%segnaics_arr(i)**
SN_SRCYR	Source year	int(4)	I4	segnaics_arr(i)%sn_srcyr
SN_SRCFYR	Source fiscal year end month	int(4)	I2	segnaics_arr(i)%sn_srcfyr
SN_SID	Segment identifier	int(4)	I4	segnaics_arr(i)%sn_sid
SN_RCST1	Reserved 1	int(4)	I4	segnaics_arr(i)%sn_rcst1
SN_STYPE	Segment type	char(16)	A8	segnaics_arr(i)%sn_stype
SN_SNAICS	NAICS code	char(8)	A6	segnaics_arr(i)%sn_snaics
SN_RANK	Ranking	int(4)	I4	segnaics_arr(i)%sn_rank
SN_SIC	Segment SIC code	int(4)	I4	segnaics_arr(i)%sn_sic

CCM_SEGGEO

CCM MNEMONIC	FIELD NAME	INTERNAL STORAGE	DISPLAY FORMAT	FIELD USAGE
CCM_SEGGEO				**ccm_seggeo_itm%arr%seggeo_arr(i)**
SG_SRCYR	Source year	int(4)	I4	seggeo_arr(i)%sg_srcyr
SG_SRCFYR	Source fiscal year end month	int(4)	I2	seggeo_arr(i)%sg_srcfyr
SG_SID	Segment identifier	int(4)	I4	seggeo_arr(i)%sg_sid
SG_RCST1	Reserved 1	int(4)	I4	seggeo_arr(i)%sg_rcst1
SG_STYPE	Segment type	char(16)	A8	seggeo_arr(i)%sg_stype
SG_SGEOCD	Geographic area code	char(16)	A8	seggeo_arr(i)%sg_sgeocd
SG_SGEOTP	Geographic area type	char(16)	A8	seggeo_arr(i)%sg_sgeotp

CRSP Fortran 95 API Functions

This section contains an alphabetical list of the functions defined in the CRSP Fortran 95 API. Each definition presents the following information about a function:

- Its prototype
- A list of arguments
- A list of return values
- Side effects
- Preconditions

crsp_f_itm_close

`crsp_f_itm_close` frees all item lists and item indexes, clears all calendar and key lists, closes the database, frees the handle set, and re-initializes the item access handle itself.

PROTOTYPE:	**integer function crsp_f_itm_close (hndl)**
ARGUMENTS:	type(CRSP_ITM_HNDL_T) :: hndl – Access handle to close.
RETURN VALUES:	CRSP_SUCCESS: If the database is successfully closed and all handle data are free CRSP_FAIL: If there is an error in the parameters, inconsistent handle, error closing databases.
SIDE EFFECTS:	If successful, the handle data are emptied: • The database will be closed and the structure cleared. • All internal storage allocated for this instance will be freed
PRECONDITIONS:	The item handle must be previously opened with function crsp_f_itm_init.

Example:

```
if (crsp_f_itm_close(hndl) == CRSP_FAIL) then
       !!--error
       print *, 'Error-- failed to close db:',TRIM(dbpath)
       stop
endif
```

crsp_f_itm_find

`crsp_f_itm_find` attaches a pointer to a `CRSP_ITM_T` item that was previously loaded. The `CRSP_ITM_T` structure describes the data item and contains the underlying time series, array, or row data.

PROTOTYPE:	**integer function crsp_f_itm_find (hndl, itm_name, keyset,itm_foundptr)**
ARGUMENTS:	type(CRSP_ITM_HNDL_T) :: hndl – Access handle containing the needed set structure information and the current item list. character(*) :: itm_name – String containing the itm_name to find. integer :: keyset – Keyset to find type(CRSP_ITM_T), pointer :: itm_foundptr – User-declared pointer that will point to the data item found.
RETURN VALUES:	CRSP_SUCCESS: If successfully found the requested item in the given keyset. CRSP NOT FOUND: If the itm_name and keyset combination are not available CRSP_FAIL: If error in parameters, handle not initialized, or error searching for the item.
SIDE EFFECTS:	If successful, the itm_foundptr will point to a CRSP_ITM_T item with data and information for the desired item and keyset. Otherwise the null() will be assigned to itm_foundptr.
PRECONDITIONS:	The item handle set must be initialized, loaded with a list of items, and opened.

