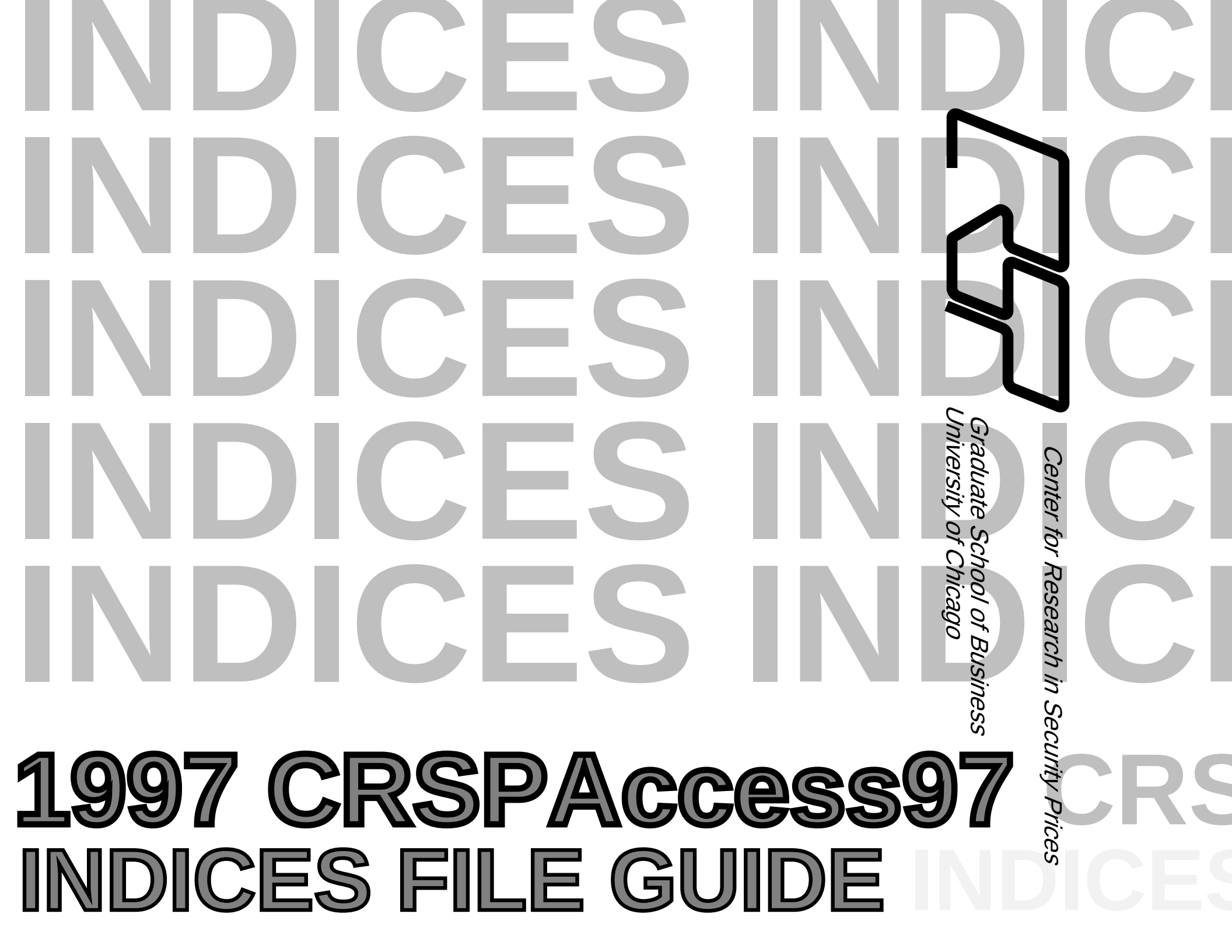


Center for Research in Security Prices
Graduate School of Business
University of Chicago

1997 CRSPAccess97
INDICES FILE GUIDE

CENTER FOR RESEARCH IN SECURITY PRICES

Graduate School of Business

The University of Chicago

CRSPAccess97 Indices File Guide

Data ending December 31, 1997

CRSP Market Indices

Total Market plus Capitalization and

Risk Based Performance Levels

CRSP US Treasury and Inflation Series (CTI)

Center for Research in Security Prices
Graduate School of Business
University of Chicago
725 South Wells Street, Suite 800
Chicago, IL 60607

Telephone: 773.702.7467
FAX: 773.702.3036

E-Mail mail@crsp.uchicago.edu

Internet Address: http://www.crsp.com

To install, follow the instructions in the appendices of the documentation.

IMPORTANT – READ CAREFULLY BEFORE OPENING OR USING DATA PACKETS

COPYRIGHT NOTICE

PROPRIETARY RIGHTS

CRSP DATA LICENSE

DISCLAIMER

CRSP will endeavor to obtain information appearing on its Data Files from sources it considers reliable, but disclaims any and all liability for the truth, accuracy or completeness of the information conveyed. THE UNIVERSITY AND CRSP MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH RESPECT TO THE MERCHANTABILITY, FITNESS, CONDITION, USE OR APPROPRIATENESS FOR SUBSCRIBER'S PURPOSES OF THE DATA FILES AND DATA FURNISHED TO THE SUBSCRIBER UNDER THIS SUBSCRIPTION, OR ANY OTHER MATTER AND ALL SUCH DATA FILES WILL BE SUPPLIED ON AN "AS IS" BASIS. CRSP will endeavor to meet the projected dates for updates, but makes no guarantee thereof, and shall not have any liability for delays, breakdowns or interruption of the subscription. In no event shall the University or CRSP be liable for any consequential damages (even if they have been advised of the possibility of such damages), for damages arising out of third party suppliers terminating agreements to supply information to CRSP, or for other causes beyond its reasonable control.

In the event that the Subscriber discovers an error in the Data Files, Subscriber's sole remedy shall be to notify CRSP and CRSP will use its best efforts to correct same and deliver corrections with the next update.

Version AIND-1997-1.0

NOTICE CONCERNING USE OF THE CRSP DATA

The CRSP data files are proprietary and should be used only for research purposes by the faculty, students, or employees of the subscribing institution**.** The Subscription Agreement, signed by each subscribing institution states:

1. Subscriber acknowledges that the data files to which it is subscribing contain factual material selected, arranged and processed by CRSP and others through research applications and methods involving much time, study, and expense.

2. The Subscriber agrees that it will not transfer, sell, publish, or release in any way any of the data files or the data contained therein to any individual or third party who is not an employee or student of the Subscriber, and that the data provided to the Subscriber by CRSP is solely for the Subscriber's use.

3. Should the Subscriber or any of the employees or students of the Subscriber (collectively referred to herein as "users") wish to utilize CRSP data for any non-academic or profit-making endeavor, said user shall first obtain CRSP's express written consent and agree to pay an applicable royalty fee to CRSP.

4. The Subscriber may not copy the data or documentation in any form onto any device or medium without the express written consent of CRSP, except solely to create back-up copies of the CRSP data files for its internal use, subject to the terms of this Agreement.

5. Subscriber agrees and warrants that it will take all necessary and appropriate steps to protect CRSP's proprietary rights and copyright in the data supplied (including, but not limited to, any and all specific steps which may be expressly required by CRSP), and that the Subscriber will protect the data in no less than the manner in which it would protect its own confidential or proprietary information.

6. The Subscriber will inform all users and potential users of CRSP data of CRSP's proprietary rights in its files and data by giving each user a copy of this paragraph and any other specific requirements CRSP may mandate under this paragraph and by requiring each such user to comply with this paragraph and all such additional requirements.

7. This subscription applies to only one campus or location of a multi-campus or multi-location system; any additional campus or location desiring access to CRSP data files must apply for a separate subscription.

8. The Subscriber agrees that its obligation under the Subscription Agreement shall survive the termination of this Agreement for any reason.

IN RESPONSE TO REQUESTS FROM THE ACADEMIC COMMUNITY, WE HAVE PERMITTED SUBSCRIBERS TO USE THE CRSP DATA FILES ON A FEE BASIS FOR CERTAIN CONSULTING, REGULATORY, AND JUDICIAL APPLICATIONS. FEES ARE BASED ON THE FILES USED AND THE LENGTH OF USAGE. PLEASE CONTACT THE CRSP PRODUCTS AND SERVICES OFFICE (773.702.7467) FOR INFORMATION REGARDING THESE FEES.

The CRSP CDs are intended to be used as a backup only. They should be copied onto your system as soon as you receive them.

See reverse side for Data Replacement Policy ➔

DATA REPLACEMENT POLICY

If any item is missing or damaged, report it immediately. All CRSP files are supplied on certified mediums to assure readability. If you encounter problems reading your CD due to shipping damage or a hard I/O error, CRSP WILL REPLACE YOUR CD FREE OF CHARGE, PROVIDED YOU HAVE FOLLOWED THE FOUR STEPS OUTLINED IN THE YELLOW DELIVERY ACKNOWLEDGMENT CARD WITHIN 30 DAYS OF RECEIPT. To obtain a replacement CD, contact the Products and Services Office at **773.702.7467**. You will be given an RGA Number and asked to return the defective CD along with a letter and/or documentation supporting the problem.

Should you desire a replacement CD for reasons other than shipping damage or hard I/O errors, the fee will be $100.00 per CD. You are entitled to one hardcopy of documentation for each product type. Additional hardcopies will be billed at $20.00 per product per database format. Documentation is available, and updated, on-line through the file guide link at http://www.crsp.com. As a subscriber, you are entitled to make copies of the documentation and to distribute it for internal use only.

Note: Replacement CDs are subject to availability. 1997 Replacement CDs will not be available after January 31, 1999.

TABLE OF CONTENTS

1. INTRODUCTION

Please read this documentation thoroughly before attempting to access the data

1.1 Document Organization

The Introduction outlines the content and development of the files, and highlights recent changes to the files.

The Data Description contains index definitions and descriptions of all data items in the files. The indices are first described by general methodology type. There are four types of indices available, and each is described in its own section with variables based on FORTRAN programming structures. This is followed by descriptions of general indices data structures used with the C programming language and CRSPAccess97 data utilities.

Accessing the Data contains technical information including descriptions of text file layouts, data utility programs, and FORTRAN and C programming access.

The Appendices contain installation instructions for supported systems and text file names.

The Index contains an alphabetical listing of the variables in the data description, and the functions, include files and sample programs available for programming access.

The CRSPAccess97 Indices File / Portfolio Assignments Product can be integrated with CRSPAccess97 Stock products. The **CRSPAccess97 Stock File Users Guide** and the **CRSPAccess97 Stock File Programmers Guide** contain documentation of security data items, data utilities, and programming libraries and sample programs that can be used with index and portfolio data. **CRSPAccess97 Release Notes** contain version-specific counts, calendar ranges, and file sizes for both the CRSPAccess97 Stock and Indices products.

Notational Conventions

- All example commands or file names are printed using a `constant-width, courier,` font.
- All names that occur within CRSP's sample programs and include files are printed using a `constant-width, courier,` font. These names include variable names, parameter names, subroutine names, subprogram names, function names, library names, and keywords. For example, CUSIP refers to the CUSIP Agency identifier, while `CUSIP` refers to the variable that the programs use to store this identifier. CRSP's FORTRAN variable mnemonics, used as names and in the descriptions, are displayed capitalized using a `CONSTANT-WIDTH` font. C variables are typically `lower case,` excepting constants and structure names, which are displayed in `UPPER CASE`.
- All names that refer to the data utilities, sample programs, or include files are printed using an *italic helvetica* font.
- Names of FORTRAN common blocks are delimited by slashes(/ /).
- A FORTRAN one-dimensional array is indicated as `arrayname()` and a FORTRAN two-dimensional array is indicated as `arrayname(,)`. In C, single pointers are indicated as `pointer_name *` and double pointers are indicated as `pointer_name **`. A single pointer usually refers to a one-dimensional array and a double pointer usually refers to a two-dimensional array. C arrays can also be indicated as `arrayname[]`. In the variable definitions section and in usage examples, the variables `I` or `J` are sometimes used in referencing an element in an array variable. In this case, `I` or `J` refers to a possible range of valid data in this array.
- The text of this document is in Times New Roman. *Italics* and **bold** styles are used to emphasize headings, names, definitions and related functions.
- The term `CRSPDB` refers to a CRSPAccess97 database.

1.2 Preface

There are four types of indices provided with the CRSPAccess97 Indices Files. In addition, security portfolio assignment data are provided in association with market segment portfolio groups. Daily and monthly frequency indices can be integrated with CRSPAccess97 Stock files to provide excess returns based on a number of criteria.

CRSP Stock File Indices

The CRSP Stock File Indices are provided for four frequencies: daily, monthly, quarterly, and annual. There are five market groups of securities for which indices are calculated: the individual NYSE, AMEX, and Nasdaq markets, and the NYSE/AMEX and NYSE/AMEX/Nasdaq market combinations. Value weighted returns, with and without dividends, as well as equal-weighted returns, with and without dividends, are calculated for each market.

There are four files for each of the five groups of securities; each file corresponds to the daily, monthly, quarterly, or annual frequency. These twenty files make up the core of the CRSP Indices File. Each of the twenty files provides the four types of returns, corresponding index levels, and summary statistics. Additional series representing performance of market segments created by ranking securities according to market capitalization are provided. These are supplemented with the publicly reported Standard & Poor's 500® Composite Index or the Nasdaq Composite Index.

Four summary statistic risk-based portfolio files are included separately from the twenty main stock indices files. These contain returns and associated index levels for market segments created by ranking securities according to two risk based criteria: beta and standard deviation of return. They are provided only for the NYSE/AMEX combined market and Nasdaq market at the daily frequency.

Only the daily and monthly returns represent true holding period returns. All quarterly and annual series are derived by compounding the monthly values. All CRSP derived index levels are initialized to 100.00 on December 29, 1972 so that the levels may be compared across series. Differences arise between the daily index levels and the index levels of other frequencies due to compounding. Therefore these series are not directly comparable.

The ranges for individual exchange data are listed below. The series containing combinations of exchanges begin at the earliest point that any of the exchanges data are available.

The New York Stock Exchange (NYSE)	monthly, quarterly and annual	begins in December, 1925
The New York Stock Exchange (NYSE)	daily series	begins in July, 1962
The American Stock Exchange (AMEX)	all series	begins in July, 1962
The Nasdaq Stock Market (NASDAQ)	all series	begins in December, 1972

CRSP Cap-Based Portfolios

CRSP Cap-Based Portfolios are monthly series of capitalization-based market segments using a different methodology and universe than the CRSP Stock File Indices. The universe includes all common stocks listed on the NYSE, AMEX, and The Nasdaq National Market excluding Unit Investment Trusts, Closed-End funds, Real Estate Investment Trusts, Americus Trusts, Foreign stocks, and American Depository Receipts. All eligible NYSE companies are ranked by market capitalization on the last trading day of each quarter. Ten equally populated portfolios, or deciles, are then formed. The capitalization of the largest company in each portfolio serves as the breakpoint. When stocks are added that trade on the AMEX and Nasdaq National Market, they are placed in deciles according to their respective market capitalizations using the NYSE breakpoints.

CRSP Indices for the S&P 500®

The CRSP Indices for the S&P 500® contain standard CRSP indices including value and equal weighted returns, with and without dividends, but only for a market of stocks as identified by the S&P 500® Composite Index. Daily data beginning July 2, 1962, and monthly data beginning December 25, 1925 are provided. The published S&P 500® Composite Index and returns are also included for comparison.

CRSP US Treasury and Inflation Series

The CRSP US Treasury and Inflation Series files contain returns and index levels on; 1, 2, 5, 7, 10, 20 and 30 year Fixed US Treasury Bonds, 30 and 90-day Treasury Bills, and the US Government's Consumer Price Index.

1.3 Development of the CRSP Stock Files

The CRSP Stock File Indices are used to derive all the indices in the CRSP Indices File except for CRSP US Treasury and Inflation Series which are generated from the CRSP US Government Bond Files and the US Government-provided Consumer Price Index.

CRSP Stock File Data Dates By Exchange

Exchange	Monthly Stock Files Beginning Date	Daily Stock Files Beginning Date
NYSE	12/31/25	07/02/62
AMEX	07/02/62	07/02/62
Nasdaq	12/29/72	12/14/72

The CRSP Data files were developed by the Center for Research in Security Prices (CRSP), Graduate School of Business, University of Chicago. The CRSP Stock Master File was originally built by Lawrence Fisher, currently at Rutgers University, who originated the basic design and content of the Master File. For a more complete discussion of the original files, see Lawrence Fisher and James H. Lorie, *A Half Century of Returns on Stocks and Bonds*, Chicago: The University of Chicago, Graduate School of Business, 1977, Appendices A and B.

The original CRSP Stock File contained month-end prices and returns from the New York Stock Exchange (NYSE) dating from December, 1925. The monthly American Stock Exchange (AMEX) data beginning in July, 1962 was combined with the NYSE Monthly File to create the current NYSE/AMEX Monthly File. This file contains information on approximately 8,800 securities to date. The NYSE/AMEX Daily File is comprised of the daily prices and returns dating from July 2, 1962, and contains historical data on roughly 8,250 securities.

The Nasdaq Stock File was originally released in 1987 with daily and monthly prices and returns for Nasdaq Stock Exchange domestic common stocks since December 14, 1972, and currently contains information on over 14,800 securities. The Nasdaq Stock File was merged into the NYSE/AMEX Files to create the NYSE/AMEX/Nasdaq Daily and Monthly Stock Files.

Data Sources for the CRSP NYSE/AMEX Files

The data used to construct the original Master File was hand collected by CRSP. Standard & Poor's Price Tape and Punched Card Dividend Service provided the daily price and dividend data between July, 1962 and September 1, 1972. On September 1, 1972, these services were acquired by Interactive Data Corporation (IDC) of Waltham, Massachusetts. IDC continued to provide the data between September, 1972 and April, 1987. Interactive Data Services, Inc. (IDSI), a subsidiary of IDC, has supplied this data since April, 1987. IDSI has additionally provided back data to include high, low, and volume data between July, 1962 and March, 1987.

Data Sources for the CRSP Nasdaq Historical Data File

The Nasdaq File contains information for over 14,800 domestic common stocks traded on The Nasdaq Stock Market since December 14, 1972. Funds for the development of the original Nasdaq file were provided by Dimensional Fund Advisors, Inc. (DFA) and the National Association of Securities Dealers, Inc. (NASD).

CRSP collected machine-readable data from three sources to build the Nasdaq file. Interactive Data Corporation (IDC) of Waltham, Massachusetts, provided data on the daily price quotes and information about capitalization and distributions to shareholders between December 12, 1972 and August 31, 1984. The National Association of Securities Dealers (NASD) provided the data from November 1, 1982 to the present, with the exception of February, 1986 IDSI provided daily price and volume data for February, 1986. IDC was used as a secondary source to NASD between November 1, 1982 and August 31, 1984.

Nasdaq Stock Markets

The Nasdaq Stock Market[SM] is comprised of two subsets of securities, The Nasdaq National Market and The Nasdaq Small Cap Market. Currently, for a security to be designated a Nasdaq National Market security, it must meet criteria setting minimum levels for: annual income, numbers of publicly traded shares, market capitalization, share price, and number of market makers. The requirements for maintaining The Nasdaq National Market status are less stringent than the original listing requirements. All other securities belong to The Nasdaq Small Cap Market. A security may move between The Nasdaq National Market and The Nasdaq Small Cap Market over time as its status changes.