Example:

```
if (crsp_f_itm_find(hndl,'HEADER',0,stkhdr_itm) == CRSP_FAIL          &
       .or. crsp_f_itm_find(hndl,'PRC',0,prc_itm) == CRSP_FAIL        &
       .or. crsp_f_itm_find(hndl,'ADJPRC',0,adjprc_itm) == CRSP_FAIL  &
       .or. crsp_f_itm_find(hndl,'ADJSHR',0,adjshr_itm) == CRSP_FAIL  &
       .or. crsp_f_itm_find(hndl,'ADJVOL',0,adjvol_itm) == CRSP_FAIL  &
       .or.  .not. associated(stkhdr_itm)  &
       .or.  .not. associated(prc_itm)     &
       .or.  .not. associated(adjprc_itm)  &
       .or.  .not. associated(adjshr_itm)  &
       .or.  .not. associated(adjvol_itm)) then
       print *,'Error - invalid item/keyset specified'
       stop
endif
```

crsp_f_itm_find_itmcal

`crsp_f_itm_find_itmcal` attaches a pointer to a `CRSP_ITM_CAL_T` item calendar that was previously loaded. The `CRSP_ITM_CAL_T` structure describes a global calendar or a calendar associated with an item and contains the underlying `CRSP_CAL_T` data.

PROTOTYPE:	**integer function crsp_f_itm_find_itmcal (hndl, calid, keyset,itmcal_foundptr)**
ARGUMENTS:	type(CRSP_ITM_HNDL_T) :: hndl – Access handle containing the needed set structure information and the current item list. Integer :: calid – Calendar id of the calendar to find: • CRSP_CALID_DAILY – daily calendar • CRSP_CALID_MONTHLY – monthly calendar • CRSP_CALID_ANNUAL – annual calendar • CRSP_CALID_QUARTERLY – quarterly calendar • CRSP_CALID_SEMIANNUAL – semi-annual calendar • CRSP_CALID_WEEKLY – weekly calendar integer :: keyset – Keyset of the calendar to find. • keyset >= 0 – when item-access configured with fiscal_disp_cd='C', the calendars associated with fiscal items will be "shifted", based on the loaded company's FYE. • keyset=-1 – will request the global "non-shifted" calendar with the specified calid. type(CRSP_ITM_CAL_T), pointer :: itmcal_foundptr – User-declared pointer that will point to the calendar found.
RETURN VALUES:	CRSP_SUCCESS: If successfully found the requested item calendar in the given keyset. CRSP NOT FOUND: If the calid and keyset combination are not available CRSP_FAIL: If error in parameters, handle not initialized, or error searching for the item calendar.
SIDE EFFECTS:	If successful, the itmcal_foundptr will point to a CRSP_ITM_CAL_T item calendar with data and information for the desired calendar and keyset. Otherwise the null() will be assigned to itmcal_foundptr.
PRECONDITIONS:	The item handle set must be initialized, loaded with a list of items, and opened.

crsp_f_itm_find_itmcal_dt

`crsp_f_itm_find_itmcal_dt` finds the array index of a date entry in `CRSP_ITM_CAL_T` item calendar.

PROTOTYPE:	**integer function crsp_f_itm_find_itmcal_dt (itmcal,dt,match_mode,ifound)**
ARGUMENTS:	type(CRSP_ITM_CAL_T), pointer :: itmcal – Pointer to the item calendar. integer :: dt – Requested date to be found. Format: yyymmdd. integer :: match_mode – Matching mode: • CRSP_EXACT – exact match requested • CRSP_NEXT – if exact is not found – return next valid • CRSP_PREV – if exact is not found – return previous valid integer ::ifound – Array index of the date found. When date is not found, ifound = -1
RETURN VALUES:	CRSP_SUCCESS: If successfully found the requested date. CRSP NOT FOUND: If the date is not found with the given matching mode. CRSP_FAIL: If error in parameters, item calendar is not set or loaded.
SIDE EFFECTS:	If successful, the ifound will be set to a valid array index for the caldt array attached to calendar object. The index can be used to directly access the corresponding time-series item data values associated with this calendar.
PRECONDITIONS:	The item calendar set must be initialized and loaded.

crsp_f_itm_get_key

`crsp_f_itm_get_key` retrieves key information for data loaded by a function `crsp_f_itm_read` call. An output key item list is prepared when the key is initialized, and loaded by function `crsp_f_itm_read`. This function finds the `key_itm_name` in the list and copies ithe value into the user-specified location.

PROTOTYPE:	**integer function crsp_f_itm_get_key (hndl, key_itm_name,keyval)**
ARGUMENTS:	type(CRSP_ITM_HNDL_T) :: hndl – Access handle containing the needed set structure information and the current item list. character(*) :: key_itm_name – String containing an itm_name of a loaded key to be retrieved. {integer OR type(CRSP_VARSTRING_T)} :: keyval – Variable to accept the value of the key item. Data type must agree with the item's type and size.
RETURN VALUES:	CRSP_SUCCESS: If data loaded successfully CRSP_FAIL: If error in parameters, handle not open, key item.
SIDE EFFECTS:	If successful, the keyval is loaded based on the item and key value type.
PRECONDITIONS:	The item handle must be initialized and opened. The item key array must be initialized based on a keytype with the function crsp_f_itm_open or function crsp_f_itm_init_key functions. The key_itm_name must be a valid item for that keytype, and the keyval data must agree with the type of that item.

crsp_f_itm_init

`crsp_f_itm_init` prepares a handle for item access operation for one database and one application id. The handle will be initialized and the database set type and set id identified, allowing loading of reference data and allocation of a set structure.

PROTOTYPE:	**integer function crsp_f_itm_init(hndl, dbpath,app_id, hndl_name)**
ARGUMENTS:	type(CRSP_ITM_HNDL_T) :: hndl – Access handle that will be used to manage the database information and item lists. character(*) :: dbpath – Path to database containing the data to load and the applicable reference data. integer :: app_id – Identifier of a defined application organizing data items into groups for access. Available app_ids can be found in the reference array, function crsp_f_itm_ app. Common app_ids have defined constants: • CRSP_CCMITEMS_ID – generic CCM usage application • CRSP_DSTKITM_ID – generic Daily Stock usage application • CRSP_MSTKITM_ID – generic Monthly Stock usage application • CRSP_DINDITEMS_ID – generic Daily Ind Stock usage application • CRSP_MINDITEMS_ID – generic Monthly Ind Stock usage application character(*) :: hndl name – Name to assign to the handle.
RETURN VALUES:	CRSP_SUCCESS: If initialized successfully CRSP_FAIL: If there is an error in the parameter, database cannot be opened, reference data unavailable, incompatibility between database and app_id.
SIDE EFFECTS:	If successful, the handle data are loaded: • The handle fields are initialized, including all lists and arrays. • The ca_ref structure is loaded with the reference data in the database. If an old database with no reference data, it will use a global reference file with a standard name based on the app_id in the CRSP_LIB directory. • Itm_grp and itm_avail arrays in the handle are loaded with available tables and items • Set_list element is allocated using the database path and setid. The database is opened with a 0 wanted, which loads reference data but allocates no module space. The root information is loaded to the set's CRSP_ROOT_INFO_T structure.
PRECONDITIONS:	The item handle object must be declared and not attached to another access instance. The app_id must exist in the reference data of the database opened.

Example:

```
if (crsp_f_itm_init(hndl,dbpath,stkappid,'stk1') == CRSP_FAIL) then
      !!--error
      print *,'Error - failed to connect to db:',TRIM(dbpath)
      stop
endif
```

crsp_f_itm_is_miss_arrval

`crsp_f_itm_is_miss_arrval` checks if the requested element in a data object attached to the item contains a missing value. `is_miss` is set to `.TRUE.` when a missing value is detected. Only items of simple (non-structured) types are accepted, while the item's underlying data-object can be of structured data-type, in which case the structure offset is used to extract the item value.

PROTOTYPE:	**integer function crsp_f_itm_is_miss_arrval (itm, ind, is_miss)**
ARGUMENTS:	type(CRSP_ITM_T),pointer :: itm – Pointer to the data item integer :: ind – Index of the data array element to check logical :: is_miss – Pointer to the resulting flag value
RETURN VALUES:	CRSP_SUCCESS: If successful, the returned value is initialized and set. CRSP_FAIL: If error in parameters, bad item or element index is out-of-range (ignored in case of CRSP_ROW_T object)
SIDE EFFECTS:	If the requested value contains a missing value, the is_miss is set to .TRUE. Otherwise .FALSE. is assigned.
PRECONDITIONS:	The item has to have a valid bound data-object. Structured items are not allowed. Field items of structures are allowed.

crsp_f_itm_load

`crsp_f_itm_load` prepares items described by a full list and loads them to an item table structure in an item handle. It splits the full list into the global section and the list section and uses the function `crsp_f_itm_expand_elem` on each list element in the list section. This will recursively expand the list elements to fill the structure and apply global qualifiers during the process.