The Nasdaq National Market was initiated in April 1982 for larger and generally more actively traded Nasdaq securities. The Nasdaq National Market Securities must meet higher financial and non-financial criteria than other Nasdaq stocks, and are subject to last-sale reporting. In June of 1992 the regular Nasdaq segment of The Nasdaq Stock Market[SM] was renamed The Nasdaq Small Cap Market and for the first time these became subject to real-time price and volume reporting.

Data Accuracy of the CRSP Stock Files

CRSP Stock Files are designed for research and educational use and have proven to be highly accurate. Considerable resources are expended to improve and to check the quality of the data. The CRSP Stock Files contain over one hundred million prices, distributions and derived data. Errors are not common. Some of the errors found in checking the data are the results of inaccuracies in the initial data source. The inaccuracies are corrected as soon as possible. Other errors are CRSP coding errors; over time these coding errors are found and corrected. Historical corrections account for differences in the data from update to update. The Annual CRSP Stock Files contain updated data through the end of the previous calendar year.

Machine-readable data are checked for internal consistency. Secondary sources including the *CUSIP Directory*, *Moody's Dividend Record*, Commerce Clearing House's *Capital Changes Reporter*, *Directory of Obsolete Securities*, *Moody's Manuals*, the New York Stock Exchange *Weekly Bulletin*, the American Stock Exchange *Weekly Bulletin*, *Bank and Quotation Record*, *The Wall Street Journal*, and *The Commercial and Financial Chronicle* are used to check suspect information. Information not available in machine-readable form is hand-coded and verified.

Development of the CRSP Indices on the S&P 500® Universe

The names and date ranges of the securities included in the S&P 500® universe were matched to corresponding CRSP permanent identifiers, `PERMNO`s. Due to historic differences in handling mergers, reorganizations and other major corporate actions, one S&P 500® range may map to more than one CRSP `PERMNO` and one CRSP `PERMNO` may map to more than one S&P 500® range. Using the CRSP `PERMNO`s and corresponding date ranges, CRSP calculated value- and equal-weighted returns with and without dividends using the same methodology as that used with the CRSP Stock File Indices. The results are not intended to match the S&P 500® index exactly; instead they should provide CRSP users with returns that mirror the S&P 500® but are more comparable to the returns of other CRSP Stock File Indices.

US Treasury and Inflation

These indices were adapted from indices in the CRSP US Government Bond Files and the US Government Consumer Price Index. They contain monthly returns and index levels and are available for the following maturity terms: Treasury Bonds of 1 year, 2 years, 5 years, 7 years, 10 years, 20 years, and 30 years, and Treasury Bills of 30 and 90 days.

1.4 Changes to the CRSP Indices File

Research Changes

- CRSP makes historical edits to the Master Stock Files from year to year, which may result in slight changes to historical indices derived from these files.
- The Ibbotson Stock, Bonds, Bills and Inflation Indices are no longer available in the CRSPAccess97 Indices Files. They have been replaced with the CRSP US Treasury and Inflation Index Series (CTI) for US government bonds. These indices were adapted from indices in the CRSP US Government Bond Files and the US Government Consumer Price Index. The CTI file layout is the same as the previous one used for the SBBI files. The following table shows a comparison between the CTI and SBBI.

CTI `INDNO`	CTI Description	Old `INDNO`	Old Description
1000500, 1000502	CRSP Value-weighted Index on the S&P500 Universe, S&P 500 Composite	1000600	SBBI S&P 500 Index
1000700	CRSP 30-year US Bond Returns	1000601	SBBI Long Term US Treasury Bond Index
n/a	n/a	1000602	SBBI Long Term Corporate Bond Index
1000708	CRSP 30-Day US Bill Returns	1000603	SBBI US Treasury Bill Index
1000709	Consumer Price Index	1000604	SBBI Consumer Price Index
1000353	CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 9-10	1000605	SBBI Small Capitalization Stocks Index

CRSPAccess97

CRSPAccess97 is a fundamental change in the access provided with CRSP Stock and Indices Data. A CRSPAccess97 database provides the following features:

- Binary data, programming libraries, and utility programs are provided for target machines.
- Utilities are provided to dump data and perform namelist searches without programming.
- FORTRAN77 common blocks previously provided by CRSP are supported. Indices series can be loaded directly to programs using CRSP stock data.
- C sample programs and libraries are available. C libraries support random access and indices series and portfolio assignment access is integrated with stock data access.
- Fixed format text files are still available in the format previously provided in the CRSP Indices Files, but no FORTRAN sample programs are provided to access this format.

CRSPAccess97 provides the same basic structure and data items as previous CRSP files for text file users and FORTRAN programmers.

CRSP time series report writer users and C programmers use a new data organization. All index series and groups use the same common structure, and are referenced by a permanent identifier called `indno` assigned by CRSP. An example of an `indno` is 1000000 for the NYSE Value-Weighted Market Index. The same `indno` is used in monthly or daily files. Returns with and without dividends are different data items for the index, and the NYSE Equal-Weighted Market Index is a different `indno` (1000001) with different weighting methodology.

Security portfolio assignment data supplemental to CRSPAccess97 Stock databases becomes available for index portfolio groups, linked to the corresponding indices portfolio assignment data.

The Data Description section contains diagrams of the FORTRAN and C data item structures, and the Accessing the Data section describes data utilities and FORTRAN and C variable usage and access functions.

Additional Indices Items

A new common C index structure is based on indices organized by individual series or portfolio group. Each index is assigned a permanent identifier called `indno` by CRSP. Each index contains header information, and groups include rebalancing breakpoint statistics.

Additional Portfolio Support

New security C portfolio structures allow multiple portfolio statistic and assignment types, each with its own rebalancing calendar. Each portfolio type is linked to a group of portfolio index results. Nine daily and eight monthly portfolio types are available and can be linked with security data in CRSPAccess97 Stock databases. Portfolio data is only available with matching CRSPAccess97 Stock Files.

Excess Returns

Beta and Standard Deviation excess returns are not available as Stock File data items. However, selected excess returns can be created using security assignments in market segment portfolio types to link to associated indices. There are nine daily and eight monthly portfolio types available with the CRSPAccess97 Indices product. If the CRSPAccess97 Indices File is integrated with a CRSPAccess97 Stock file, the *ts_print* application or FORTRAN and C programs can generate selected excess returns using available security and index returns series.

FORTRAN Programming Dates Note

For year-2000 compatibility, dates are stored as YYYYMMDD format in CRSPAccess97 databases. However, to provide backward compatibility with existing CRSP indices text formats, dates are stored in YYMMDD format in the text files provided with this product. FORTRAN programmers using indices subroutines described in Section 3 can control the format of dates with the `DATEFFLAG` variable. If `DATEFFLAG` is set to the default value of zero, dates are loaded in YYMMDD format. If `DATEFFLAG` is set to one, dates are loaded in YYYYMMDD format.

2. DATA DESCRIPTION

This section is broken into six parts:

- formulas used to calculate returns, weights, and index levels;
- descriptions of the methods used and the variables in the CRSP Stock File Indices;
- descriptions of the methods used and the variables in the CRSP Cap-Based Portfolios;
- descriptions of the variables in the CRSP Indices for the S&P 500® universe;
- descriptions of the variables in the CRSP US Treasury and Inflation Series (CTI); and
- descriptions of general C index structures and variables used for all methodologies.

2.1 Derivation of Indices

Formula for Individual Returns Calculations

For a security *n* on a trading day *t*, the return ($r_{n,t}$) is defined as the change in the total dollar value of an investment in that security, over some period of time, per dollar of initial investment.

Returns are calculated as follows:

For time *t* (a trading day or month), let

$r_{n,t}$ = return on purchase at t - 1, sale at *t*
$p_{n,t}$ = last sale price or closing bid/ask average at time *t*
$d_{n,t}$ = cash adjustment for *t*
$f_{n,t}$ = price adjustment factor for *t*

then

$$r_{n,t} = \frac{p_{n,t} \quad f_{n,t} + d_{n,t}}{p_{n,t-1}} - 1$$

A return without dividends is calculated with the same formula, except that ordinary dividends are excluded.

Formulas for Index Returns and Weights

$$R_t = \frac{\sum_n w_{n,t} \quad r_{n,t}}{\sum_n w_{n,t}}$$

The return on a portfolio (R_t) is calculated as the weighted average of the returns for the individual stocks in the portfolio:

In a value-weighted portfolio, the weight $w_{n,t}$ assigned to security *n*'s return is its total market value $v_{n,t}$. CRSP defines the market value of a security ($v_{n,t}$) as the product of its price ($p_{n,t-1}$) and its number of shares outstanding ($s_{n,t-1}$), i.e., $V_{n,t} \ ^{o} \ p_{n,t-1}s_{n,t-1}$.

Income return on a given day is the return without dividends subtracted from the return with dividends.

The total market value of the portfolio

$$v_t = \sum_n v_{n,t} = \sum_n w_{n,t}$$

is given in the USDVAL array in the CRSP Stock File Indices and CRSP Indices for the S&P 500®, and the PRTWGT array in the Cap-Based Portfolios.

The value-weighted returns form the VWRETD and VWRETX arrays in the CRSP Stock File Indices and the CRSP Indices for the S&P 500®. All Cap-Based Portfolio returns are value-weighted.

$$\sum_n w_{n,t} = \sum_n 1 = N_t$$

In an equally-weighted portfolio, $w_{n,t} = 1$ for every stock. Such a portfolio would consist of N stocks, with the same amount invested in each stock. The number of stocks used in the equally-weighted portfolio is given the USDCNT array. The equally-weighted returns form the EWRETD and EWRETX arrays.

Derivation of Index Levels

For the CRSP Stock File Indices, the CRSP Cap-Based Portfolios, and CRSP US Government Treasury and Inflation Indices (CTI) there are index levels associated with each return series. An index level is the value of an investment relative to its value at one fixed point in time. The index levels allow convenient comparison of the relative performance of the different portfolios or asset classes. Differences arise between the daily index levels and the index levels of other frequencies due to compounding; therefore, these series are not directly comparable.

CRSP Stock File Indices and CRSP US Government Treasury and Inflation Indices (CTI) data index levels have been initialized so that December 29, 1972 = 100.00. CRSP Cap-Based index levels have been initialized so that December 31, 1925 = 1.00. The index level for any series at any time after the initial point indicates the value at that time of the initial value invested at the initial point. The index level of a series is set to zero prior to available data. Let:

i_t = index level for any series at time t
r_t = return for the period $t - 1$ to t
t_0 = the time of the first non-missing return of the series
D_0 = 19721229 for CRSP stock indices and CTI indices, 19251231 for Cap-Based indices
V_0 = 100.0 for CRSP stock indices and CTI indices, 1.0 for Cap-Based indices

then

$$\begin{array}{ll} \text{if } t = D_0 & i_t = V_0 \\ \text{if } t > D_0 & i_t = i_{t-1}(1+r_t) \\ \text{if } t < D_0 \text{ and } t \geq t_0 & i_t = i_{t+1} \,/\, (1+r_{t+1}) \\ \text{if } t < t_0 & i_t = 0 \end{array}$$

2.2 CRSP Stock File Indices

The CRSP Stock File Indices are a group of index series based on all issues trading on a defined exchange or group of exchanges. Each set of indices furnishes five types of data:

(1) Trading calendar dates;

(2) Returns and index levels on whole market portfolios;

(3) Summary statistics on these whole market portfolios;

(4) Published S&P 500® and Nasdaq Composite Index data with derived returns;

(5) Total returns and index levels on Market Segment Portfolios.

The daily calendar contains trading days listed sequentially, ignoring weekends and market holidays. The monthly, quarterly and annual calendars contain the last trading day of each period. CRSPAccess97 Release Notes lists the date and subscript ranges of available data for each exchange.

All indices derived from the CRSP Stock Files use the NYSE/AMEX and Nasdaq files' prices, distributions, shares outstanding, and calculated returns. The daily series use one-day holding period returns, while the monthly series use one-month holding period returns. Quarterly and annual indices are compounded from the monthly values.

CRSP derives four whole market indices for each of the three exchanges, for the combined NYSE and AMEX and for all three exchanges together. The four indices are:

(1) Value-Weighted returns including all distributions (`VWRETD`);

(2) Value-Weighted returns excluding dividends (`VWRETX`);

(3) Equally-Weighted returns including all distributions (`EWRETD`); and

(4) Equally-Weighted returns excluding dividends (`EWRETX`).

Summary statistics providing the total market capitalizations and counts of securities are included.

CRSP also furnishes closing levels of the S&P 500® Index (`SPINDX`) and Nasdaq Composite Index (`NCINDX`), and respectively calculates returns for each (`SPRTRN` and `NCRTRN`).

There are three sets of market segment portfolios; one based on capitalization and two based on relative risk. The capitalization portfolios are derived for each of the three exchanges, the NYSE/AMEX combination, and the NYSE/AMEX/Nasdaq combination. The securities on a given exchange or combination of exchanges are ranked according to capitalization and then divided into ten equal parts. Value-weighted total returns and associated index levels are then calculated for each of the ten portfolios. The remaining two sets of market segment portfolios are created for the daily NYSE/AMEX and Nasdaq sets. The securities in the exchange sets are ranked by beta or standard deviation of total returns and divided into ten equal parts. Equal-weighted total returns and associated index levels are then calculated for each set of ten portfolios.

Creation of Market Segment Portfolios

The capitalization portfolios are created by ranking all eligible stocks by capitalization and then dividing them into ten equal portfolios. The portfolios are rebalanced yearly, using the capitalization at the previous year-end. A value-weighted return is calculated on the stocks of each portfolio. The first portfolio contains the lowest capitalization stocks.

There are four files containing daily returns and associated index levels for risk-based portfolios. The portfolios are created by ranking the securities according to a measurement of the risk of their returns. One ranking uses beta values computed using the methods developed by Scholes and Williams (Myron Scholes and Joseph Williams, "Estimating Betas from Nonsynchronous Data", *Journal of Financial Economics*, vol 5, 1977, 309-327). The other ranking uses the annual standard deviation of the daily returns for its ranking. The portfolio assignment is based on the statistic describing the previous year. Portfolio one contains the securities with the highest statistics, and portfolio ten contains the securities with the lowest statistics.

Beta is calculated each year as follows:

$ret_{i,t}$ = log of (1 + return for security i on day t)
$mret_t$ = log of (1 + value-weighted market return on day t)
$mret3_t = mret_{t-1} + mret_t + mret_{t+1}$ (a 3 day moving average market window)
n = number of observations for the year

$$b_i = \frac{\sum_t (ret_{i,t}\ mret3_t) - \left(\frac{1}{n}\right)\left(\sum_t ret_{i,t}\right)\left(\sum_t mret3_t\right)}{\sum_t (mret_t\ mret3_t) - \left(\frac{1}{n}\right)\left(\sum_t mret_t\right)\left(\sum_t mret3_t\right)}$$

where summations over t are over all days on which security i traded, beginning with the second trading day of the year and ending with the second to last trading day of the year.

In the NYSE/AMEX File, only trading prices are considered in the beta calculation and a security must have traded half the days in a year to be included for that year. Trade only returns are also used for calculating the NYSE/AMEX value-weighted market return with dividends in the calculation and in the returns for the ten beta portfolios.

Betas for the Nasdaq Daily File do not use the standard Scholes-Williams trade data only restriction, since most Nasdaq securities were not required to report transactions. Removing bid/ask averages would restrict Nasdaq data to only Nasdaq National Market securities after 1982 and Nasdaq Small Cap securities after June 15, 1992. Nasdaq returns based on bid/ask averages have different characteristics than trade-based returns, and betas are provided for comparison. Nasdaq betas are based on the Nasdaq value-weighted market index.

The formula used in calculating the standard deviation is:

$ret_{i,t}$ = daily return (trade or average of bid and asked) of security i on day t.
n = number of observations for the year (of $ret_{i,t}$)
$s_i \equiv$ yearly standard deviation for the i^{th} company

$$s_i = \frac{\sqrt{\sum_t (ret_{i,t})^2 - \frac{1}{n}\left(\sum_t ret_{i,t}\right)^2}}{n-1}$$

where summation over t is over all returns for the i^{th} company in the given calendar year.

CRSP Stock File Indices FORTRAN Variable Descriptions

Each FORTRAN variable description is preceded with a line containing three items:

1. The variable name followed by parentheses ("()") to indicate the variable is an array or parentheses with an enclosed comma ("(,)") to indicate a two dimensional array;
2. A short description of the variable; and
3. The FORTRAN type for the variable.

CALDT() **Calendar Date** **INTEGER**

CALDT contains the trading dates. These dates are stored in the form YYMMDD (year, month, date). Examples from the files are listed below.

Daily File Examples:
```
CALDT(1)    = 620702 (July 2, 1962)
CALDT(2610) = 721214 (December 14, 1972)
CALDT(8939) = 971231 (December 31, 1997)
CALDT(8939) = CALDT(NDAYS)  = 971231
```

Monthly File Examples:
```
CALDT(1)    = 251231
CALDT(440)  = 620731
CALDT(565)  = 721229
CALDT(865)  = CALDT(NDAYS)  = 971231
```

Quarterly File Examples:
```
CALDT(1)    = 251231
CALDT(153)  = 620731
CALDT(193)  = 721229
CALDT(289)  = CALDT(NDAYS)  = 971231
```

Annual File Examples:
```
CALDT(1)    = 251231
CALDT(38)   = 621231
CALDT(48)   = 721229
CALDT(73)   = CALDT(NDAYS)  = 971231
```

Dates are stored as YYYYMMDD format in CRSPAccess97 databases. However, to provide backward compatibility with existing CRSP indices text formats, dates are stored in YYMMDD format in the text files provided with this product. FORTRAN programmers using indices subroutines described in Section 3 can control the format of dates with the DATEFFLAG variable. If DATEFFLAG is set to the default value of zero, dates are loaded in YYMMDD format. If DATEFFLAG is set to one, dates are loaded in YYYYMMDD format.

VWRETD() **Value-Weighted Return (including all distributions)** **REAL**
VWINDD() **Index Level Associated With VWRETD** **REAL**

VWRETD contains the returns, including all distributions, on a value-weighted market portfolio (excluding ADRs.)

VWRETX() **Value-Weighted Return (excluding dividends)** **REAL**
VWINDX() **Index Level Associated With VWRETX**

VWRETX contains the returns, excluding all ordinary distributions and all taxable distribution, on a value-weighted portfolio (excluding ADRs).

EWRETD() **Equal-Weighted Return (including all distributions)** **REAL**
EWINDD() **Index level associated with EWRETD** **REAL**

EWRETD contains the returns, including all distributions, on an equally-weighted market portfolio (including ADRs).

EWRETX() **Equal-Weighted Return (excluding dividends)** **REAL**
EWINDX() **Index Level Associated With EWRETX** **REAL**

EWRETX contains the returns, excluding all ordinary distributions and all taxable distributions, on an equally-weighted market portfolio (including ADRs).