PROTOTYPE:	**integer function crsp_f_itm_load(hndl, full_list,match_flag)**
ARGUMENTS:	type(CRSP_ITM_HNDL_T) :: hndl – Access handle containing the needed set structure information and the current item list. character(*) :: full_list – String describing all items to add, used on standard item notation. integer :: match_flag – Flag setting the behavior when an item if found but not the keyset. Values are: • CRSP_MATCH_REQUIRED (=0) – if any indicated item and keyset is not found no further items will be added, and CRSP_ NOT_FOUND returned. • CRSP_MATCH_FILL (=1)– a dummy item will be created for any item if the item exists but the keyset does not exist for that item in the current database. • CRSP_MATCH_IGNORE (=2) – items will not be added if the keyset is not found, but the return remains CRSP_SUCCESS.
RETURN VALUES:	CRSP_SUCCESS: If successful, and all indicated items loaded according to match_flag CRSP_FAIL: Error in parameters, bad list, handle not initialized, or reference data not available.
SIDE EFFECTS:	If successful, the CRSP_ITM_GRPis loaded with all indicated items. A CRSP_ITM is allocated for each item/keyset pair not already loaded. Object pointers are not set by this function.
PRECONDITIONS:	The item handle set must be loaded. The item table must be initialized with an available app_id. The first set in the set list must agree with the app_id.

crsp_f_itm_load_key

`crsp_f_itm_load_key` defines the keytype that will be used for subsequent reads.

PROTOTYPE:	**integer function crsp_f_itm_load_key(hndl, keytype)**
ARGUMENTS:	type(CRSP_ITM_HNDL_T) :: hndl – Access handle containing the needed set structure information and the current item list. character(*) :: keytype – Name of the key to initialize. Values are: • gvkey – Compustat company key (default) • gvkeyx – Compustat index key • ccmid – gvkey or gvkeyx • permno – CRSP permno found in any links • permco – CRSP permco found in any links • apermno – CRSP-centric composite records by permno • ppermco – CRSP-centric composite records by permno – primary links only • sic – Compustat company SIC code • ticker – Compustat security ticker symbol • cusip – security CUSIP
RETURN VALUES:	CRSP_SUCCESS: If successful CRSP_FAIL: Error in parameters, handle not initialized, or keytype not found.
SIDE EFFECTS:	If successful, the handle is prepared to handle reads.
PRECONDITIONS:	The item handle must be initialized. Keytype must be known for the app_id.

crsp_f_itm_open

`crsp_f_itm_open` registers selected items in a handle by expanding structures and keysets, preparing keys, determining modules needed to access items, opens the needed modules, and binds data in the item lists to the data structure locations. It also builds a master index of all items available in the handle.

PROTOTYPE:	**integer function crsp_f_itm_open (hndl)**
DESCRIPTION:	
ARGUMENTS:	type(CRSP_ITM_HNDL_T) :: hndl – Access handle containing the needed set structure information and the current item list.
RETURN VALUES:	CRSP_SUCCESS: If opens successfully and binds the data CRSP_FAIL: If error in parameters, inconsistent handle, error opening databases or binding items.
SIDE EFFECTS:	If successful, the handle is ready for access: • All items in the loaded list will have object pointer set to the data location in the set data structure. • If the handle grp_fill_cd is 'Y', then the item lists are filled to ensure full tables. Filling creates items to ensure that every itm_name and keyset present in a group each combination is present even if not specified. Filling also arranges the lists so if multiple keysets, each is sorted in the same order as the first keyset seen.
PRECONDITIONS:	The item handle must be previously initialized with function crsp_f_itm_init. It generally follows one or more instances of item load function calls.

Example:

```
if (crsp_f_itm_open(hndl) == CRSP_FAIL) then
       !!--error
       print *,'Error - failed to open db for access'
       stop
endif
```

crsp_f_itm_read

`crsp_f_itm_read` oads data from handle based on item keys specified in prior function `crsp_f_itm_key` calls and the `keyflag` argument. Depending on the level of the entity class, the operation may include reading data from the database into structures and/or specifying data already loaded. This allows a direct or positional read based on `keyflag`.

If the value of the access handle property `fiscal_disp_cd` is "C", any fiscal-based time series are shifted to a calendar basis as part of the read operation.

PROTOTYPE:	**integer function crsp_f_itm_read (hndl, keyflag, key_status)**
ARGUMENTS:	type(CRSP_ITM_HNDL_T) :: hndl – Access handle containing the needed set structure information and the current item list. integer :: keyflag – Code determining how the key is interpreted. • CRSP_EXACT – look for a specific value, • CRSP_BACK or CRSP_FORWARD – direct selection when partial matches are allowed, or a positional qualifier to base selection on the position relative to the last key accessed. • CRSP_NEXT (=-99) – read next key in sequence • CRSP_PREV (=-96) – read previous key in sequence • CRSP_SAME (=-98) – read same key, possibly with different information • CRSP_FIRST (=-95) – read first key in the database • CRSP_LAST (=-97) – read last key in the database integer :: key_status – User provided variable to load with the level of the read. It will be loaded with a 0 if the load results in reading new master data. It will be loaded with a number greater than 0 if the load impacts detail or global data, but no master data are affected.
RETURN VALUES:	CRSP_SUCCESS: If data loaded successfully CRSP_EOF: If positional read reaches the end of the file CRSP_NOT_FOUND: If key not found on exact read. If a detail input key is not provided and no items of that entity class are selected, the return is CRSP_SUCCESS as long as the primary key matches. CRSP_FAIL: If error in parameters, handle not opened, error in read operations.