SPINDX() **Level of the S&P 500® Index*** **REAL**

SPINDX(I) is the level of the S&P 500® index at the end of the trading day.

SPRTRN() **Return on the S&P 500® Index*** **REAL**

SPRTRN(I) is the return on the S&P 500® index for period I as defined as:

$$\frac{SPINDX(I)}{SPINDX(I-1)} - 1$$

NCINDX() **Nasdaq Composite Index*** **REAL**

NCINDX(I) contains the closing quote for the Nasdaq Composite Index.

NCRTRN() **Return on the Nasdaq Composite Index*** **REAL**

NCRTRN(I) is the return on the Nasdaq Composite Index for period I.

DECRET(,) **Decile Return** **REAL**
DECIND(,) **Index Level Associated With DECRET** **REAL**

DECRET(I,J) is the return on the I^{th} decile portfolio on the date CALDT(J).

TOTVAL() **Total Market Value** **REAL**

TOTVAL(I) contains the total market values, in $1000's of all non-ADR securities with valid prices[1] on the date CALDT(I).

TOTCNT() **Total Market Count** **INTEGER**

TOTCNT(I) is the number of stocks on the exchange or combination of exchanges with a valid price on the date CALDT(I).

---

* NCINDX and NCRTRN are EQUIVALENCEd to SPINDX and SPRTRN. Only the four Nasdaq files, xSIO.97C, will be filled with NCINDX and NCRTRN. The other files are filled with SPINDX and SPRTRN.

[1] A valid price in the file is either a bid/ask average or a closing price.

USDVAL() **Market Value of Securities Used** **REAL**

USDVAL(I) is the total market value, in $1000's, of all securities that are used in the value-weighted indices on date CALDT(I). To be used for value-weighting, a security cannot be an ADR and must have valid prices on the current day and the previous trading day or month-end. The value of USDVAL in the Quarterly and Annual Files is the value from the corresponding Monthly files.

USDCNT() **Count of Securities Used** **INTEGER**

USDCNT(I) is the number of stocks on the exchange or combination of exchanges used in the equal-weighted indices on date CALDT(I). To be used for equal-weighting, a security must have valid prices on the current day and the previous trading day or month-end. The value of USDCNT in the Quarterly and Annual Files is the count from the corresponding Monthly Files.

2.3 CRSP Cap-Based Portfolios

CRSP Cap-Based Portfolios are derived from CRSP Stock Files but use a different methodology than CRSP Stock File Indices. The universe and partitioning rules are different, but the method for calculating returns, weights, and index levels is the same as described in Section 2.1. Cap-Based Portfolio indices are monthly series based on portfolios rebalanced quarterly. Monthly history and quarterly rebalancing history files are provided.

The universe includes all common stocks listed on the NYSE, AMEX, and The Nasdaq National Market excluding:

- Unit Investment Trusts, Closed-End Funds, Real Estate Investment Trusts, Americus Trusts, Foreign Stocks, and American Depository Receipts.

All eligible companies listed on the NYSE are ranked by market capitalization and then split into ten equally populated groups, or deciles. The capitalization of the largest company in each decile serves as the breakpoint for that decile. When multiple issues of a company trade, the sum of the issue capitalizations is used for the company capitalization so that the company is counted only once. The portfolios are reformed every quarter using the price and shares at the end of the previous quarter.

The largest companies are placed in portfolio 1, the smallest in portfolio 10. In addition to the 10 individual portfolios, CRSP produces a single return number for the portfolio formed by combining: Deciles 1 and 2 to create CRSP 1-2; Deciles 3, 4 and 5 to create CRSP 3-5; Deciles 1 through 5 to create CRSP 1-5, Deciles 6, 7 and 8 to create CRSP 6-8; Deciles 9 and 10 to create CRSP 9-10; Deciles 6 through 10 to create CRSP 6-10; Deciles 1 through 10 form the market portfolio.

There are three series based on exchange:

- NYSE only.
- NYSE and AMEX. AMEX data are added beginning July, 1962
- NYSE + AMEX + The Nasdaq National Market. The Nasdaq National Market data are added beginning April, 1982.

Breakpoints for all three series are based exclusively on companies with issues traded on the NYSE. In the second and third series, non-NYSE companies are assigned to appropriate portfolios according to their capitalization in relation to the decile breakpoints.

Companies becoming eligible or ineligible during a quarter are handled with the following rules:

- Securities added during a quarter are assigned to appropriate portfolios when two consecutive month-end prices are available.
- When the last price is a month-end price, that month's return is included in the portfolios' quarterly return
- When the month-end price is missing, we derive a replacement month-end value from merger terms, regional exchanges, etc. If the derived replacement month-end price is not available, the last available daily price is used.
- If an issue becomes ineligible for an index in the middle of a quarter but is still active, such as after an exchange change or because the issue is leaving the Nasdaq National Market, the issue is considered held until the end of the month and then dropped.

CRSP Cap-Based Reports (Monthly) History FORTRAN Variable Descriptions

CALDT() **Calendar Date** **INTEGER**

CALDT contains the trading dates. These dates are stored in the form YYMMDD (year, month, date). Examples from the files are listed below.

```
CALDT(1)      = 251231
CALDT(440)    = 620731
CALDT(565)    = 721229
CALDT(865)    = CALDT(NDAYS)  = 971231
```

Dates are stored as YYYYMMDD format in CRSPAccess97 databases. However, to provide backward compatibility with existing CRSP indices text formats, dates are stored in YYMMDD format in the text files provided with this product. FORTRAN programmers using indices subroutines described in Section 3 can control the format of dates with the DATEFFLAG variable. If DATEFFLAG is set to the default value of zero, dates are loaded in YYMMDD format. If DATEFFLAG is set to one, dates are loaded in YYYYMMDD format.

PRTNAM() **Portfolio Name** **CHARACTER*4**

PRTNAM(I) contains a short character string describing portfolio I. The first ten based on NYSE deciles are numbered 1 - 10, with the largest companies in portfolio 1 and the smallest in portfolio 10. Portfolios 11 - 17 are composites of the first ten. The sequence of the portfolios is listed in the following table.

PRTNAM	Name	Sequence
1	CRSP 1	1
2	CRSP 2	2
3	CRSP 3	3
4	CRSP 4	4
5	CRSP 5	5
6	CRSP 6	6
7	CRSP 7	7
8	CRSP 8	8
9	CRSP 9	9
10	CRSP 10	10
1_2	CRSP 1-2	11
3_5	CRSP 3-5	12
6_8	CRSP 6-8	13
9_10	CRSP 9-10	14
1_5	CRSP 1-5	15
6_10	CRSP 6-10	16
Mrkt	CRSP Market	17

PRTCNT(,) **Portfolio Issue Count** **INTEGER**

PRTCNT(I,J) contains the number of issues included in portfolio I on date CALDT(J) for a CRSP Cap-Based portfolio. I is the portfolio sequence number described under PRTNAM.

PRTWGT(,) **Portfolio Weight** **REAL*8**

PRTWGT(I,J) contains the total capitalization in 1000's of dollars included in portfolio I on date CALDT(J) for a CRSP Cap-Based portfolio. I is the portfolio sequence number described under PRTNAM. Weight is as of the end of the previous month.

TOTRET(,) **Total Return On Portfolio** **REAL**
TOTIND(,) **Index Level Associated With TOTRET** **REAL**

TOTRET(I,J) is the total return on the I^{th} portfolio on the date CALDT(J) for a CRSP Cap-Based portfolio. TOTIND(I,J) is the index level for the I^{th} portfolio on the date CALDT(J). I is the portfolio sequence number described under PRTNAM.

CAPRET(,) **Capital Appreciation On Portfolio** **REAL**
CAPIND(,) **Index Level Associated With CAPRET** **REAL**

CAPRET(I,J) is the capital appreciation on the I^{th} portfolio on the date CALDT(J) for a CRSP Cap-Based portfolio. CAPIND(I,J) is the index level for the I^{th} portfolio on the date CALDT(J). I is the portfolio sequence number described under PRTNAM.

INCRET(,) **Income Return On Portfolio** **REAL**
INCIND(,) **Index Level Associated With INCRET** **REAL**

INCRET(I,J) is the income return on the I^{th} portfolio on the date CALDT(J) for a CRSP Cap-Based portfolio. INCIND(I,J) is the index level on the I^{th} portfolio on the date CALDT(J). I is the portfolio sequence number described under PRTNAM.

CRSP Cap-Based Reports Rebalancing History FORTRAN Variable Descriptions

YYMM() **Year And Month of Quarter** **INTEGER**

YYMM contains the year and month of the quarter. A quarter is labeled by the last month in the quarter. The CRSP quarterly calendar used with Cap-Based rebalancing histories begins in March, 1926, the end of the first quarter containing complete rebalanced data rather than at the inception of the file, December 31, 1925. Rebalancing is based on data at end of the previous quarter. For instance, the March, 1926 portfolios were formed using data from December 31, 1925 (251231). Examples from the files are listed below. The number in parenthesis following YYMM below

```
YYMM(1)      = 2603
YYMM(289)    = 9712
```

PRTNO(,) **Decile Portfolio Number** **INTEGER**

PRTNO(I,J) is the decile portfolio number of the I^{th} CRSP Cap-Based portfolio at the beginning of quarter YYMM(J). There are 10 portfolios based on NYSE deciles included, with the largest companies in portfolio 1 and the smallest in portfolio 10. PRTNO(I,J) is always equal to I.

PRTCCT(,) **Portfolio Company Count** **INTEGER**

PRTCCT(I,J) is the number of companies included in the I^{th} CRSP Cap-Based portfolio at the beginning of quarter YYMM(J). There are 10 portfolios based on NYSE deciles included, with the largest companies in portfolio 1 and the smallest in portfolio 10.

MINCWT(,) **Capitalization of Smallest Company In Portfolio** **INTEGER**

MINCWT(I,J) is the capitalization of the smallest company included in the I^{th} CRSP Cap-Based portfolio at the beginning of quarter YYMM(J).

MINCNM(,) **Name of Smallest Company In Portfolio** **CHARACTER*32**

MINCNM(I,J) is the name of the smallest company included in the I^{th} CRSP Cap-Based portfolio at the beginning of quarter YYMM(J).

MAXCWT(,) **Capitalization of Largest Company In Portfolio** **INTEGER**

MAXCWT(I,J) is the capitalization of the largest company included in the I^{th} CRSP Cap-Based portfolio at the beginning of quarter YYMM(J).

MAXCNM(,) **Name of Largest Company In Portfolio** **CHARACTER*32**

MAXCNM(I,J) is the name of the largest company included in the I^{th} CRSP Cap-Based portfolio at the beginning of quarter YYMM(J).

2.4 CRSP Indices on the S&P 500® Universe

CRSP Indices for the S&P 500® universe, formerly the S&P 90®, are standard CRSP indices derived from CRSP Stock Files but include only the S&P 500® universe. Issue returns, weighting, and calculation of index returns are described in Section 2.1.

The CRSP Indices on the S&P 500® series contain value- and equal-weighted returns with and without dividends for a market of stocks in the S&P 500® universe. Daily data beginning July 2, 1962, and monthly data beginning December 25, 1925 are provided. The published S&P 500® index and returns are also included for comparison.

Prior to March, 1957, the index contains 90 issues. CRSP does not have data for two securities that were part of the S&P 90® at different times between 1925 and 1931.

Company Name	Start Date	End Date
INT'L MERCANTILE MARINE PFD.	31-dec-1925	22-jul-1929
STANDARD POWER & LIGHT "B"	06-feb-1930	16-nov-1931

Due to differences in handling mergers, reorganizations, and other major corporate actions, CRSP data and the S&P 500® universe do not always have a one-to-one mapping. In some cases this results in a short period where CRSP is missing prices or has multiple prices.

USDCNT is not always 500 (90 prior to March 1957) due to missing prices. Known reasons for missing prices are when-issued trading, halts, and suspensions.

CRSP Indices on the S&P 500® FORTRAN Variable Descriptions

CALDT() **Calendar Date** **INTEGER**

CALDT contains the trading dates for the file. These dates are stored in the form YYMMDD (year, month, date). Examples from the daily file and monthly file are listed below.

Daily File Examples:

```
CALDT(1)    =  620702 (July 2, 1962)
CALDT(8939) =  971231 (December 31, 1997)
```

Monthly File Examples:

```
CALDT(1)    =  251231
CALDT(865)  =  CALDT (NDAYS) = 971231
```

Dates are stored as YYYYMMDD format in CRSPAccess97 databases. However, to provide backward compatibility with existing CRSP indices text formats, dates are stored in YYMMDD format in the text files provided with this product. FORTRAN programmers using indices subroutines described in Section 3 can control the format of dates with the DATEFFLAG variable. If DATEFFLAG is set to the default value of zero, dates are loaded in YYMMDD format. If DATEFFLAG is set to one, dates are loaded in YYYYMMDD format.

VWRETD() **Value-Weighted Return (including all distributions)** **REAL**

VWRETD contains either the daily or monthly returns, including all distributions, on a value-weighted market portfolio of all securities in the S&P 500® universe.

VWRETX() **Value-Weighted Return (excluding dividends)** **REAL**

VWRETX contains either the daily or monthly returns, excluding all ordinary distributions and all taxable distributions, on a value-weighted market portfolio of all securities in the S&P 500® universe.

EWRETD() **Equal-Weighted Return (including all distributions)** **REAL**

EWRETD contains either the daily or monthly returns, including all distributions, on an equally-weighted market portfolio of all securities in the S&P 500® universe.

EWRETX() **Equal-Weighted Return (excluding dividends)** **REAL**

EWRETX contains either the daily or monthly returns, excluding all ordinary distributions and all taxable distributions, on an equally-weighted market portfolio of all securities in the S&P 500® universe.

TOTVAL() **Total Market Value** **REAL**

TOTVAL(I) contains the total market values, in $1000's, of all securities in the S&P 500® universe with valid prices[2] on day I.

TOTCNT() **Total Market Count** **INTEGER**

TOTCNT(I) is the number of stocks in the S&P 500® universe assigned a CRSP PERMNO on day I.

USDVAL() **Market Value of Securities Used** **REAL**

USDVAL(I) is the total market value, in $1000's, of all securities that are used in the value-weighted indices on day I. To be used for a value-weighting, a security must have valid prices on the current day and the previous trading day.

USDCNT() **Count of Securities Used** **INTEGER**

USDCNT(I) is the number of stocks in the index used in the equal-weighted indices on day I. To be used for equal-weighting, a security must have valid prices on the current day and the previous trading period I-1.

SPINDX() **Level of the S&P 500® Index** **REAL**

SPINDX(I) is the level of the S&P 500® index (prior to March 1957, 90-stock index) at the end of the trading day or month. This data is collected from publicly available sources such as The Wall Street Journal or Standard & Poor's Statistical Service. SPINDX(I) does not included dividends. The index indicates the change in price of the component securities.

SPRTRN() **Return on the S&P 500® Index** **REAL**

SPRTRN(I) is the return on the S&P 500® index for period I as defined as:

$$\frac{SPINDX(I)}{SPINDX(I-1)} - 1$$

[2] A valid price in the file is either a negative bid/ask average or a positive closing price.

2.5 CRSP US Treasury and Inflation Series (CTI)

The CRSP US Treasury and Inflation Series (CTI) Files are provided on a monthly frequency. The series contains returns adapted from the CRSP US Government Bond Fixed Term Index Series, the CRSP Risk Free Rates File, and the US Government Consumer Price Index. These derived files offer 10 groups of indices: 30 year, 20 year, 10 year, 7 year, 5 year, 2 year, 1 year, 90 day, and 30 day target maturity indices, as well as the Consumer Price Index.

For the 30, 20, 10, 7, 5, 2, and 1 year fixed term indices, a valid issue that best represents each term is chosen at the end of each month for each of the seven fixed terms and held through the next month. Valid issues are at least six months from the target maturity date and are fully taxable. The selection process consists of selecting a representative bond from each of the fixed term groups by filtering available issues on the basis of their characteristics. First, a non-callable, non-flower bond that is closest to the target maturity of its group and fully taxable is found. If more than one security meets these criteria, the one most recently issued is used. If there is no other suitable issue, a second pass is made where flower bonds are accepted. Due to the unavailability of consistently suitable issues for maturities before 1950, the series contain missing values before the starting dates given in the individual variable descriptions below.

The issue selected in the 30 day series is the Treasury Bill closest to but with not less than 30 days to maturity. The 90 day series uses a 90 day target. For these two series, where bills were not available certificates and, in a few cases, notes were used. The selection amongst alternatives was somewhat subjective in early periods. The issue with the maturity closest to target was sometimes rejected because the quotes were suspicious. In no case was an issue used which did not mature on its next coupon payment date. Also excluded were issues with bid quotations implying negative yields. This resulted in some very short nominally 90 day maturities prior to 1942. Similarly, scarcity of available issues results in some very long nominal one month issues being used prior to 1937. The range of maturities of both series after 1942 is within a few days of the targets. Users may wish to restrict their usage to this period.

Each monthly return is calculated as price change plus interest, divided by last month's price. The returns and corresponding index values are set to –99 for months in which a return cannot be calculated, i.e. if the price is missing for either this month or last month, or if no valid issue was available.

The issue chosen for the 30, 20, 10, 7, 5, 2, and 1 year Fixed Term Index series for a given CALDT was selected based on its length to maturity as of the CALDT. The returns contained in these series are calculated under the assumption that the relevant issue is bought one month prior to the CALDT and sold on the CALDT.

The issue chosen for the 90 and 30 day Treasury Bill series on a given CALDT was selected based on its length to maturity as of the month immediately prior to the CALDT. The 90 and 30 day series returns were calculated on the basis of buying the relevant issue one month prior to the CALDT and selling it on the CALDT. For example, a 90 day bill return is calculated between a date approximately 90 days prior to the bill's maturity and the CALDT, which is a month after this date. Likewise, a 30 day bill return is calculated between a date approximately 30 days prior to the bill's maturity and the CALDT, which is a date one month later. In cases where the CALDT chronologically approached or exceeded the maturity date, thereby making a final price unavailable, the return was calculated based on a final price of $100.

The associated index levels of the CRSP US Treasury and Inflation Series all have been initialized so that December 29, 1972 (19721229) equals 100. This facilitates comparison between the CTI indices and Stock File Indices.

CTI FORTRAN Variable Descriptions

CALDT() **Last Trading Date of Period (YYMMDD)** **INTEGER**

CALDT contains the trading dates. These dates are stored in the form YYMMDD (year, month, date). Examples from the files are listed below.

Monthly File Examples:

```
CALDT(1)    =  251231
CALDT(865)  =  CALDT(NDAYS)  = 971231
```

Quarterly File Examples:

```
CALDT(1)    =  251231
CALDT(2)    =  260331
```

Annual File Examples:

```
CALDT(1)    =  251231
CALDT(2)    =  261231
```

Dates are stored as YYYYMMDD format in CRSPAccess97 databases. However, to provide backward compatibility with existing CRSP indices text formats, dates are stored in YYMMDD format in the text files provided with this product. FORTRAN programmers using indices subroutines described in Section 3 can control the format of dates with the DATEFFLAG variable. If DATEFFLAG is set to the default value of zero, dates are loaded in YYMMDD format. If DATEFFLAG is set to one, dates are loaded in YYYYMMDD format.