SIDE EFFECTS:	If successful, the wanted data for the key are loaded into the handle set structure which allows item objects to point to the loaded data. The key found for each level is loaded into the outkey item list. If the handle fiscal_disp_cd is set to calendar-based and items are fiscal-based, shifted calendars are created and time series are converted to calendar basis. The status argument is loaded based on whether the primary key changed. Handle primkey field and readlvl are set. readlvl is set to the rank of the first entity class changed. If the primary key changed, getlvl is set to 0.
PRECONDITIONS:	The item handle must be initialized and opened. The item key must be initialized based on the key type, key element, and the entity class. If not a positional qualifier, the item key inpkey list must be loaded.

Example:

```
sts = crsp_f_itm_read(hndl,CRSP_EXACT, key_sts)
if ( sts == CRSP_FAIL) then
       got_db_error = .true.
       print *,'Error - failed to read db for key:',key
       exit
endif
```

crsp_f_itm_set_key

`crsp_f_itm_set_key` loads key information that will be used to load data in a function `crsp_f_itm_read` call. The key is setup during the function `crsp_f_itm_open` based on the active keytype. The value passed to this function is entered into the handle attached to the input key item.

PROTOTYPE:	**integer function crsp_f_itm_set_key (hndl, key_itm_name, keyval)**
ARGUMENTS:	type(CRSP_ITM_HNDL_T) :: hndl – Access handle containing the needed set structure information and the current item list. character(*) :: key_itm_name – String containing an itm_name of an input key item to be loaded. {integer OR type(CRSP_VARSTRING_T)} :: keyval – Data to be loaded into the key item. Data must agree with the key item's type.
RETURN VALUES:	CRSP_SUCCESS: If data loaded successfully CRSP_FAIL: If error in parameters, handle not open, key item.
SIDE EFFECTS:	If successful, the keyval is copied into the data location for the input key item element in the handle.
PRECONDITIONS:	The item handle must be initialized and opened. The item key array must be initialized based on a keytype with the function crsp_f_itm_open or function crsp_f_itm_init_key functions. The key_itm_name must be a valid item for that keytype, and the keyval data must agree with the type of that item.

Example:

```
if (crsp_f_itm_set_key(hndl,'KYPERMNO',key) == CRSP_FAIL) then
    !!-- error
    print *,'Error - failed to set key:',key
    stop
endif
```

Reference Information

CRSP Fortran 95 API Data Types

All derived types used in the CRSP Fortran 95 API are defined in the module `crsp_f_itm_types.` They are included in user programs automatically through the use of `crsp_f_itm_lib` module.

Note: This document lists only selected properties of the defined types that are relevant in the user-scope of item-based access. The full individual definitions of the specific Fortran 95 derived types can be referenced from the respective include source files. These files are already included in the `crsp_f_itm_lib` module and an explicit include statement is **not** necessary to use the defined types in your programs. The supplied CRSP Fortran 95 API include files are listed in the following table:

PLATFORM/ LOCATION	FILE	DESCRIPTION
Windows 32-bit **Windows 64-bit** %CRSP_INCLUDE% **SunOS sparc** **SunOS i86pc** **Linux 32-bit** **Linux 64-bit** $CRSP_INCLUDE	crsp_itm_ccm_types.inc	CRSP CCM/Compustat specific data types
	crsp_itm_stk_types.inc	CRSP Stock specific data types
	crsp_itm_ind_types.inc	CRSP Index specific data types
	crsp_itm_gen_types.inc	CRSP generic data types used in all supported data sets
	crsp_itm_types.inc	Data types used in context of item-access
	crsp_itm_params.inc	Declarations of constant parameters used.