B30RET () **30 Year Bond Returns** **REAL**
B30IND () **Index Level Associated With B30RET** **REAL**

The 30 Year Bond Returns begin November 29, 1941. Values prior to this date are set to –99.

B20RET () **20 Year Bond Returns** **REAL**
B20IND () **Index Level Associated With B20RET** **REAL**

The 20 Year Bond Returns begin January 31, 1942. Values prior to this date are set to –99.

B10RET () **10 Year Bond Returns** **REAL**
B10IND () **Index Level Associated With B10RET** **REAL**

The 10 Year Bond Returns begin May 31, 1941. Values prior to this date are set to –99.

B7RET () **7 Year Bond Returns** **REAL**
B7IND () **Index Level Associated With B7RET** **REAL**

The 7 Year Bond Returns begin April 30, 1941. Values prior to this date are set to –99.

B5RET () **5 Year Bond Returns** **REAL**
B5IND () **Index Level Associated With B5RET** **REAL**

The 5 Year Bond Returns begin April 30, 1941. Values prior to this date are set to –99.

B2RET () **2 Year Bond Returns** **REAL**
B2IND () **Index Level Associated With B2RET** **REAL**

The 2 Year Bond Returns begin January 31, 1941. Values prior to this date are set to –99.

B1RET ()	**1 Year Bond Returns**	**REAL**
B1IND ()	**Index Level Associated With B1RET**	**REAL**

The 1 Year Bond Returns begin January 31, 1941. Values prior to this date are set to –99.

T90RET ()	**90 Day Bill Returns**	**REAL**
T90IND ()	**Index Level Associated With T30RET**	**REAL**

The scarcity of available issues prior to 1942 resulted in some very short nominally 90 day maturities. The range of maturities after 1942 is within a few days of the 90 day target, and users may therefore wish to restrict their usage of T90RET and T90IND to this period.

T30RET ()	**30-Day Bill Returns**	**REAL**
T30IND ()	**Index Level Associated With T90RET**	**REAL**

The scarcity of available issues prior to 1937 resulted in the use of some very long nominal one month issues. The range of maturities after 1937 is within a few days of the 30 day target, and users may wish to restrict their usage of T30RET and T30IND to this period.

CPIRET()	**Rate of Change In Consumer Price Index**	**REAL**
CPIIND()	**Index Level Associated With CPI**	**REAL**

The Consumer Price Index for All Urban Consumers, not seasonally adjusted (CPI-U NSA), is utilized to measure inflation, which is the rate of change of prices of consumer goods. The inflation measures are constructed by the US Department of Labor, Bureau of Labor Statistics.

2.6 CRSPAccess97 Index Structure and Data Items

The CRSPAccess97 Index structure is organized by individual index rather than methodology set. This section applies to the *ts_print* report writer and C programming.

In CRSPAccess97 databases, all indexes are assigned a permanent identification number called `indno`, and have common data items. The data items include identifying and descriptive information, index composition information, characteristic counts and weights, result returns and index levels based on total returns, and capital appreciation, and dividend yields. Not all of these items are available for each index.

There are two types of index sets supported:

1. `series` – all `indno`s represent a single index, such as the CRSP NYSE Value-Weighted Market Index , the CRSP NYSE Equal-Weighted Market Index, or the CRSP 9-10 Cap-Based index for the NYSE/AMEX/Nasdaq National Market. Each index is an independent portfolio.

2. `group` – all `indno`s represent a group of related series based on the same rules, such as all the portfolios in the CRSP Cap-Based index for the NYSE/AMEX/Nasdaq National Market. All group `indnos` are tied to CRSP Stock portfolio types. These types can be used to determine which index series in the group is applicable to each security at any given point in time. Each series within the group is also available in the `series` set under its own `indno`.

An index must be referenced by a set identification number, `setid`, and index identifier, `indno`. The `setid` is based on the frequency of the time series and whether the `indno` refers to a `series` or `group`. An `indno` will be present in either a `series` or a `group` but not both. However, the same `indno` can appear in daily and monthly sets. The following set identifiers are available:

SETID	Description
400	Monthly frequency `groups`
420	Monthly frequency `series`
440	Daily frequency `groups`
460	Daily frequency `series`

Each CRSPAccess97 database directory contains a header file named `headind.dat` with a list of `setid`s and `indno`s available in that index. Utility functions *dindsearch* and *mindsearch* can be used to search the list for `indno`s and `setid`s. The complete list of available `indno`s is in the following table. See sections 2.1 through 2.5 for the methodologies used.

CRSP Indices

Index	indno	Daily setid	Monthly setid
CRSP NYSE Value-Weighted Market Index	1000000	460	420
CRSP NYSE Equal-Weighted Market Index	1000001	460	420
CRSP NYSE Market Capitalization Decile 1	1000002	460	420
CRSP NYSE Market Capitalization Decile 2	1000003	460	420
CRSP NYSE Market Capitalization Decile 3	1000004	460	420
CRSP NYSE Market Capitalization Decile 4	1000005	460	420
CRSP NYSE Market Capitalization Decile 5	1000006	460	420
CRSP NYSE Market Capitalization Decile 6	1000007	460	420
CRSP NYSE Market Capitalization Decile 7	1000008	460	420
CRSP NYSE Market Capitalization Decile 8	1000009	460	420
CRSP NYSE Market Capitalization Decile 9	1000010	460	420
CRSP NYSE Market Capitalization Decile 10	1000011	460	420
CRSP NYSE Market Capitalization Deciles	1000012	440	400
CRSP AMEX Value-Weighted Market Index	1000020	460	420
CRSP AMEX Equal-Weighted Market Index	1000021	460	420
CRSP AMEX Market Capitalization Decile 1	1000022	460	420
CRSP AMEX Market Capitalization Decile 2	1000023	460	420
CRSP AMEX Market Capitalization Decile 3	1000024	460	420
CRSP AMEX Market Capitalization Decile 4	1000025	460	420
CRSP AMEX Market Capitalization Decile 5	1000026	460	420
CRSP AMEX Market Capitalization Decile 6	1000027	460	420
CRSP AMEX Market Capitalization Decile 7	1000028	460	420
CRSP AMEX Market Capitalization Decile 8	1000029	460	420
CRSP AMEX Market Capitalization Decile 9	1000030	460	420
CRSP AMEX Market Capitalization Decile 10	1000031	460	420
CRSP AMEX Market Capitalization Deciles	1000032	440	400
CRSP NYSE/AMEX Value-Weighted Market Index	1000040	460	420
CRSP NYSE/AMEX Equal-Weighted Market Index	1000041	460	420
CRSP NYSE/AMEX Market Capitalization Decile 1	1000042	460	420
CRSP NYSE/AMEX Market Capitalization Decile 2	1000043	460	420
CRSP NYSE/AMEX Market Capitalization Decile 3	1000044	460	420
CRSP NYSE/AMEX Market Capitalization Decile 4	1000045	460	420
CRSP NYSE/AMEX Market Capitalization Decile 5	1000046	460	420
CRSP NYSE/AMEX Market Capitalization Decile 6	1000047	460	420
CRSP NYSE/AMEX Market Capitalization Decile 7	1000048	460	420
CRSP NYSE/AMEX Market Capitalization Decile 8	1000049	460	420
CRSP NYSE/AMEX Market Capitalization Decile 9	1000050	460	420
CRSP NYSE/AMEX Market Capitalization Decile 10	1000051	460	420
CRSP NYSE/AMEX Market Capitalization Deciles	1000052	440	400
CRSP Nasdaq Value-Weighted Market Index	1000060	460	420
CRSP Nasdaq Equal-Weighted Market Index	1000061	460	420
CRSP Nasdaq Market Capitalization Decile 1	1000062	460	420
CRSP Nasdaq Market Capitalization Decile 2	1000063	460	420
CRSP Nasdaq Market Capitalization Decile 3	1000064	460	420
CRSP Nasdaq Market Capitalization Decile 4	1000065	460	420
CRSP Nasdaq Market Capitalization Decile 5	1000066	460	420
CRSP Nasdaq Market Capitalization Decile 6	1000067	460	420
CRSP Nasdaq Market Capitalization Decile 7	1000068	460	420
CRSP Nasdaq Market Capitalization Decile 8	1000069	460	420
CRSP Nasdaq Market Capitalization Decile 9	1000070	460	420
CRSP Nasdaq Market Capitalization Decile 10	1000071	460	420
CRSP Nasdaq Market Capitalization Deciles	1000072	440	400
CRSP NYSE/AMEX/Nasdaq Value-Weighted Market Index	1000080	460	420
CRSP NYSE/AMEX/Nasdaq Equal-Weighted Market Index	1000081	460	420
CRSP NYSE/AMEX/Nasdaq Market Capitalization Decile 1	1000082	460	420
CRSP NYSE/AMEX/Nasdaq Market Capitalization Decile 2	1000083	460	420
CRSP NYSE/AMEX/Nasdaq Market Capitalization Decile 3	1000084	460	420
CRSP NYSE/AMEX/Nasdaq Market Capitalization Decile 4	1000085	460	420
CRSP NYSE/AMEX/Nasdaq Market Capitalization Decile 5	1000086	460	420
CRSP NYSE/AMEX/Nasdaq Market Capitalization Decile 6	1000087	460	420
CRSP NYSE/AMEX/Nasdaq Market Capitalization Decile 7	1000088	460	420

CRSP Indices *(Cont.)*

Index	indno	Daily setid	Monthly setid
CRSP NYSE/AMEX/Nasdaq Market Capitalization Decile 8	1000089	460	420
CRSP NYSE/AMEX/Nasdaq Market Capitalization Decile 9	1000090	460	420
CRSP NYSE/AMEX/Nasdaq Market Capitalization Decile 10	1000091	460	420
CRSP NYSE/AMEX/Nasdaq Market Capitalization Deciles	1000092	440	400
CRSP NYSE/AMEX Beta Decile 1	1000102	460	-
CRSP NYSE/AMEX Beta Decile 2	1000103	460	-
CRSP NYSE/AMEX Beta Decile 3	1000104	460	-
CRSP NYSE/AMEX Beta Decile 4	1000105	460	-
CRSP NYSE/AMEX Beta Decile 5	1000106	460	-
CRSP NYSE/AMEX Beta Decile 6	1000107	460	-
CRSP NYSE/AMEX Beta Decile 7	1000108	460	-
CRSP NYSE/AMEX Beta Decile 8	1000109	460	-
CRSP NYSE/AMEX Beta Decile 9	1000110	460	-
CRSP NYSE/AMEX Beta Decile 10	1000111	460	-
CRSP NYSE/AMEX Beta Deciles	1000112	440	-
CRSP NYSE/AMEX Standard Deviation Decile 1	1000122	460	-
CRSP NYSE/AMEX Standard Deviation Decile 2	1000123	460	-
CRSP NYSE/AMEX Standard Deviation Decile 3	1000124	460	-
CRSP NYSE/AMEX Standard Deviation Decile 4	1000125	460	-
CRSP NYSE/AMEX Standard Deviation Decile 5	1000126	460	-
CRSP NYSE/AMEX Standard Deviation Decile 6	1000127	460	-
CRSP NYSE/AMEX Standard Deviation Decile 7	1000128	460	-
CRSP NYSE/AMEX Standard Deviation Decile 8	1000129	460	-
CRSP NYSE/AMEX Standard Deviation Decile 9	1000130	460	-
CRSP NYSE/AMEX Standard Deviation Decile 10	1000131	460	-
CRSP NYSE/AMEX Standard Deviation Deciles	1000132	440	-
CRSP Nasdaq Beta Decile 1	1000142	460	-
CRSP Nasdaq Beta Decile 2	1000143	460	-
CRSP Nasdaq Beta Decile 3	1000144	460	-
CRSP Nasdaq Beta Decile 4	1000145	460	-
CRSP Nasdaq Beta Decile 5	1000146	460	-
CRSP Nasdaq Beta Decile 6	1000147	460	-
CRSP Nasdaq Beta Decile 7	1000148	460	-
CRSP Nasdaq Beta Decile 8	1000149	460	-
CRSP Nasdaq Beta Decile 9	1000150	460	-
CRSP Nasdaq Beta Decile 10	1000151	460	-
CRSP Nasdaq Beta Deciles	1000152	440	-
CRSP Nasdaq Standard Deviation Decile 1	1000162	460	-
CRSP Nasdaq Standard Deviation Decile 2	1000163	460	-
CRSP Nasdaq Standard Deviation Decile 3	1000164	460	-
CRSP Nasdaq Standard Deviation Decile 4	1000165	460	-
CRSP Nasdaq Standard Deviation Decile 5	1000166	460	-
CRSP Nasdaq Standard Deviation Decile 6	1000167	460	-
CRSP Nasdaq Standard Deviation Decile 7	1000168	460	-
CRSP Nasdaq Standard Deviation Decile 8	1000169	460	-
CRSP Nasdaq Standard Deviation Decile 9	1000170	460	-
CRSP Nasdaq Standard Deviation Decile 10	1000171	460	-
CRSP Nasdaq Standard Deviation Deciles	1000172	440	-

CRSP Indices *(Cont.)*

Index	indno	Daily setid	Monthly setid
CRSP NYSE Cap-Based Portfolio 3-5	1000311	-	420
CRSP NYSE Cap-Based Portfolio 6-8	1000312	-	420
CRSP NYSE Cap-Based Portfolio 9-10	1000313	-	420
CRSP NYSE Cap-Based Portfolio 1-5	1000314	-	420
CRSP NYSE Cap-Based Portfolio 6-10	1000315	-	420
CRSP NYSE Cap-Based Portfolio Market	1000316	-	420
CRSP NYSE Cap-Based Portfolios	1000317	-	400
CRSP NYSE/AMEX Cap-Based Portfolio 1	1000320	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 2	1000321	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 3	1000322	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 4	1000323	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 5	1000324	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 6	1000325	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 7	1000326	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 8	1000327	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 9	1000328	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 10	1000329	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 1-2	1000330	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 3-5	1000331	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 6-8	1000332	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 9-10	1000333	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 1-5	1000334	-	420
CRSP NYSE/AMEX Cap-Based Portfolio 6-10	1000335	-	420
CRSP NYSE/AMEX Cap-Based Portfolio Market	1000336	-	420
CRSP NYSE/AMEX Cap-Based Portfolios	1000337	-	400
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 1	1000340	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 2	1000341	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 3	1000342	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 4	1000343	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 5	1000344	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 6	1000345	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 7	1000346	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 8	1000347	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 9	1000348	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 10	1000349	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 1-2	1000350	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 3-5	1000351	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 6-8	1000352	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 9-10	1000353	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 1-5	1000354	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio 6-10	1000355	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolio Market	1000356	-	420
CRSP NYSE/AMEX/Nasdaq National Market Cap-Based Portfolios	1000357	-	400
CRSP Value-Weighted Index of the S&P 500 Universe	1000500	460	420
CRSP Equal-Weighted Index of the S&P 500 Universe	1000501	460	420
S&P 500 Composite	1000502	460	420
Nasdaq Composite	1000503	460	420
CRSP 30-Year Bond Returns	1000700	-	420
CRSP 20-Year Bond Returns	1000701	-	420
CRSP 10-Year Bond Returns	1000702	-	420
CRSP 7-Year Bond Returns	1000703	-	420
CRSP 5-Year Bond Returns	1000704	-	420
CRSP 2-Year Bond Returns	1000705	-	420
CRSP 1-Year Bond Returns	1000706	-	420
CRSP 90-Day Bill Returns	1000707	-	420
CRSP 30-Day Bill Returns	1000708	-	420
Consumer Price Index	1000709	-	420

Security Portfolios

Each index in monthly set 400 or daily set 440 is associated with a portfolio type available in stock security data. The `port` time series set in CRSPAccess97 stock databases has `subtype` set to the `indno` of the index group. The portfolio assignments in the `port` structure refer to series numbers in the associated index group. Portfolio data is only accessible in a CRSPAccess97 Stock database when CRSPAccess97 Indices File data is integrated. Portfolio access is described in the **CRSPAccess97 Stock Users Guide** and the **CRSPAccess97 Stock Programmers Guide**. The current slots are in the following table:

Security Portfolio Types

Index Group	indno	Daily Portfolio Type (setid = 440)	Monthly Portfolio Type (setid=400)
NYSE/AMEX/Nasdaq Capitalization Deciles	1000092	1	1
NYSE/AMEX Capitalization Deciles	1000052	2	2
Nasdaq Capitalization Deciles	1000072	3	3
NYSE Capitalization Deciles	1000012	4	4
AMEX Capitalization Deciles	1000032	5	5
NYSE/AMEX Beta Deciles	1000112	6	-
NYSE/AMEX Standard Deviation Deciles	1000132	7	-
Nasdaq Beta Deciles	1000152	8	-
Nasdaq Standard Deviation Deciles	1000172	9	-
Cap-Based NYSE/AMEX/Nasdaq National Market Portfolios	1000357	-	6
Cap-Based NYSE Portfolios	1000317	-	7
Cap-Based NYSE/AMEX Portfolios	1000337	-	8

CRSPAccess97 Data Objects

There are four basic data objects used in CRSPAccess97 databases; `CRSP_ROW`, `CRSP_ARRAY`, `CRSP_TIMESERIES` and `CRSP_CAL`. The objects and their common fields are described below. The components of indices objects are described in the C Index Structure and Data Items sections. See the CRSPAccess97 Stock Programmers Guide or the C Programming section in this guide for examples of usage of objects and components, and codes for predefined CRSP types of arrays or sub categories.

`CRSP_ROW` contains a single data structure. The components of the structure are applicable to the key at all times. The fields in the data structure are dependent on the specific data item.

`objtype` **Object Type Code** **`int`**

`objtype` is a code defining the type of object. `objtype` is always 5 for a `CRSP_ROW`.

`arrtype` **Array Type Code** **`int`**

`arrtype` is a code defining the type of data in the object. `arrtype` can define a basic data type or a CRSP-defined structure.

`subtype` **Data Subcategory Type Code** **`int`**

`subtype` is a code further defining categories of data that otherwise have the same structure, such as the difference between a return and index level data item.

`size_of_array_width` **Size of Associated Structure** **`int`**

`size_of_array_width` is the number of characters used in each structure element for this array type.

CRSP_ARRAY contains a list, or history of events, observations, or status changes. The array also maintains a count of the changes with num. The fields in each event are dependent on the specific data item. The fields usually contain one or more date fields indicating the effective time of the event, plus descriptive fields describing the event.

objtype **Object Type Code** **int**

objtype is a code defining the type of object. objtype is always 3 for a CRSP_ARRAY.

arrtype **Array Type Code** **int**

arrtype is a code defining the type of data in the object. arrtype can define a basic data type or a CRSP-defined structure.

subtype **Data Subcategory Type Code** **int**

subtype is a code further defining categories of data that otherwise have the same structure, such as the difference between a return and index level data item.

size_of_array_width **Size of Associated Structure** **int**

size_of_array_width is the number of characters used in each structure element for this array type.

maxarr **Maximum Number of Events** **int**

maxarr is the maximum number of event structures available in the CRSP_ARRAY.

num **Actual Number of Events** **int**

num is the count, the number of actual event structures available in the CRSP_ARRAY for the current entity.

CRSP_TIMESERIES contains a list of observations synchronized with a specific calendar of time periods. There is a beginning and ending index and a link to a calendar defining the time periods. The observation can contain a simple or multiple-component data structure, depending on the specific data item.

objtype **Object Type Code** **int**

objtype is a code defining the type of object. objtype is always 2 for a CRSP_TIMESERIES.

arrtype **Array Type Code** **int**

arrtype is a code defining the type of data in the object. arrtype can define a basic data type or a CRSP-defined structure.

subtype **Data Subcategory Type Code** **int**

subtype is a code further defining categories of data that otherwise have the same structure, such as the difference between a return and index level data item.

size_of_array_width **Object Type Code** **int**

objtype is a code defining the type of object. objtype is always 2 for a CRSP_TIMESERIES.

maxarr **Maximum Number of Time Periods** **int**

maxarr is the maximum number of time periods available in the CRSP_TIMESERIES, counting the 0^{th} period.

beg **First Time Period with Valid Data** **int**

beg is the first element in the data array with valid observations. If beg is 0 there is no data for the timeseries.

end **Last Time Period with Valid Data** **int**

end is the last element in the data array with valid observations. If end is 0 there is no data for this timeseries.

cal **Associated Calendar** **CRSP_CAL ***

cal is a pointer to the associated CRSP calendar structure. CRSP_CAL is described below.

CRSP_CAL contains a set of calendar time periods and a count of the number of periods. The calendars have an identifier and can be used by multiple time series. The calendar time periods are marked with an ending date and are synchronized with a time series such that the n^{th} time series observation is effective on the n^{th} calendar day.

objtype **Object Type Code** **int**

objtype is a code defining the type of object. objtype is always 1 for a CRSP_CAL.

calid **Unique Calendar Identifier** **int**

calid is a code assigned by CRSP to trading calendars. The following calendars are available in CRSPAccess97 databases.

calid	Description	Beginning Date
100	Daily Trading Calendar	19620702
101	Month-end Trading Calendar	19251231
300	Annual Trading Calendar	19251231
310	Quarterly Trading Calendar	19251231

ndays **Index of Last Period in Calendar** **int**

ndays is the number of periods in the calendar. Calendar and time series data can be available up to this index in the calendar array.

name **Calendar Name** **char[80]**

calname is a text description of the calendar.

caldt **Calendar Date** **int***

caldt contains the list of trading dates. Each date represents the last date in a calendar period, in YYYYMMDD (year, month, date) format. The caldt dates begin in element 1 of the array and continue to element ndays. The value in the nth position of a CRSP_TIMESERIES array is effective on the n^{th} caldt of its associated calendar.

CRSP Indices Data Objects

The following table summarizes the primary data items available in a CRSP index record. See the CRSP Data Objects Section for definitions of the object types and the CRSP Index Data Items section for the elements in the index structures.

The object name variable includes header object data plus the item array. Actual data is stored in the item array. Sets of objects represent an array of the given object type. The number of sets variable is the number of objects in this array. Each object in a set has its own set of object header variables. Sets of objects are used to represent groups of related objects, such as the different decile portfolios in a group index.

Item	Object Type	Object Name	Item Name	Number of Sets
Index Header	CRSP_ROW	indhdr_row	indhdr	
Set of Rebalancing Arrays	CRSP_ARRAY	rebal_arr	rebal	rebaltypes
Set of List Arrays	CRSP_ARRAY	list_arr	list	listtypes
Set of Used Count Time Series	CRSP_TIMESERIES	usdcnt_ts	usdcnt	indtypes
Set of Total Count Time Series	CRSP_TIMESERIES	totcnt_ts	totcnt	indtypes
Set of Used Value Time Series	CRSP_TIMESERIES	usdval_ts	usdval	indtypes
Set of Total Value Time Series	CRSP_TIMESERIES	totval_ts	totval	indtypes
Set of Total Return Time Series	CRSP_TIMESERIES	tret_ts	tret	indtypes
Set of Capital Appreciation Time Series	CRSP_TIMESERIES	aret_ts	aret	indtypes
Set of Income Return Time Series	CRSP_TIMESERIES	iret_ts	iret	indtypes
Set of Total Return Index Levels	CRSP_TIMESERIES	tind_ts	tind	indtypes
Set of Capital Appreciation Index Levels	CRSP_TIMESERIES	aind_ts	aind	indtypes
Set of Income Return Index Levels	CRSP_TIMESERIES	iind_ts	iind	indtypes

CRSP Index Data Item Descriptions

indhdr **Index header** **CRSP_IND_HEADER**

indhdr is a structure containing index header information. This includes identification and description fields describing the composition and methodology of the index.

indno **CRSP Permanent Index Number** **int**

indno is the unique permanent identifier assigned by CRSP to every supported index. All indnos are seven digit numbers. There is no inherent meaning in the numbers. The indices sets in a database are sorted by this field.

indco **CRSP Permanent Index Group Number** **int**

indco is the permanent number assigned by CRSP to all groups of indices. All indices based on the same market and statistical grouping are assigned the same indco.

primflag **Link to Primary Index** **int**

primflag is either the indno of a group index or a zero if there is no primary group index. A series index representing one portfolio of a group can use primflag to refer back to the primary index. The primary index will contain rebalancing information and data for all portfolios in that group. Only series indices have nonzero indno.

portnum **Portfolio Number if Subset Series** **int**

portnum is the portfolio number of an index series that is also a member of an index group. Primflag is the indno of the index group. This index is the portnumth series within the group index or zero if there is no primary group index.

indname **Index Name** **char[80]**

indname is the name of the portfolio index. The index names are listed in the CRSP indnos table.

typename **Index Group Name** **char[80]**

typename is the name of the index group to which the index belongs. All indices with the same indco will have the same typename.

method **Methodology Description Structure** **CRSP_IND_METHOD**

method contains fields describing the rules used to build the index. Fields contain information on primary and secondary methodologies and rules for weighting securities within the index.

methcode **Method Type Code** **int**

methcode is a code for the combination of primtype, subtype, wgttype, and wgtflag characteristics. Methodologies are described earlier in this section. Current codes are:

Code	Description
1	CRSP Cap-Based Portfolios
3	CRSP Risk-Based Indices
4	CRSP Value-Weighted Market Indices
5	CRSP Equal-Weighted Market Indices
6	CRSP Capitalization Decile Market Indices
7	S&P 500® Composite
8	CRSP Value-Weighted Index on the S&P 500® Universe
9	CRSP Equal-Weighted Index on the S&P 500® Universe
10	Nasdaq Composite
12	CRSP Fixed Term Bond Returns
13	CRSP Fixed Term Bill Returns
14	Provided by External Source

primtype **Primary Methodology Type** **int**

primtype is a code describing the type of index. The possible index types are

Code	Name	Description
0	Fractile Index	Breakpoints based on some rule and/or statistic are used to divide eligible issues into portfolios at different intervals. The breakpoint function is continuous so that all eligible issues are in exactly one portfolio during each period.
1	Selected Index	Universe is supplied from an outside source, with given issues or companies and the data ranges for each.
3	Market Index	Portfolio of all eligible issues is reevaluated each period based on constant universe restrictions
4	Other	Not Applicable

subtype **Secondary Methodology Group** **int**

subtype is a code for further detail of the primary index methodology type. The following codes are used

Code	Description
0	No further description
10	Portfolios based on market capitalization
12	Portfolios based on result statistic: beta or standard deviation
13	Issues in S&P 500® Index
14	Issues in the Nasdaq Composite Index
15	Treasury Issues of Selected Maturity Ranges

wgttype **Reweighting Type Flag** **int**

wgttype is a code indicating the method of weighting the issues in the portfolio index. The following codes are used.

Code	Description
0	Not available
1	Value-weighted, weights not supplied by CRSP
2	Value-weighted
3	Equal-weighted

wgtflag **Reweighting Timing Flag** **int**

wgtflag is a code indicating how frequently the weights are applied to the existing portfolio. The following codes are used.

Code	Description
0	Not available
11	Weights are applied each time period

flags **Exception Handling Flags** **CRSP_IND_FLAGS**

flags is a group of fields describing how the index supports exceptions in the data, such as new and delisted issues and missing data. Flags contains fields flagcode, addflag, delflag, delretflag, and missflag.

flagcode **Code of Basic Exception Types** **int**

flagcode is a code of the basic exception types. The following codes are currently used.

Code	Description
0	Unknown or not available
1	CRSP market index flags
2	Cap-Based index flags
3	CRSP market index trade-only prices flags

addflag **Handling of New Issues Flag** **int**

addflag is a code describing how new issues are used in an index. The following codes are used.

Code	Description
0	Unknown or not available
1	Adding securities are included the first period they meet existing portfolio restrictions

delflag **Handling of Ineligible Issues Flag** **int**

delflag is a code describing how issues that become ineligible are handled. The following codes are used.

Code	Description
0	Unknown or not available
1	Issues becoming ineligible are held until the next time period

delretflag **Return of Delisted Issues Flag** **int**

delretflag is a code describing whether delisting returns are applied to securities delisting from the exchange during a rebalancing period. The following codes are used.

Code	Description
0	Unknown or not applicable
1	Delisting return is applied to issues that delist during the period
2	Only returns while trading on the target exchange are used

missflag **Flag for Handling Missing Data** **int**

missflag describes the possible actions taken for securities with missing data during the range in an index portfolio. The following codes are used.

Code	Description
0	Unknown or not applicable
3	Issues without single period returns are excluded
5	Alternate prices are used if possible to generate single period returns
13	Quotes without trades are treated as missing prices

partuniv **Partition Subset Screening Structure** **CRSP_UNIV_PARAM**

induniv **Index Subset Screening Structure** **CRSP_UNIV_PARAM**

partuniv and induniv are structures of fields used to restrict a database using various screening variables. The screen fields are univcode, begdt, enddt, wantexch, wantnms, wantwi, wantinc, and shrcd. partuniv screens are used to restrict the securities used in defining partition breakpoints. induniv screens are used to restrict the securities used in the actual index.

univcode **Code of Universe Subset Types** **int**

univcode is a code defining a set of restrictions used in subset screening. The following codes are used.

Code	Description
0	Identifier restriction not applicable
10	NYSE common excluding foreign, ADR, REIT, Closed End Funds
11	NYSE/AMEX common excluding foreign, ADR, REIT, Closed End Funds
12	NYSE/AMEX/The Nasdaq National Market common excluding foreign, ADR's, REIT, Closed End Funds
20	NYSE common excluding ADRs
21	AMEX common excluding ADRs
22	NYSE/AMEX common excluding ADRs
23	Nasdaq common excluding ADRs
24	NYSE/AMEX/Nasdaq common excluding ADRs
30	NYSE common
31	AMEX common
32	NYSE/AMEX common
33	Nasdaq common
34	NYSE/AMEX/Nasdaq common
35	NYSE common excluding ADRs and foreigns
36	AMEX common excluding ADRs and foreigns
37	NYSE/AMEX common excluding ADRs and foreigns
38	Nasdaq common excluding ADRs and foreigns
39	NYSE/AMEX/Nasdaq common excluding ADRs and foreigns

begdt **First Trading Date Allowed in Restriction** **int**

begdt is the first date, in YYYYMMDD format, of data included in the universe. begdt is set to 0 if there is no date restriction.

enddt **Last Trading Date Allowed in Restriction** **int**

enddt is the first date, in YYYYMMDD format, of data included in the universe. enddt is set to 0 if there is no date restriction.

wantexch **Valid Exchange Codes in Universe** **int**

wantexch is code indicating the base exchanges in the universe subset. The following codes are used. The sum of two or more codes indicates all selected exchanges are valid.

Code	Description
0	No exchange restriction
1	New York Stock Exchange
2	American Stock Exchange
4	Nasdaq Stock Market

wantnms **Valid Nasdaq Market Groups in Universe** **int**

wantnms is a code indicating valid Nasdaq markets in the universe subset. The Nasdaq markets are subsets of The Nasdaq Stock Market[SM]. The following codes are used.

Code	Description
0	No National Market restriction or not applicable
1	Only issues listed on The Nasdaq National Market are included

wantwi **Valid When-Issued Securities in Universe** **int**

wantwi describes the types of when-issued trading allowed in a subset. The following codes are used.

Code	Description
0	No when-issued restrictions or not applicable
10	Initial when-issued trading is included when available. Ex-distributed trading is excluded. When-issued trading during reorganizations is included
110	Initial when-issued trading is excluded until issue attains regular-way status. ex-distributed trading is excluded. When-issued trading during reorganizations is included.

wantinc **Valid Incorporation of Securities in Universe** **int**

wantinc describes the incorporation of companies selected in a subset. The following codes are used.

Code	Description
0	Not applicable or no restriction by country of incorporation
1	Companies incorporated outside of the US are excluded

shrcd **Share Code Screen Structure** **CRSP_UNIV_SHRCD**

shrcd contains fields defining groups of CRSP security share codes included in the subset. See the following table for details of the two-digit share codes used by CRSP in the shrcd data item. The fields in the structure are sccode, fstdig, and secdig.

Share Type - Digit #1 - Security Traded

Code	Definition
1	Ordinary Common Shares
2	Certificates
3	ADRs (American Depository Receipts)
4	SBIs (Shares Of Beneficial Interest)
7	Units (Depository Units, Units Of Beneficial Interest, Units Of Limited Partnership Interest, Depository Receipts, Etc.)

Share Type - Digit #2 - Type of Security

Code	Definition
0	Securities Which Have Not Been Further Defined.
1	Securities Which Need Not Be Further Defined.
2	Companies Incorporated Outside The U.S.
3	Americus Trust Components (Primes And Scores)
4	Closed-End Funds
5	Closed-End Fund Companies Incorporated Outside The U.S.
8	REIT's (Real Estate Investment Trusts)

sccode **Share Code Groupings for Subsets** **int**

sccode is a code describing the generic share code groupings used in subsets. The following codes are used.

Code	Description
0	No share code restriction or not applicable
1	Common stocks excluding ADRs
2	Common stocks excluding ADRs and foreign incorporated companies
3	Common stocks excluding ADR's, foreign incorporated companies, REITS, and closed end funds
4	Common stocks

fstdig **Valid First Digit of Share Code** **int**

fstdig is a binary code describing the valid digits in the first digit of the share code in a subset universe. fstdig is the decimal representation of a 10-digit binary number. The **n**th digit of the binary number is 1 if an **n** in the first digit of the share code is valid in the subset, and a 0 otherwise.

secdig **Valid Second Digit of Share Code** **int**

secdig is a binary code describing the valid digits in the second digit of the share code in a subset universe. secdig is the decimal representation of a 10-digit binary number. The **n**th digit of the binary number is 1 if an **n** in the second digit of the share code is valid in the subset, and a 0 otherwise.

rules **Rules For Building Portfolios** **CRSP_IND_RULES**

rules is a group of fields describing any rules used to build portfolios. rules contains fields rulecode, buyfnct, sellfnct, statfnct, and groupflag.

rulecode **Code of Basic Rule Types** **int**

rulecode is a code of basic rule types. The following codes are currently used.

Code	Description
0	Unknown or not applicable
1	Group by previous period end issue capitalization
2	Group by previous period end company capitalization
3	Group by Scholes-Williams Beta over previous year
4	Group by standard deviation over previous year

buyfnct **Function Code for Buy Rules** **int**

buyfnct is a code defining a function used to determine whether to buy new issues in a portfolio at a rebalancing period. This is not yet available, and is set to 0.

sellfnct **Function Code for Sell Rules** **int**

sellfnct is a code defining a function used to determine whether to sell current issues in a portfolio at a rebalancing period. This is not yet available, and is set to 0.

statfnct **Function Code for Generating Statistics** **int**

statfnct is a code defining a function used to generate a statistic to be used in determining inclusion in a portfolio when it is rebalanced. The following codes are used.

Code	Description
0	Unknown or not applicable
1	Capitalization at end of previous period
2	Scholes-Williams Beta over previous year
3	Standard deviation over previous year

groupflag **Statistic Grouping Code** **int**

groupflag is a code describing the type of grouping done on issues before any statistics are applied. The following codes are used.

Code	Description
0	Unknown or not applicable
1	Each issues is grouped independently
2	Multiple issues of a company are combined

assign **Related Assignment Information** **CRSP_IND_ASSIGN**

assign is a group of fields defining the time periods and associated indexes used to form portfolios. It primarily defines the rebalancing periods when the portfolio is reformed based on new information. assign contains fields assigncode, asperm, asport, rebalcal, assigncal, and calccal.

assigncode **Code of Basic Assignment Types** **int**

assigncode is a code of basic assignment types. The following codes are currently used.

Code	Description
0	Unknown or not applicable
1	Annual rebalancing
2	Quarterly rebalancing
3	Monthly rebalancing

asperm **Indno of Associated Index** **int**

asperm is an indno of an associated index used to supply rebalancing breakpoint information used for assignments or buy/sell rules to this index. This is not yet available, and is set to 0.

asport **Portfolio Number in Associated Index** **int**

asport is the portfolio number in an associated index defined in asperm. The associated portfolio's breakpoint information for that portfolio is used for the current index. This is not yet available, and is set to 0.

rebalcal **Calid of Rebalancing Calendar** **int**

rebalcal identifies a calendar that determines the dates when the portfolios in the index are held. The new portfolio universe is held from one period in the rebalancing calendar until the next. The following calendars are used.

Code	Description
0	Unknown or not applicable
300	Annual calendar
310	Quarterly calendar

assigncal **Calid of Assignment Calendar** **int**

assigncal identifies a calendar that determines the dates when breakpoints and buy/sell rules are valid. The assignment calendar is used when using rebalancing information to assign issues to a portfolio. The period numbers of the calendars of the rebalcal, assigncal, and calccal are synchronized, although the actual date ranges for each period number may be different. assigncal uses the same calendars described in rebalcal.

calccal **Calid of Calculations Calendar** **int**

calccal identifies a calendar that determines the range of dates used to calculate statistics used to form portfolios. The period numbers of the calendars of the rebalcal, assigncal, and calccal are synchronized, although the actual date ranges for each period number may be different. calccal uses the same calendars described in rebalcal.

rebal **Rebalancing History** **CRSP_IND_REBAL**

Rebal is a set of event array structures containing historical rebalancing statistical information for rebalancing periods in an index. The variable rebaltypes contains the count of the rebalancing history arrays available for all indices in a set. There are 10 possible rebalancing arrays in group indices and 1 in series indices. Each array has its own count of periods, which is zero if not applicable to the particular index.