To use the CRSP Fortran 95 API library in your program simply include a 'use' statement for the top-level module `crsp_f_itm_lib`. All of the required underlying modules will be included automatically. The supplied CRSP Fortran 95 API module files are listed in the following table:

PLATFORM/ LOCATION	FILE	DESCRIPTION
Windows 32-bit **Windows 64-bit** %CRSP_INCLUDE%\mod **SunOS sparc** **SunOS i86pc** $CRSP_INCLUDE/mod **Linux 32-bit** **Linux 64-bit** $CRSP_INCLUDE/mod $CRSP_INCLUDE/mod_g95	crsp_f_itm_lib.mod	CRSP Fortran 95 itm-API user-level module
	crsp_f_varstring.mod	CRSP implementation of varying-length string (CRSP_VARSTRING_T Fortran 95 derived type)
	crsp_f_itm_utils.mod	Implementations of CRSP itm-API interfaces
	crsp_f_itm_types.mod	Fortran 95 derived types used in context of CRSP Fortran 95 itm-API
	crsp_f_itm_xfer.mod	Internal functions and types for CRSP Fortran 95/C exchange layer

Container Objects

CRSP container objects are used to uniformly define the storage for various CRSP data types. Generally, the container's data is stored in the associated Fortran 95 array, except in the case of the CRSP_ROW_T container, where the storage is allocated for an Fortran 95 scalar of the specified data type. The associated storage array is externally allocated with 0-based array bounds.

The CRSP time-series object has an associated calendar of the CRSP_CAL_T object type which is aligned with the time-series data array, attributing the date to the values stored in the time-series array.

CRSP calendar data is stored in the CRSP_CAL_T container object, which defines the loaded calendar and also stores the actual calendar data of the defined type. In the context of the CRSP Fortran 95 API, the calendars associated to the time-series items are of day date-type and are accessed with `caldt` array.

Each container (except CRSP_ROW_T) has a defined availability range, with missing values set beyond the defined range. The missing value is specific to the data type of the stored data and is located at the pre-defined array index position.

Properties of the CRSP container object types are listed in the following tables:

CRSP_TS_T

CRSP time-series container object

NAME	Fortran 95 TYPE	DESCRIPTION
objtype	integer	Object type id (CRSP_TS_OTID)
arrtype	integer	Type id of the data stored in the container
subtype	integer	Subtype id of the data stored in the container
maxarr	integer	Maximum bound for the storage array (index is 0-based)
beg	integer	Lower index of the available stored data
end	integer	Upper index of the available stored data
caltype	integer	Calendar type of the associated calendar object
cal	CRSP_CAL_T	Pointer to associated calendar object
miss_val_at = 0		Array index of the missing value for the stored data type

CRSP_ARRAY_T

CRSP array container object

NAME	Fortran 95 TYPE	DESCRIPTION
objtype	integer	Object type id (CRSP_ARRAY_OTID)
arrtype	integer	Type id of the data stored in the container
subtype	integer	Subtype id of the data stored in the container
maxarr	integer	Maximum bound for the storage array (index is 0-based)
num	integer	Upper index of the available stored data (index is 0-based)
miss_val_at = maxarr - 1		Array index of the missing value for the stored data type

CRSP_ROW_T

CRSP row container object

NAME	Fortran 95 TYPE	DESCRIPTION
objtype	integer	Object type id (CRSP_ROW_OTID)
arrtype	integer	Type id of the data stored in the container
subtype	integer	Subtype id of the data stored in the container

CRSP_CAL_T

CRSP calendar container object

NAME	Fortran 95 TYPE	DESCRIPTION
objtype	integer	Object type id (CRSP_CAL_OTID)
calid	integer	Id code of the loaded calendar: • CRSP_CALID_DAILY • CRSP_CALID_MONTHLY • CRSP_CALID_ANNUAL • CRSP_CALID_QUARTERLY • CRSP_CALID_SEMIANNUAL • CRSP_CALID_WEEKLY
maxarr	integer	Maximum bound of the date storage array
gmtoffset	integer	Minutes offset from GMT
timezone	integer	Code for time zone GMT
relflag	integer	If beg and end absolute or relative
beg	integer	Valid range subset begin
end	integer	Valid range subset end
ndays	integer	Number of periods in calendar
name	character(LEN=CRSP_NAMESIZE)	Calendar name
caldt	integer,dimension(:)	Array of the available day dates (yyyymmdd) in the calendar

Supporting Types

The CRSP Fortran 95 itm-API supporting types provide additional information about data items and other associated objects in the context of item-based access.

An item object is usually associated to a keyset and calendar (in case of time-series items). The details of the keyset (when non-zero) and calendar are given in the `CRSP_ITM_KEYSET_T` and `CRSP_ITM_CAL_T` derived types.

Additionally, the details of the current CRSP data set (such as set name, product name, version, etc.) are provided in the `CRSP_ITM_SET_T` and `CRSP_ROOT_INFO_T` derived types.