Each period contains the fields rbbegdt, rbenddt, usdcnt, maxcnt, totcnt, endcnt, minid, maxid, minstat, maxstat, medstat, and avgstat. Not all statistics are available for each index.

rbbegdt **Rebalancing Beginning Date** **int**

rbbegdt is the date, in YYYYMMDD format, of the first date in the rebalancing period.

rbenddt **Rebalancing Ending Date** **int**

rbenddt is the date, in YYYYMMDD format, of the last date in the rebalancing period.

usdcnt **Count used as of Rebalancing** **int**

usdcnt is the count of entities in the portfolio as of the beginning of the rebalancing period.

maxcnt **Maximum Count During Period** **int**

maxcnt is the largest count of issues in the portfolio at any point within the rebalancing period.

totcnt **Available Count as of Rebalancing** **int**

totcnt is the count of entities available in the universe eligible for the portfolio at the beginning of the rebalancing period

endcnt **Count at End of Rebalancing Period** **int**

endcnt is the count of entities belonging to the portfolio at the end of the rebalancing period.

minid **Identifier of Entity with the Minimum Statistic** **int**

minid is the identifier of the entity in the portfolio with the minimum statistic at the beginning of the rebalancing period.

maxid **Identifier of Entity with the Maximum Statistic** **int**

maxid is the identifier of the entity in the portfolio with the maximum statistic at the beginning of the rebalancing period.

minstat **Minimum Statistic in Period** **double**

minstat is the minimum statistic value in the portfolio at the beginning of the rebalancing period.

maxstat **Maximum Statistic in Period** **double**

maxstat is the maximum statistic value in the portfolio at the beginning of the rebalancing period.

medstat **Median Statistic in Period** **double**

medstat is the median statistic value in the portfolio at the beginning of the rebalancing period.

avgstat **Average Statistic in Period** **double**

avgstat is the average statistic value in the portfolio at the beginning of the rebalancing period.

list **List History** **CRSP_IND_LIST**

list is a set of array structures containing lists of issues comprising an index. The variable listtypes contains the count of the issue lists available for all indices in a set. There is 1 possible list arrays in group and series indices. Each array has its own count, which is set to zero if not applicable to the particular index.

Each list structure contains the fields permno, begdt, enddt, subind, and weight.

No list histories are available in the current index database.

permno **CRSP Permanent Number** **int**

permno is the CRSP permanent number, identifying the security included in the list.

begdt **First Date Included** **int**

begdt is the date, in YYYYMMDD format, of the first date the issue is included in the portfolio.

enddt **Last Date Included** **int**

enddt is the date, in YYYYMMDD format, of the last date the issue is included in the portfolio.

subind **Subcategory Code** **int**

subind is a flag indicating a subcategory of the primary index to which the security belongs. It is set to zero if no subcategory is applicable.

weight **Weight of Issue** **double**

weight is the defined weight of the issue within the index during the range indicated. It is set to zero if weighting is defined based on data and not part of the list definition.

usdcnt **Portfolio Used Count Arrays** **int**

usdcnt is a set of time series containing counts of issues used to create the index each period. In a market index, a security must have valid prices the current period and previous period to be included in the count.

The variable indtypes contains the count of the result time series available for all indices in a set. There are 17 possible usdcnt time series in group indices and 1 in series indices. Each time series has its own beginning and ending range, which is set to zero if usdcnt is not available for the particular index.

totcnt **Portfolio Eligible Count Arrays** **int**

totcnt is a set of time series containing counts of issues eligible for the index each period. In a market index, a security must have valid prices the current period to be included in the count.

The variable indtypes contains the count of the result time series available for all indices in a set. There are 17 possible totcnt time series in group indices and 1 in series indices. Each time series has its own beginning and ending range, which is set to zero if totcnt is not available for the particular index.

usdval **Portfolio Used Weight Arrays** **double**

usdval is a set of time series containing weights of issues used to create a value-weighted index each period. In a capitalization market index, usdval is the total market value, in $1000's, of all securities that are used in the index for a given period. A security must have available shares outstanding and valid prices the current period and previous period to be included in the weight.

The variable indtypes contains the count of the result time series available for all indices in a set. There are 17 possible usdval time series in group indices and 1 in series indices. Each time series has its own beginning and ending range, which is set to zero if usdval is not available for the particular index.

tret **Total Returns Arrays** **float**

tret is a set of time series containing the return including all distributions of the index or portfolio each period. See index methodology for the weighting and derivation of the index.

The variable indtypes contains the count of the result time series available for all indices in a set. There are 17 possible tret time series in group indices and 1 in series indices. Each time series has its own beginning and ending range, which is set to zero if tret is not available for the particular index.

aret **Capital Appreciation Arrays** **float**

aret is a set of time series containing the return excluding dividends of the index or portfolio each period. See index methodology for the weighting and derivation of the index.

The variable indtypes contains the count of the result time series available for all indices in a set. There are 17 possible aret time series in group indices and 1 in series indices. Each time series has its own beginning and ending range, which is set to zero if aret is not available for the particular index.

iret **Income Returns Arrays** **float**

iret is a set of time series containing the income return of the index or portfolio each period. Income return is defined as the difference between the total return and the capital appreciation. See index methodology for the weighting and derivation of the index.

The variable indtypes contains the count of the result time series available for all indices in a set. There are 17 possible iret time series in group indices and 1 in series indices. Each time series has its own beginning and ending range, which is set to zero if iret is not available for the particular index.

tind **Total Return Index Levels Arrays** **float**

tind is a set of time series containing the index level based on returns including all distributions of the index or portfolio each period. See index methodology for the base date and value.

The variable indtypes contains the count of the result time series available for all indices in a set. There are 17 possible tind time series in group indices and 1 in series indices. Each time series has its own beginning and ending range, which is set to zero if tind is not available for the particular index.

aind **Capital Appreciation Index Levels Arrays** **float**

aind is a set of time series containing the index level based on returns excluding dividends of the index or portfolio each period. See index methodology for the base date and value.

The variable indtypes contains the count of the result time series available for all indices in a set. There are 17 possible aind time series in group indices and 1 in series indices. Each time series has its own beginning and ending range, which is set to zero if aind is not available for the particular index.

iind **Income Return Index Levels Arrays** **float**

iind is a set of time series containing the index level based on income returns of the index or portfolio each period. Income returns are defined as the difference between total return and capital appreciation. See index methodology for the base date and value.

The variable indtypes contains the count of the result time series available for all indices in a set. There are 17 possible iind time series in group indices and 1 in series indices. Each time series has its own beginning and ending range, which is set to zero if iind is not available for the particular index.

3. ACCESSING THE DATA

The CRSP indices data can be accessed in three ways:

They are provided in fixed-length formatted character files that can be directly loaded into a number of software packages. There are four different file layouts of character files. This is the recommended method for users with no CRSPAccess97 stock databases.

Data can be overlaid on daily and/or monthly CRSPAccess97 stock databases. When integrated with stock data, the Indices File / Portfolio Assignments Product supplies additional portfolios assignment series to securities and additional market and portfolio group indices.

Index data integrated with CRSPAccess97 stock databases can be accessed with CRSPAccess97 utility programs *ts_print*, *dsxport*, and *msxport*, and portfolio data can be accessed with the *ts_print* and *stk_print* utility programs. Header files and search utilities can be used to list the available index series. See the **CRSPAccess97 Stock File Users Guide** Data Utilities section for details of the utility programs.

Index data integrated with CRSPAccess97 stock databases can be accessed with FORTRAN or C programs using CRSP programming libraries. The **CRSPAccess97 Stock File Programmers Guide** contains descriptions of stock sample programs and index and portfolio usage. This section describes additional FORTRAN indices access functions not available in a Stock-only database.

Index series data, but not portfolio data, can be used in a CRSPAccess97 database without an associated stock database. The *ts_print* utility and FORTRAN and C programs can be used to access the index data in these databases.

Indices are organized in two different ways. In C programming and with the *ts_print* utility, all indices series share a common structure and are accessed by `indno`, a unique permanent identifier assigned by CRSP. In FORTRAN programming and in the text files, indices are grouped into four common sets of indices based on methodology and accessed as a group.

The following sections list text file specifications, describe available reporting tools, and describe FORTRAN and C programming usage. See the data description section for detailed methodologies and data structure organization.

3.1 Indices Text File Specifications

CRSP provides four types of index text files: CRSP Stock Indices Files, Cap-Based Monthly History Files, CRSP Indices Files on the S&P 500®, and the CRSP US Treasury and Inflation Series (CTI) Files. All the files of one type have the same format. Only the data in the files are different since the files are based on different frequencies and exchanges. Dates are all have a YYMMDD format in the index text files.

CRSP Stock Indices File Specifications

This section shows the exact specifications of a formatted CRSP Stock Indices File. A CRSP Stock Indices File contains a record for each trading date in that data file, sorted by date. Records are all fixed-length of 466 characters long. Fields are delimited by spaces. Most floating point numbers are written using the FORTRAN E format specifier to ensure a constant number of significant digits. All files use the ASCII character set

The following table contains the format of each return and index level field in a CRSP Stock Indices File record. The character positions show where in the 466-character record each field is positioned. The FORTRAN format is the format that is written on the tape. The alternative format indicates the size and type of data found in the field. The data type shows whether the field contains real or integer data, and the associated name is the variable defined in the CRSP stock indices section of the variable definitions.

CRSP Stock Indices Record

Character Positions	FORTRAN Format	Data Type	Associated Name	Alternative FORTRAN Output Format
2-7	I6	Integer	CALDT	I6
9-21	E13.6	Real	VWRETD	F10.6
23-35	E13.6	Real	VWINDD	F10.6
37-49	E13.6	Real	VWRETX	F10.6
51-63	E13.6	Real	VWINDX	F10.6
65-77	E13.6	Real	EWRETD	F10.6
79-91	E13.6	Real	EWINDD	F10.6
93-105	E13.6	Real	EWRETX	F10.6
107-119	E13.6	Real	EWINDX	F10.6
121-133	E13.6	Real	SPRTRN or NCRTRN	F10.6
135-141	F7.2	Real	SPINDX or NCINDX	F7.2
143-155	E13.6	Real	DECRET(1)	F10.6
157-169	E13.6	Real	DECIND(1)	F10.6
-	E13.6	Real	DECRET(I)	F10.6
-	E13.6	Real	DECIND(I)	F10.6
395-407	E13.6	Real	DECRET(10)	F10.6
409-421	E13.6	Real	DECIND(10)	F10.6
423-437	E15.6	Real	TOTVAL	F14.2
439-443	I5	Integer	TOTCNT	I.5
445-459	E15.6	Real	USDVAL	F14.2
461-465	I5	Integer	USDCNT	I.5

There are 10 portfolios described by DECRET and DECIND. DECRET(I) is in character positions 143+28*(I-1) through 155+28*(I-1) and DECIND(I) is in character positions 157+28*(I-1) through 169+28*(I-1).

CRSP Cap-Based Portfolios File Specifications

This section shows the exact specifications of the Cap-Based Portfolios monthly results and rebalancing reports. The monthly results report contains a record for each decile or composite portfolio for each month, sorted by date, then portfolio. Records are all fixed-length 92 characters long. Fields are delimited by spaces. The rebalancing report contains a record for each decile portfolio each quarter, sorted by date, then portfolio. Fields are delimited by pipe characters (|).

Files use the ASCII character set.

The following table contains the format of each field in a monthly results record. The character positions show where in the 92-character record each field is positioned. The FORTRAN format is the format that is written on the tape. The data type shows whether the field contains character, real, or integer data, and the associated name is the variable defined in the Cap-Based Portfolios section of the variable definitions.

Monthly History Record

Character Positions	Fortran Format	Data Type	Associated Name
1-6	I6	Integer	CALDT
8-11	A4	Character	PRTNAM
13-16	I4	Integer	PRTCNT
18-28	F11.0	Real*8	PRTWGT
30-39	F10.6	Real	TOTRET
41-49	F9.3	Real	TOTIND
51-60	F10.6	Real	CAPRET
62-70	F9.3	Real	CAPIND
72-81	F10.6	Real	INCRET
83-91	F9.3	Real	INCIND

The following table contains the format of each field in a quarterly rebalancing record. The character positions show where in the 100-character record each field is positioned. The FORTRAN format is the format that is written on the tape. The data type shows whether the field contains character, real, or integer data, and the associated name is the variable defined in the Cap-Based Portfolios section of the variable definitions.

Quarterly Rebalancing Record

Character Positions	Fortran Format	Data Type	Associated Name
1-4	I4	Integer	YYMM
6-7	I2	Integer	PRTNO
9-13	I5	Integer	PRTCCT
15-23	F9.0	Real	MINCWT
25-56	A32	Character	MINCNM
58-66	F9.0	Real	MAXCWT
68-99	A32	Character	MAXCNM

File Specifications for the CRSP Indices on the S&P 500â

This section shows the exact specifications of the CRSP Indices files for the S&P 500® universe. The files contain a record for each date, sorted by date. Records are all fixed-length 130 characters long. Fields are delimited by spaces.

Data is provided with the ASCII character set.

The following table contains the format of each field in a file record for the CRSP Index on the S&P 500®. The character positions show where in the 130-character record each field is positioned. The FORTRAN format is the format that is written on the tape. The data type shows whether the field contains character, real, or integer data, and the associated name is the variable defined in the variable definitions section for the CRSP Indices on the S&P 500®.

File Record for CRSP Indices on the S&P 500â Universe

Character Positions	FORTRAN Format	Data Type	Associated Name	Alternative FORTRAN Output Format
2-7	I6	Integer	CALDT	I6
9-21	E13.6	Real	VWRETD	F10.6
23-35	E13.6	Real	VWRETX	F10 6
37-49	E13.6	Real	EWRETD	F10.6
51-63	E13.6	Real	EWRETX	F10.6
65-79	E13.6	Real	TOTVAL	F14.2
81-85	I5	Integer	TOTCNT	I5
87-101	E15.8	Real	USDVAL	F14.2
103-107	I5	Integer	USDCNT	I5
109-115	F7.2	Real	SPINDX	F7.2
117-129	E13.6	Real	SPRTRN	F10.6

CRSP US Treasury and Inflation Index Series

This section shows the exact specifications of a formatted CTI data file. A CTI file contains a record for each monthly trading date in that data file, sorted by date. Records are all fixed-length 288 characters long. Fields are delimited by spaces. Floating point numbers are written using the FORTRAN E format specifier to ensure a constant number of significant digits. Data is provided in the ASCII character set.

The following table contains the format of each calendar and index field in a CTI record. The character positions show where in the 288-character record each field is positioned. The FORTRAN format is the format that is written on the tape. The alternative format indicates the size and type of data found in the field. The data type shows whether the field contains real or integer data, and the associated name is the variable defined in the CTI section of the variable definitions.

CTI Record

Character Positions	Fortran Format	Data Type	Associated Name	Alternative Fortran Output Format
2-7	I6	Integer	CALDT	I6
9-21	E13.6	Real	B30RET	F10.6
23-35	E13.6	Real	B30IND	F10.6
37-49	E13.6	Real	B20RET	F10.6
51-63	E13.6	Real	B20IND	F10.6
65-77	E13.6	Real	B10RET	F10.6
79-91	E13.6	Real	B10IND	F10.6
93-105	E13.6	Real	B7RET	F10.6
107-119	E13.6	Real	B7IND	F10.6
121-133	E13.6	Real	B5RET	F10.6
135-147	E13.6	Real	B5IND	F10.6
149-161	E13.6	Real	B2RET	F10.6
163-175	E13.6	Real	B2IND	F10.6
177-189	E13.6	Real	B1RET	F10.6
191-203	E13.6	Real	B1IND	F10.6
205-217	E13.6	Real	T90RET	F10.6
219-231	E13.6	Real	T90IND	F10.6
233-245	E13.6	Real	T30RET	F10.6
247-259	E13.6	Real	T30IND	F10.6
261-273	E13.6	Real	CPIRET	F10.6
275-287	E13.6	Real	CPIIND	F10.1

3.2 Indices Data Utilities

Index data integrated with CRSPAccess97 stock databases can be accessed with CRSPAccess97 utility programs *ts_print*, *dsxport*, and *msxport*, and portfolio assignment data can be accessed with the *ts_print* and *stk_print* utility programs. Header files and search utilities can be used to list the available index series. Use the **CRSPAccess97 Stock File Users Guide** Data Utilities section for details of the utility programs if using an integrated stock and indices database.

If using a CRSPAccess97 Indices-only database, only a subset of the CRSPAccess97 Utilities are available. These are described below.

ts_print Time Series Report Writer

ts_print is a report-writer program for CRSPAccess97 data that generates report-style columnar text report files with raw and derived CRSP stock and indices data. *ts_print* is a command line executable program and enables users to control all of the specifications of reports without requiring previous programming knowledge. To maximize the potential of *ts_print*, the user should have a general familiarity of the data and the structure of their product.

ts_print is designed to facilitate data use by using CRSP time series data for raw and derived data types together with selected calendars. The input file is a text file that contains the specifications for the output data file or report. The output data file is in a delimited tabular text format and can either be used as is, or imported into spreadsheet or database programs for further processing.

Time series data is three-dimensional. Each data point refers an issue/index (`ENTITY`), a data item value (`ITEM`), and a date (`DATE`). *ts_print* allows the user to define these three components and to control the appearance of the output (`OPTIONS`), the fourth component. An input specification file controlled by the user determines the input and the output content and format.

ts_print is particularly useful for producing customized data sets for portfolio analysis and event studies. Time series data can be converted to different calendar frequencies, and header and event items can be mapped to a value at each period in a time series.

This section contains the following information:

- How to run *ts_print*
- Sample *ts_print* input files and output data
- Detail on creating the input file
- Tables containing the supported daily and monthly indices data items

ts_print is not designed for use with financial data other than data available through CRSP.

Running *ts_print*

ts_print runs in your command line terminal window.

At the command line, type `ts_print` and your input file name. The sample below would run the input file, `sample.txt`, created by the user. The file name can include a full path if it is not in the current directory. Note that the filenames must all be in lower case when using case-sensitive systems such as UNIX.

i.e. >`ts_print sample.txt` (enter)

A file name can be entered as the third parameter on the command line. If only two parameters are given, *ts_print* looks in the `OPTIONS` component of the input file for a file name. When the prompt returns, *ts_print* has generated the specified output file. If it returns an error message, revise your input file accordingly. The content of the error message should indicate what component you will need to revise to successfully run *ts_print*. The files default to the current directory that you are working in. If you want them in a specific directory, you will need to either work from that directory, specify file names with full path information, or move the files when you are done. For further data manipulation, these files can be edited with a text editor, or imported into a statistical package, a spreadsheet, or a database environment.

Sample *ts_print* input

- A basic example follows:

`sample.txt`

```
1.    ENTITY
      INDEX|INDNO 1000081
      END
2.    ITEM
      ITEMID ret
      END
3.    DATE
      CALNAME monthly|RANGE 19950101-19960101|CALFORMAT 4
      END
4.    OPTIONS
      X ITEM,YES|Y DATE,YES|Z ENTITY,YES,1
      OUTNAME finsamp.out|REPNAME Sample One
      END
```

In *ts_print*, `ENTITY`, `ITEM` and `DATE` identify what your report will contain, while `OPTIONS` determines how your report will appear.

Explanation of Input File: `sample.txt`

1. In the sample layout above the `ENTITY` is for one index, `INDNO` 1000081, the NYSE/AMEX/Nasdaq Value-Weighted Market Index.

2. Under `ITEM`, the report will contain total return data for the entity.

3. In this sample, `DATE` specifies that the report will contain a monthly value (`CALNAME`). Note the source of the `ITEM` selected above is a daily item. Thus, the monthly value for Daily `ITEM` is a monthly summary of the selected daily data item. In this case, ret is the compounded daily return during the month, reported between January ,1 1995 and December 31, 1995. Each date in the output file will be in a MM|DD|YYYY calendar format (`CALFORMAT` 4).