The relevant fields of the supporting types are listed in the following tables:

CRSP_ITM_INFO_T

Item detail information

NAME	Fortran 95 TYPE	DESCRIPTION
itm_name	character(LEN=CRSP_NAME_LEN)	Item mnemonic name
dflt_keyset	integer	Default keyset
full_name	character(LEN=CRSP_NAMESIZE)	Full non-mnenonic name
itm_type	character(LEN=CRSP_TYPE_LEN)	Type of data item
derv_flg	character(LEN=CRSP_TYPE_LEN)	Item is stored/derived
unit_type	character(LEN=CRSP_CODE_LEN)	Type of units (money, ratio)
unit_mult	double precision	Multiplier to get actual value
cat_type	character(LEN=CRSP_CODE_LEN)	Category (BS, IS, CF, MKT)
src_type	character(LEN=CRSP_CODE_LEN)	Source (filing, market)
freq_type	character(LEN=CRSP_TYPE_LEN)	Reporting frequency type
disp_fmt	character(LEN=CRSP_ITEMNAME_LEN)	Display format specifier
disp_len	integer	Field width for formatted output
disp_precn	integer	Number of decimal places in output
ca_data_size	integer	Internal length
ca_arrtype	integer	Internal structure it belongs
ca_subtype	integer	Internal data sub type
subno_type	integer	Type of variant id
epsflag	integer	Epsilon type/digits for diffs
cepsflag	integer	Epsilon type for character(LEN=diffs)
epsilon	double precision	Absolute epsilon for diffs
desc	character(LEN=CRSP_DESC_LEN)	Default description for field

CRSP_ITM_KEYSET_T

Keyset descriptor

NAME	Fortran 95 TYPE	DESCRIPTION
keyset	integer	Keyset number
is_loaded	logical	True when items where requested with this keyset
keyset_info	CRSP_KEYSET_T	Information about the keyset
items_arr	CRSP_ARRAY_T	CRSP array container definition for keyset composing items
items	CRSP_ITM_T,dimension(:)	Array of the items composing the keyset

CRSP_ITM_CAL_T

Calendar descriptor

NAME	Fortran 95 TYPE	DESCRIPTION
calid	integer	Calendar ID
keyset	integer	Keyset number keyset = -1 for global base calendars (non-shifted)
is_shifted	loigcal	True if the calendar day dates were shifted based on company's FYE
calcd	character(LEN=CRSP_CALCD_LEN)	Base calendar name
freqcd	character(LEN=CRSP_CHAR_STRSIZE)	Frequency code of the calendar
cal	CRSP_CAL_T, pointer	Pointer to CRSP calendar container object

CRSP_KEYSET_T

Keyset information

NAME	Fortran 95 TYPE	DESCRIPTION
keyset	integer	Keyset number
keyset_tag	character(LEN=CRSP_NAME_LEN)	Keyset tag name
desc	character(LEN=CRSP_DESC_LEN)	Keyset description

CRSP_ITM_SET_T

Data set descriptor

NAME	Fortran 95 TYPE	DESCRIPTION
set_name	character(LEN=CRSP_NAME_LEN)	Keyset number
path	character(LEN=CRSP_PATHSIZE)	Keyset tag name
root_info	CRSP_ROOT_INFO_T	Database root information

CRSP_ROOT_INFO_T

Database root information

NAME	Fortran 95 TYPE	DESCRIPTION
product_name	character(LEN=CRSP_PROD_NAMESIZE)	Database name
product_code	character(LEN=CRSP_CODE_NAMESIZE)	Database code
version	integer	Version number of db
crt_date	character(LEN=CRSP_DATE_SIZE)	Dates are Dow Mon DD HH:MM:SS YYYY
mod_date	character(LEN=CRSP_DATE_SIZE)	Last modification date of db
cut_date	character(LEN=CRSP_DATE_SIZE)	Cut date of db
binary_type	character(LEN=CRSP_CHAR_STRSIZE)	L (IEEE little-endian) or B (big)
code_version	character(LEN=CRSP_OS_NAMESIZE)	CA97 version

CRSP_VARSTRING_T Type

The varying-length string Fortran 95 derived type CRSP_VARSTRING_T allows for flexible use of expandable character strings and complements the standard fixed-length Fortran 95 character type. The _VARSTRING_T string can be constructed from and converted into a regular fixed-length string. The internals of the derived type handle the necessary storage allocation and provide public functions for basic string-related operations.

The CRSP_VARSTRING_T derived type is implemented in a separate module, `crsp_f_varstring`. This module is automatically included in the `crsp_f_itm_types` module and is available to programs using the CRSP Fortran 95 API.

Fortran 95 MODULE	NAME	Fortran 95 TYPE	DESCRIPTION
crsp_f_varstring	CRSP_VARSTRING_T		varying-length character string

assignment (=)

This interface extends the built-in assignment operator. It allows for construction, assignment, and conversion of a varying-length string, which are handled by internal elemental functions.

INTERFACE:	**assignment(=)** **left = right**
ARGUMENTS1:	type(CRSP_VARSTRING_T) :: left type(CRSP_VARSTRING_T) :: right
ARGUMENTS2:	type(CRSP_VARSTRING_T) :: left character(*) :: right
ARGUMENTS3:	character(*) :: left type(CRSP_VARSTRING_T) :: right
RETURN VALUES:	None
SIDE EFFECTS:	Left variable gets assigned the value of the right variable. The left CRSP_VARSTRING_T variable will be re-initialized to accommodate the string value on the right.
PRECONDITIONS:	Available heap memory necessary for dynamic allocation of the internal string storage.
EXAMPLE:	type(CRSP_VARSTRING_T) ::vstr1, vstr2 character(LEN=10) :: fixstr = '1235' vstr1 = 'TEST' vstr 2 = fixstr vstr1 = vstr2

len

len extends the intrinsic LEN() function to operate on CRSP_VARSTRING_T strings. It returns the current allocated length of the stored string.

INTERFACE:	**LEN(vstr)**
ARGUMENTS:	type(CRSP_VARSTRING_T) :: vstr
RETURN VALUES:	integer :: len – Length of the stored string. If string is not allocated, returned len=0.
SIDE EFFECTS:	None
PRECONDITIONS:	None
EXAMPLE:	type(CRSP_VARSTRING_T) ::vstr integer :: len vstr = 'TEST' len = LEN(vstr)

trim

trim extends the intrinsic TRIM() function to operate on CRSP_VARSTRING_T strings. It returns a [right] blank-trimmed stored string as a fixed-length string.

INTERFACE:	**TRIM(vstr)**
ARGUMENTS:	type(CRSP_VARSTRING_T) :: vstr
RETURN VALUES:	character(*) :: fixstr – Fixed-length string resulting from the stored string with blanks trimmed from the right side.
SIDE EFFECTS:	None
PRECONDITIONS:	None
EXAMPLE:	type(CRSP_VARSTRING_T) ::vstr character(LEN=10) :: fixstr fixstr='TEST' !! is blank-padded to allocated length vstr = fixstr print *,'trimmed_str:[',TRIM(vstr),']'

char

Explicitly converts the stored string into fixed-length string. char is often used on string arguments to output statements (such as PRINT and WRITE).

INTERFACE:	**CHAR(vstr)**
ARGUMENTS:	type(CRSP_VARSTRING_T) :: vstr
RETURN VALUES:	character(*) :: fixstr – Fixed-length string from the stored string
SIDE EFFECTS:	None
PRECONDITIONS:	None
EXAMPLE:	type(CRSP_VARSTRING_T) ::vstr vstr = 'TEST' print *,'vstr:[',CHAR(vstr),']'

crsp_f_vstr_free

crsp_f_vstr_free frees the internally allocated heap storage. It is expected to be called by the user when the CRSP_VARSTRING_T variable goes out of its scope of use, so that the allocated memory is released back to the process heap.

PROTOTYPE:	**pure subroutine crsp_f_vstr_free (str,stat)**
ARGUMENTS:	type(CRSP_VARSTRING_T) :: vstr integer, optional :: stat – Error code to indicate status of the operation: • stat = 0 – SUCCESS • stat = non-zero – FAILED
RETURN VALUES:	None
SIDE EFFECTS:	The error code is set to the stat variable when passed as argument.
PRECONDITIONS:	None
EXAMPLE:	type(CRSP_VARSTRING_T) ::vstr integer :: errcd vstr = 'TEST' call crsp_fvstr_free(vstr,stat=errcd) if (errcd /= 0) stop

crsp_f_vstr_init

crsp_f_vstr_init reserves internal storage to hold a string of the specified length and initializes the reserved string with blanks. Normally an explicit call to this function is not required from user programs, as it is called internally by the defined assignment operators.

PROTOTYPE:	**pure subroutine crsp_f_vstr_init (str,len, stat)**
ARGUMENTS:	type(CRSP_VARSTRING_T) :: vstr – resulting varying-length string integer,intent(in) :: len – Reserved length of the string integer, optional :: stat – Error code to indicate status of the operation: • stat = 0 – SUCCESS • stat = non-zero – FAILED
RETURN VALUES:	None
SIDE EFFECTS:	The error code is set to the stat variable when passed as argument.
PRECONDITIONS:	Available heap memory necessary for dynamic allocation of the internal string storage.
EXAMPLE:	type(CRSP_VARSTRING_T) ::vstr integer :: errcd call crsp_fvstr_init(vstr,10, stat=errcd) if (errcd /= 0) stop