4. The `OPTIONS` selected contain x, y and z axes. `ITEM` options will be displayed on the X-axis, the `DATE` options on the Y axis and the entities will append themselves to the date or Y axis. (This is indicated by the

number 1 at the end of the Z options.) The YES in each of the axis groups indicates that the report will contain axes headers. OUTNAME, finsamp.out, is the name of the output file and REPNAME is the report title in the output file.

The following table is finsamp.out, the output from the input file, sample.txt, above. Keywords, REPNAME(report title), ENTITY, DATE and ITEMS were added manually, and the ENTITY (INDNO 1000081) and the ITEM (RET) were bolded.

sample index (REPNAME report title)
1000081

	Ret
19950131	0.042337
19950228	0.041606
19950331	0.037873
19950428	0.041396
19950531	0.034551
19950630	0.064973
19950731	0.068369
19950831	0.045558
19950929	0.037334
19951031	-0.027237
19951130	0.031716
19951229	0.024416

Each of the four components and their use are further detailed in the following sections.

Creating the Input File

It is necessary to create an input text file to run *ts_print*. The input file contains the data specifications and controls the appearance of the report or output data file. Each input file must contain each of the following four components: `ENTITY`, `ITEM`, `DATE` and `OPTIONS`, in order.

1. **`ENTITY`** –One or more index series `indnos` for precalculated CRSP supported indexes.
2. **`ITEM`** – One or more *ts_print* supported CRSP timeseries variables (data items). These can have possible translations from one periodicity (time period of aggregation) to another.
3. **`DATE`** - Dates can be a set of `YYYYMMDD` ranges or a set of relative days to be matched against event dates supplied with each entity.
4. **`OPTIONS`** – controls the appearance and name of the output file.

Input File Rules

- Names in uppercase `COURIER` are keywords and must be typed "as is"
- # represents a number to be supplied by the user
- # . . .represents a list of numbers, which can be a single number, a range of numbers separated by a dash, or any number of these two types separated by commas. For example, 1, 3-5, 7-9, 15 represents the numbers l ,3,4,5,7,8,9,15.
- Z represents an alphanumeric character to be supplied by the user
- names in lowercase `courier` are replaced by the user. For example, a filename is replaced by the name of a user's file.
- Anything in brackets is optional. If names in brackets are used all punctuation in the brackets are required. The brackets to not appear in the input file.
- Two or more keywords on a line must be separated with the `|(pipe)` character. Information specifying a keyword must be on the same line as the keyword. Additional keywords can also be placed on the second line; in this case the first line does not end in a pipe character.

ts_print is case sensitive. The user must follow the notational conventions, provided in this section, when creating the input file. The input file should be a text file with a `*.txt` extension. We recommend that the user not work with the input file in both windows-based and command line text editors, as they may appear to have different layouts.

The following section details each of the four components, `ENTITY`, `ITEM`, `DATE`, and `OPTIONS` and the keywords available for each.

Each component entry, numbered below, is comprised of three parts:

- A heading row which identifies the component,
- The center row(s) which detail(s) the desired function(s) of the component, and
- The `END` row, which closes the component input information

See the sample input for examples.

ENTITY Specifications

There is one way to describe indices-only entities:

1. An INDEX describes a pre-calculated index series supported by CRSP, specified by INDNO.

Each component entry is comprised of three parts:

- A heading row which identifies the component,
- The center row(s) which detail(s) the desired function(s) of the component, and
- The END row, which closes the component input information. A basic example follows:

The input file must begin with a heading or label, ENTITY. The center row(s) must contain the INDEX that the report is being run for, and the last row of the ENTITY section must read END.

Heading Row:
```
ENTITY
```

Center Row(s):
```
INDEX|INDNO #
```
(Additional options described below can be added to each line with the following syntax of entity options:)
```
|EVDATE #|USERHEAD text
```

End Row:
```
END
```
Following are portions of an example provided earlier in this section, which demonstrate the primary way to set up the ENTITY component of your input file. It pulls data for an indno.

```
e.g.  ENTITY
      INDEX|INDNO 1000081
      END
```

ENTITY Keywords and Usage

The capitalized words need to be used as is. The lowercase words and symbols are indicative of user-specified information.

INDEX - Indicator that the following ENTITY is an available CRSP index series.

> **INDNO** - (index number) permanent identifier of an index. Available indno series are listed in Section 2.6. The index search programs, *dindsearch* and *mindsearch*, can also be used to find available daily or monthly indices and their indnos.

USERHEAD - (user description for header) is used to generate output headers (short descriptions) for the ENTITY, in the output file. The default header, the ENTITY indno, is used unless the user specifies a USERHEAD string (up to 20 characters including spaces).

Note: All entities will have result data for the calendar date range specified in the DATE component of the input file for the data. If dates are selected outside this range in the D1, D2 or EVDATE Fields, the output data will contain missing value codes rather than values for the ITEMS selected.

Missing Return Codes

Parameter	ret(I)	Reason For Missing Return
rmissg	-66.0	more than 10 trading days between this day and the day of the latest preceding price
rmissr	-88.0	no return, array index not within range of begret and endret
rmissp	-99.0	missing return due to missing price

Data Item Specification

A complete list of currently supported data items are in the *ts_print* Data Items Tables at the end of the *ts_print* section. They are organized alphabetically by ITEMID, and contain the following information:

- group identifier (GROUPID),
- a further breakdown of available items into SUBNOs,
- the default header for each ITEM as it appears in the output file,
- the default format and data type assigned to each ITEM and
- the ENTITY types the data ITEMs are compatible with.

Each ITEMID selected will generate one output for each ENTITY on each DATE.

There are daily and monthly sets of items. Monthly item names are prefixed with an **m**. A monthly item requires an available CRSPAccess97 monthly database and a daily item requires an available CRSPAccess97 daily database. If both databases are available, both daily and monthly items can be included in the same report. A monthly or daily indices-only database can be used with *ts_print*, but only INDNO entities and index-compatible items can be used.

The ITEM specification is comprised of three parts

- A heading row which identifies the component
- The center row(s) which detail(s) the desired function(s) of the component, and
- The END row, which closes the component input information. A basic example follows:

Heading Row:

```
ITEM
```

Center Row:

```
ITEMID text  or  GROUPID text
```

One of ITEMID or GROUPID must be chosen. Additional options can be added to each line with the following syntax:

```
|SUBNO #|SDESC text |FORMAT text|DATALEN #
```

A line with end is used after the last item is specified.

End Row:

```
END
```

Data Item Keywords and Usage

Additional details for each of the ITEMs, can be found in the Daily and Monthly File Item Tables at the end of the chapter. Please refer to this table when creating your input file.

ITEMID - (item or variables selected) Code for the specific data item, ITEMID, requested. The data ITEMs are variables available in the database. Please refer to the variable descriptions (data definitions) for a more extensive definition.

SUBNO - ITEMIDs can contain secondary data item qualifiers. The secondary code refines the data request for specific data ITEMs. There is one secondary type of information in indices-only databases entered with the SUBNO field: subno flags to change the interpretation of a specific data item. Possible options and codes are identified in the SUBNO column of the Daily and Monthly File Item Tables at the end of this section

— **subno** - number is a code changing the interpretation of the data item as specified in the data item table. SUBNO 0 is the default for all ITEMs with SUBNOs. If the default is used, the SUBNO specification is not required

GROUPID - Code for a grouping of specific data items. If a group is selected all the items in the group are selected. At the end of the Data ITEMs section, there is a list, alphabetical by group, which provides the GROUPID with associated ITEMIDs and a brief description for each.

SDESC - (ITEM-header description) a short description for headers to override the default. The defaults are listed in the Daily and Monthly File Item Tables at the end of this section. If a group, this will be used for all items if specified. This is user-defined and not in the File Items Tables. This field may contain up to the maximum number character spaces allocated for in the format of each item (see the Daily & Monthly File ITEM Tables. This allows the user to define the names or headings for the items.

FORMAT – (report formatting) allows you to modify the formatting assigned to each of the data items. The user may not change the data type, but may change the length of spaces and justification (right or left). FORMAT must contain a % sign followed by the field width and data type. Data types include:

— d for integers,

— f for floating point numbers,

— s for character strings,

— lf for double precision numbers.

Floats and doubles can have a decimal point followed by a second number to indicate values after a decimal point. Any field can have a dash after the % sign to left justify the output in the field. Examples are %11.6f to print the output in an 11 character field with an explicit decimal point in the 5th position, or %-5s to print a left justified character field with a width of five. The FORMAT field allows you to change the character length and justification, but not the data type.

DATALEN - (columnar data length) the number of characters needed to store the output data to override or exceed the default. This should be at least as large as any field width specified in the format. This field should be modified when you wish to assign the field a header, which does not fit within the default FORMAT for the ITEM.

The following page contains the available daily and monthly data items, and group options, organized by group. See the data item tables (end of section) for a complete description of the data items for the input file.

Note that when using a GROUPID, the SUBNOs must be compatible. Use the Data Items table to verify compatibility.

The data length has been set to produce an output file, which is easily readable. If you are importing the data into another program for additional data manipulation, you may need to change the DATALEN (data length) field. This is particularly true with the character fields. The non-character fields may add spaces to the total allocated. If this occurs, use the FORMAT field to correct the total spaces for importing.

Following are two tables that contain the daily and monthly group items available with the CRSPAccess97 Indices File / Portfolio Assignments Product.

Daily File Items, Arranged by Group

groupid	itemid	Item Description
caldt	caldt	yyyymmdd trading date
cumrets	cumaret	cumulative capital appreciation from first day in range
	cumiret	cumulative income return from first day in range
	cumtret	cumulative total return from first day in range
inddata	cap	effective (end of previous period) capitalization of issue or index
indres	aind	capital appreciation index level
	iind	income return index level
	tind	total return index level
indstat	cnt	count in index valid current and previous day
rdata	reti	total income return –compounded daily return of period
	retx	capital appreciation return – compounded daily return of period

Monthly File Items, Arranged by Group

groupid	itemid	Item Description
mcaldt	mcaldt	yyyymmdd trading date
mcumrets	mcumaret	cumulative capital appreciation from first day in range
	mcumiret	cumulative income return from first day in range
	mcumtret	cumulative total return from first day in range
minddata	mcap	effective (end of previous period) capitalization of issue or index
mindres	maind	capital appreciation index
	miind	income return index level
	mtind	total return index level
mindstat	mcnt	count in index valid current and previous day
rdata	mreti	total income return –compounded monthly return of period
	mretx	capital appreciation return – compounded monthly return of period

DATE Specification

The DATE input specifies the set of dates or ranges that will appear in the output file for the ITEMs selected. The calendar may be one of four calendars in the database: daily, monthly, quarterly, or annual. The ranges can be either the same for all input entities, or be based on an event date for each permno.

Each component is comprised of three parts

- A heading row which identifies the component
- The center row(s) which detail(s) the desired function(s) of the component, and
- The END row, which closes the component input information. A basic example follows:

Heading Row:
```
DATE
```

Center Row:
```
CALNAME text|RANGE or RELATIVE dates|CALFORMAT #
```

End Row:
```
END
```

The calendar name and either an absolute or relative range must be chosen. The date specification keywords are described below.

ts_print contains a 'smart calendar', which allows the user to select the various calendar options with the various data items. For example a daily CALNAME with a monthly data file would return a daily range of data, using the month end value to fill in values for the days there is no additional data in the file. Likewise, a daily calendar with a monthly, quarterly or annual reporting frequency will utilize monthly, quarterly, and annual figures for the selected ITEMS.

Following are 2 examples. The first example will pull the daily or monthly items in the date range between January 1, 1990 and December 31, 1995. The calendar indicates the frequency of the data items selected for the report. The second example will report on a daily basis a total of 4 days, from 2 days before the event date, the event date (EVDATE), and 1 day after the event date. The event date is specified in the ENTITY specification section of your input file.

```
e.g.  DATE
      CALNAME quarterly|RANGE 19900101-19951231
      END

e.g.  DATE
      CALNAME daily|RELATIVE -2,1
      END
```

DATE Specification– Keywords

CALNAME - (calendar name) name of an existing calendar to report on. *ts_print* supports reporting for Daily, Monthly, Quarterly and Annual Calendars. Either daily or monthly file data items can be used with any of the supported calendars. Input data frequency is determined by the data item specified in the ITEM section. The supported calendars must be chosen from the following table and set the frequency of reporting in the output file.

The following table contains a list of the calendars currently supported.

CALNAME	Calendar Description
daily	CRSP daily stock calendar
weekly	CRSP weekly stock Calendar
monthly	CRSP monthly stock calendar
quarterly	CRSP quarterly stock calendar
annual	CRSP annual stock calendar

RANGE daterange - sets the fixed time period, with beginning and end dates, of the selected calendar that *ts_print* reports data. Ranges can be expressed as YYYY-YYYY, YYYYMM-YYYYMM, YYYYMMDD-YYYYMMDD, or YYMMDD-YYMMDD. If only a month or year is specified, all dates in the calendar belonging to that month or year will be included. If the chosen dates are not in the selected calendar, the beginning range uses the next following date in the calendar and the ending range uses the last previous date in the calendar. Output will be produced for all entities for all items for each period in the range. If the entity does not have data during the range or is restricted by date range in the ENTITY description section, missing values will be filled in.

CALFORMAT - (calendar format) a numeric code for the formatting of the dates in a range when date headers are chosen in the output options:

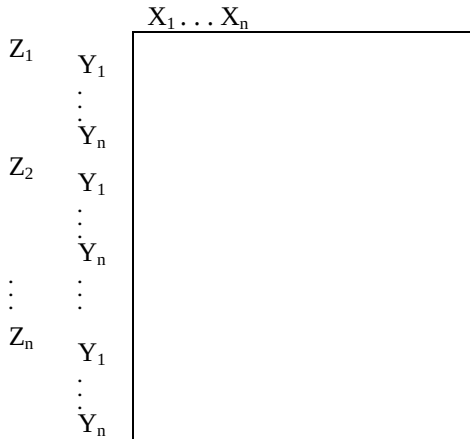
- 1 (default) =YYYYMMDD
- 2 =YYMMDD
- 3 =MM|DD|YY
- 4 =MM|DD|YYYY
- 5 =DD-MM-YYYY

OPTIONS and Output Specification

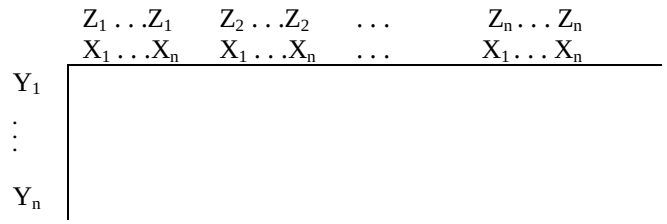
Each data point represents the data ITEM value for one ENTITY on a given DATE. These three points are plotted in a table to produce the output file. The OPTIONS component specifies the appearance of the output file.

Each of the three data dimensions, ITEM, ENTITY and DATE, are assigned by the user to the X, Y, or Z axis. The Z axis can be represented in two dimensions in one of the following three ways:

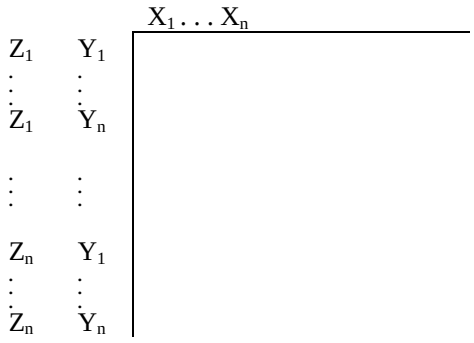
X and Y table are repeated for each Z.



On X axis, All X items for Z item 1, All X items for Z item 2, ..., All X items for Z item N.



On Y axis, All Y items for Z item 1, All Y items for Z item 2, ..., All Y items for Z item N. This is similar to the first option, but Z headers are available on each line.



These three dimension assignments are required in the OPTIONS component.

Each component is comprised of three parts

- A heading row which identifies the component
- The center row(s) which detail(s) the desired function(s) of the component, and
- The END row, which closes the component input information. A basic example follows:

Heading Row:

```
OPTIONS
```

Center Row:

```
X type[,headers] |Y type[,headers] |Z type[,headers], zflag#
OUTNAME filename|REPNAME text|FIELDDELIM text|BUFSIZE #
CHARDELIM text|ROWDELIM #,#|DEFAULT #
```

End Row:

```
END
```

The following example contains the required X, Y, and Z axes specifications. The ENTITY data will lie along the X axis, the DATE on the Y axis and Z will also lie along the y axis. The headers will all be displayed in the report, since they have been omitted in this example, but are the default for each axis specification. *ts_print* will generate an output file (OUTNAME) named ts_samp3.dat into the directory you are working in. The report will have a heading called Sample 6.

```
e.g. OPTIONS
     X ENTITY|Y DATE|Z ITEM,3|OUTNAME ts_samp3.dat|REPNAME Sample6
     END
```

Report OPTIONS Keywords and Usage

X axis, Y axis, and Z axis assignments are mandatory, and must allocate ENTITY, ITEM and DATE to the graphical axes.

— Header references must be included in the output file; YES, to show or NO, to be hidden. Header specification is included with each axis specification. The default is YES. The default header for an ENTITY is the permno for a security and indno for an index. The default header for a data ITEM is the item header in the Data Item Table at the end of the *ts_print* section. The default header for DATE is YYYYMMDD for calendar ranges and relative period numbers for relative dates.

— The Z-flag field is required. It is a number, 1, 2, or 3, as described in the diagrams on the previous page, which determines how the z axis data is printed in the two dimensional output.

OUTNAME - (output file name) is the name of the file where the output will be stored. Without this, the data will dump to the screen.

REPNAME - (report name) is a text description that will be placed at the top of the report (default is none)

NOFILL - rows outside an issue's date range or the user's date specification will not print to the output file NOFILL is only applicable if ITEM is chosen for the X axis, DATE for the Y axis and ENTITY for the Z axis, and zflag# is 1 or 3.

FIELDDELIM - (field delimiter-columnar) is a specified character string that will be placed as a delimiter between fields in output file rows (default is none)

BUFSIZE - (number of bytes) is the size of memory that will be allocated by the program. In a large study, the program will save intermediate data in a temporary file, and this can degrade performance severely. If memory is

available on your system, you can use the BUFSIZE option to increase the size of the internal buffer. Do not use BUFSIZE for small reports. The program will report the necessary buffer size if a specific report requires more than the default limit. Switching axes can also be used to improve performance for large datasets.

CHARDELIM - (character delimiter-column) is a character string placed around all character string fields in output file rows (default is none). *ts_print* left justifies the character fields. This means fields that have a character format 's' or ITEMIDs – comnam/mcomnam, ticker/mticker, cusip/mcusip, or ncusip/mncusip, are left justified. If you expect to read the *ts_print* results as delimited data, it is recommended that you add this to your OPTIONS component when creating your input file.

ROWDELIM - (field delimiter-row) is the number of rows between output lines. The first is the number between tables when the Z dimension is a separate table (default is 1). The second is the number between data rows (default is 0). Use of this will look like this (#,#)

DEFAULT - (default settings) is a summary of output formats. It must be set to 1 if used. It can be used instead of the other output options. Each Z is a separate table. Field delimiter is a space. Within each table, rows are single-spaced. There are two rows between tables. All headers are printed. If default is chosen, the other output format options are ignored.

Note: The row detailing the functionality of a single option must wrap. Different keywords can be on separate lines, but the last keyword on a line cannot end with a pipe character, and the beginning of a line must be a keyword.

ts_print Daily Index Items

itemid	groupid	subno	Item Header	Item Description	Format
aind	indres		Aind	Capital appreciation index level	%11.5f
caldt	caldt		Caldt	yyyymmdd trading date	%9d
cap	inddata	0 - usdval 1 - totval	Cap Cape	effective (end of previous period) capitalization of index – usdval ending capitalization of index - totval	%15.5lf
cnt	indstat	0 - usdcnt 1 - totcnt	Cnt Cntprev	count in index valid current and previous day -usdcnt count in index valid current day - totcnt	%6d
cumaret	cumrets		Cumaret	cumulative capital appreciation from first day in range	%11.6f
cumiret	cumrets		Cumiret	cumulative income return from first day in range	%11.6f
cumtret	cumrets		Cumtret	cumulative total return from first day in range	%11.6f
iind	indres		Iind	income return index level	%11.6f
permco	headid		PERMCO	CRSP INDCO	%8d
permno	headid		PERMNO	CRSP INDNO	%8d
ret	ddata		Ret	total return compounded over period	%11.6f
reti	rdata		Reti	income return –compounded over period	%11.6f
retx	rdata		Retx	capital appreciation compounded over period	%11.6f
tind	indres		Tind	total return index level	%11.5f

ts_print Monthly Index Items

itemid	groupid	subno	Item Header	Item Description	Format
maind	mindres		Aind	Capital appreciation index level	%11.5f
mcaldt	mcaldt		Caldt	yyyymmdd trading date	%9d
mcap	minddata	0 - usdval 1 - totval	Cap Cape	effective (end of previous period) capitalization of index – usdval ending capitalization of index – totval	%15.5lf
mcnt	mindstat	0 - usdcnt 1 - totcnt	Cnt Cntprev	count in index valid current and previous day –usdcnt count in index valid current day – totcnt	%6d
mcumaret	mcumrets		Cumaret	cumulative capital appreciation from first day in range without dividends	%11.6f
mcumiret	mcumrets		Cumiret	cumulative income return from first day in range	%11.6f
mcumtret	mcumrets		Cumtret	cumulative total return from first day in range	%11.6f
miind	mindres		Iind	income return index level	%11.6f
mpermco	mheadid		PERMCO	CRSP INDCO	%8d
mpermno	mheadid		PERMNO	CRSP INDNO	%8d
mret	mddata		Ret	total return compounded daily return of period	%11.6f
mreti	mrdata		Reti	total income return –compounded daily return of period	%11.6f
mretx	mrdata		Retx	capital appreciation return - compounded daily return of period	%11.6f
mtind	indres		Tind	total return index level	%11.5f

See the **CRSPAccess97 Stock File Users Guide** for all items available when using the indices with the stock products.

Header Search Utilities

CRSP provides a header file for a CRSPAccess97 Indices-only database. The header file is useful as an online list of index identifiers, set identifiers, and descriptions of the indices available in the database. The file is a fixed format text file and be accessed with system utilities or other tools. CRSP provides search utilities available to search the header file.

Every indices-only database contains one file:

`headind.dat` - index description, `setid`, and `indno` of all index series and groups in the database.

Header Search Utilities

There are two index header search utilities,

1. *DINDSEARCH* for index header list in daily database

2. *MINDSEARCH* for index header list in monthly database

The utility uses a search string as input and finds program will ask for a search string. It will print the full list of header rows satisfying the simple search.

Operating System Specific Search Instructions

Windows NT

The command and the string, enclosed in double quotes, are entered at the command line at a command prompt window. For example,

```
>dindsearch "S&P"
>echo off

                      Daily CRSP Indices
 INDNO SETID    Index Name

---------- N:\DATA\DX9612\HEADIND.DAT
1000500 460 CRSP Value-Weighted Index of the S&P 500 Universe
1000501 460 CRSP Equal-Weighted Index of the S&P 500 Universe
1000502 460 S&P 500 Composite
```

Unix or OpenVMS

Type the name of the search function. You will be prompted for the search string. No quotes are needed and case is ignored. For example,

```
$dindsearch

Enter search string: S&P

                           Daily Indices

PERMNO SETID   Index Name

1000500 460 CRSP Value-Weighted Index of the S&P 500 Universe
1000501 460 CRSP Equal-Weighted Index of the S&P 500 Universe
1000502 460 S&P 500 Composite

Try another string [y] ? n
```

3.3 FORTRAN Programming

There are six FORTRAN access subroutines and four FORTRAN `INCLUDE` files with common block declarations that can be used to access CRSPAccess97 Indices databases. An Indices database overlaid on a CRSPAccess97 Stock Database or an Indices-only CRSPAccess97 database has access to these functions. No FORTRAN programming support is provided for CRSPAccess97 text files.

There are six access subroutines that load indices data, `UDCAL`, `UMCAL`, `UDDEC`, `UMDEC`, `UMCAP`, and `UMCTI`. `UDCAL` and `UMCAL` load the CRSP calendar/indices for an exchange or combination of exchanges into the /`CAL`/ common block. `UDDEC` and `UMDEC` load the CRSP stock indices' index levels and decile returns into the /`DECILE`/ common block. `UMCAP` loads the CRSP Cap-Based monthly results into the /`CAPBAS`/ common block. `UMCTI` loads the monthly CTI data into the /`CTIS`/ common block. The `USSAMP7` FORTRAN program described in the **CRSPAccess97 Stock File Programmers Guide** contains examples of their usage.

Calendar dates are YYMMDD date format by default. The variable `DATEFFLAG` can be changed from the default value of zero to one to load dates in YYYYMMDD format.

FORTRAN Include Files

Include files are used in FORTRAN as a convenient way to replace blocks of code with a single statement. There are four include files available in the CRSP_INCLUDE directory with variables corresponding to definitions in Sections 2.2 through 2.5 of this guide. The statements can be included with the INCLUDE statement in FORTRAN. For example, the line

```
INCLUDE 'uscti.txt'
```

at the top of a FORTRAN program is used to add the CTI variable definitions.

The four header files and the relevant common blocks are described below. Each of the access subroutines load one of the common blocks.

us.txt

The *us.txt* include file declares and dimensions all the arrays loaded by the UDCAL and UMCAL subroutines. This includes all data items described in Section 2.2 and Section 2.4, except index levels and decile variables. The /CAL/ common block is declared in *us.txt*, which is also used for stock calendar/indices data.

```
1          ************************************************************************
2          * Common blocks, dimension statements and equivalence statements       *
3          * for CRSP Stock Files user access programs.                            *
4          ************************************************************************
5
6                include 'usparm.txt'       !usparm has all the parameters
7
8          * Calendar block                 !MAXDAT = max # trading days
9
10               INTEGER  CALDT(MAXDAT)     !Trading date YYMMDD
11               REAL     VWRETD(MAXDAT)    !value-weighted market return w dividends
12               REAL     VWRETX(MAXDAT)    !value-weighted market return w/o dividends
13               REAL     EWRETD(MAXDAT)    !equal-weighted market return w dividends
14               REAL     EWRETX(MAXDAT)    !equal-weighted market return w/o dividends
15               REAL     TOTVAL(MAXDAT)    !total value of stocks w non-missing price
16               INTEGER  TOTCNT(MAXDAT)    !total count of stocks w non-missing price
17               REAL     USDVAL(MAXDAT)    !total value of stocks used for value-weighting
18               INTEGER  USDCNT(MAXDAT)    !total count of stocks used for value-weighting
19               REAL     SPINDX(MAXDAT)    !S&P 500 Composite index level
20               REAL     SPRTRN(MAXDAT)    !S&P 500 Composite index return
21               REAL     NCINDX(MAXDAT)    !NASDAQ index level
22               REAL     NCRTRN(MAXDAT)    !NASDAQ index return
23               INTEGER  NDAYS             !number of dates in the current calendar
24
25               COMMON /CAL/ CALDT,VWRETD,VWRETX,EWRETD,EWRETX,
26              *                   TOTVAL,TOTCNT,USDVAL,USDCNT,
27              *                   SPINDX,SPRTRN,NDAYS
28
29               EQUIVALENCE(SPINDX(1),NCINDX(1))
30               EQUIVALENCE(SPRTRN(1),NCRTRN(1))
```

usdec.txt

The *usdec.txt* include file declares and dimensions all the arrays loaded by the UDDEC and UMDEC subroutines. This includes decile and index level data items described in Section 2.2.

```
      ********************************************************************
      *      declarations for decile portfolio indices and returns.
      ********************************************************************

            INTEGER MAXPORTS, IUNDEC
            PARAMETER (MAXPORTS = 10)         ! number of portfolios
            PARAMETER (IUNDEC = 105)          !Logical unit number for decile file

            REAL VWINDX(MAXDAT)              ! index value for vwretx
                                             !    index = 100.0 on 721229
            REAL VWINDD(MAXDAT)              ! index value for vwretd
            REAL EWINDX(MAXDAT)              ! index value for ewretx
            REAL EWINDD(MAXDAT)              ! index value for exretd

            REAL DECRET(MAXPORTS,MAXDAT)     ! decile portfolio return
            REAL DECIND(MAXPORTS,MAXDAT)     ! decile portfolio index
                                             !    index = 100.0 on 721229

            COMMON /DECILE/ VWINDX, VWINDD, EWINDX, EWINDD, DECRET, DECIND

      C
```

uscap.txt

The *uscap.txt* include file declares and dimensions all the arrays loaded by the UMCAP subroutine. This includes all data items described in Section 2.3.

```
      ********************************************************************
      *      declarations for cap-based portfolio indices and returns.
      ********************************************************************

            INTEGER MAXCPT, MAXMONC, IUNCAP
            PARAMETER (MAXCPT = 17)         ! number of portfolios
            PARAMETER (MAXMONC = 900)       ! number of months
            PARAMETER (IUNCAP = 107)        !Logical unit number for cap file

            CHARACTER*4 PRTNAM(MAXCPT)      !  portfolio name
            INTEGER PRTCNT(MAXCPT,MAXMONC)  !  portfolio count
            REAL*8  PRTWGT(MAXCPT,MAXMONC)  !  portfolio weight
            REAL TOTRET(MAXCPT,MAXMONC)     !  portfolio total return
            REAL TOTIND(MAXCPT,MAXMONC)     !  portfolio total return level
            REAL CAPRET(MAXCPT,MAXMONC)     !  portfolio cap appreciation
            REAL CAPIND(MAXCPT,MAXMONC)     !  portfolio cap appreciation level
            REAL INCRET(MAXCPT,MAXMONC)     !  portfolio income return
            REAL INCIND(MAXCPT,MAXMONC)     !  portfolio income return level
                                            !    index = 1.0 on 251231

            COMMON /CAPBAS/ PRTNAM, PRTCNT, TOTRET, TOTIND,
           .           CAPRET, CAPIND, INCRET, INCIND
            COMMON /CAPWGT/ PRTWGT
```

uscti.txt

The *uscti.txt* include file declares and dimensions all the arrays loaded by the UMCTI subroutine. The arrays are described in section 2.5.

```
1          ************************************************************************
2          *    USCTI.TXT                                                         *
3          *         Parameter and arrays used to load monthly CTI indices into   *
4          *         US stock file programs.                                      *
5          ************************************************************************
6
7                INTEGER IUNCTI
8                PARAMETER (IUNCTI = 106)         !Logical unit number for SBBI file
9
10               INTEGER MAXMON
11               PARAMETER (MAXMON=900)
12
13               REAL
14              + B30RET(MAXMON),          !U.S Treasury Bonds 30 Year Bond Return
15              + B30IND(MAXMON),          !   index value (721229 = 100.0)
16              + B20RET(MAXMON),          !U.S Treasury Bonds 20 Year Bond Return
17              + B20IND(MAXMON),          !   index value
18              + B10RET(MAXMON),          !U.S Treasury Bonds 10 Year Bond Return
19              + B10IND(MAXMON),          !   index value
20              +  B7RET(MAXMON),          !U.S Treasury Bonds 7 Year Bond Return
21              +  B7IND(MAXMON),          !   index value
22              +  B5RET(MAXMON),          !U.S Treasury Bonds 5 Year Bond Return
23              +  B5IND(MAXMON),          !   index value
24              +  B2RET(MAXMON),          !U.S Treasury Bonds 2 Year Bond Return
25              +  B2IND(MAXMON),          !   index value
26              +  B1RET(MAXMON),          !U.S Treasury Bonds 1 Year Bond Return
27              +  B1IND(MAXMON),          !   index value
28              + T90RET(MAXMON),          !U.S. Treasury Bills 90-Day Bill Return
29              + T90IND(MAXMON),          !   index value
30              + T30RET(MAXMON),          !U.S. Treasury Bills 30-Day Bill Return
31              + T30IND(MAXMON),          !   index value
32              + CPIRET(MAXMON),          !Rate of change, Consumer Price Index
33              + CPIIND(MAXMON)           !   index value
34
35               COMMON /CTIS/ B30RET, B30IND, B20RET, B20IND, B10RET, B10IND,
36              +               B7RET,  B7IND,  B5RET,  B5IND,  B2RET,  B2IND,
37              +               B1RET,  B1IND, T90RET, T90IND, T30RET, T30IND,
38              +             CPIRET, CPIIND
```

The statement INCLUDE 'usdec.txt' must be added at the top of any program that calls UDDEC or UMDEC, the statement INCLUDE 'uscap.txt' must be added at the top of any program that calls UMCAP, and the statement INCLUDE 'uscti.txt' must be added to programs that call UMCTI.

For example, for access to both the CRSP stock indices and monthly CTI:

```
INCLUDE 'us.txt'
INCLUDE 'usdec.txt'
INCLUDE 'uscti.txt'
```

FORTRAN Indices Access Subroutines

There are six indices access subroutines available — UDCAL,UMCAL,UDDEC,UMDEC,UMCAP, and UMCTI. The indices access subroutines use parameters to specify a certain set of data if more than one set is available. Each subroutine loads one of the common blocks described in the FORTRAN INCLUDE File's descriptions, and overwrites and data previously loaded in the corresponding common block. The subroutines are used in a program with the syntax CALL subname(parameterlist)

UDCAL (Keyword)
UMCAL (Keyword)

UDCAL loads daily data into the /CAL/ common block, and UMCAL loads monthly data in to the /CAL/ common block. All three of the individual exchanges and the two combinations of exchanges are supported. The keywords or their integer equivalents can be used as parameters.

Keyword	Value	Data Loaded
NYSEAMEX	1	NYSE and AMEX Combined
NASDAQ	2	Nasdaq only
NYSEAMEX+NASDAQ	3	NYSE/AMEX and Nasdaq Combined
NYSE	4	NYSE only
AMEX	5	AMEX only
SP500	6	Indexes on the S&P 500 Universe

A call to UDCAL and UMCAL will overwrite what is currently in the /CAL/ common block, even if it was loaded by a call to UDOPEN or UMOPEN.

UDDEC (Exchange,Type)

UDDEC loads all daily index levels and the ten market segment returns into the /DECILE/ common block. The keywords or their integer equivalents can be used as parameters. The following eight combinations of the arguments are supported for daily data.

EXCHANGE		TYPE		
Keyword	Value	Keyword	Value	Data Loaded
NYSEAMEX	1	CAP	1	NYSE/AMEX Capitalization Deciles
NYSEAMEX	1	BETA	3	NYSE/AMEX Beta Deciles
NYSEAMEX	1	SDEV	2	NYSE/AMEX Std. Deviation Deciles
NASDAQ	2	CAP	1	Nasdaq Capitalization Deciles
NASDAQ	2	BETA	3	Nasdaq Beta Deciles
NASDAQ	2	SDEV	2	Nasdaq Std. Deviation Deciles
NYSEAMEX + NASDAQ	3	CAP	1	Combined Capitalization Deciles
NYSE	4	CAP	1	NYSE-only Capitalization Deciles
AMEX	5	CAP	1	AMEX-only Capitalization Deciles

A call to UDDEC reinitializes and loads large arrays, and programs should be organized to minimize the number of calls made.

UMDEC (Exchange,Type)

UMDEC loads all monthly index levels and the ten market segment returns into the /DECILE/ common block. The keywords or their integer equivalents can be used as parameters. The following four combinations of the arguments are supported for monthly data.

Exchange		Type		
Keyword	**Value**	**Keyword**	**Value**	**Data Loaded**
NYSEAMEX	1	CAP	1	NYSE/AMEX Capitalization Deciles
NASDAQ	2	CAP	1	Nasdaq Capitalization Deciles
NYSEAMEX + NASDAQ	3	CAP	1	Combined Capitalization Deciles
NYSE	4	CAP	1	NYSE-only Capitalization Deciles
AMEX	5	CAP	1	AMEX-only Capitalization Deciles

A call to UMDEC reinitializes and loads large arrays, and programs should be organized to minimize the number of calls made.

UMCAP (Keyword)

UMCAP loads all monthly CRSP Cap-Based counts, returns and associated index levels into the /CAPBAS/ common block. These keywords or their integer equivalents can be used as parameters. No daily Cap-Based data are available.

Keyword	Value	Data Loaded
NYSEAMEX	1	NYSE and AMEX Combined
NYSEAMEX+NASDAQ	3	NYSE/AMEX and Nasdaq National Market Combined
NYSE	4	NYSE only

A call to UMCAP reinitializes and loads large arrays, and programs should be organized to minimize the number of calls made.

UMCTI

UMCTI loads all monthly CTI returns and associated index levels into the /CTIS/ common block. There are no arguments to this subroutine. No daily CTI data are available.

3.4 CRSPAccess97 C Programming

CRSP indices database have full C Programming support, including random access on `indno` and a common indices data structure used for all index series and portfolio groups.

Data Organization for C Programming

The basic levels of a CRSPAccess97 database are the database, set type, set id, module, object, and array. They are defined as follows:

- **Database (`CRSPDB`)** is the directory containing the database files. A `CRSPDB` is identified by the database path.

- **Set type** is a predefined type of financial data. Each set type has its own defined set of data structures, specialized access functions, and keys. CRSPAccess97 stock databases support stock (**`STK`**) and index (**`IND`**) set types. A `CRSPDB` can support more than one set type within the available CRSPAccess97 Stock and Indices products.

- **Set identifier (`SETID`)** is a defined subset of a set type. `SETID`s of the same set type use the same access functions, structures, and keys, but have different characteristics within those structures. For example, daily stock sets use the same data structure and monthly stock sets, but time series are associated with different calendars. Multiple `SETID`s of the same set type can be present in one `CRSPDB`.

- **Modules** are the groupings of data found in the data files in a `CRSPDB`. Multiple data items can be present in a module. Data is retrieved at a module level, and access functions retrieve data items for keys based on selected modules. Modules correspond to the physical data files.

- **Objects** are the fundamental data types defined for each set type. There are three fundamental object types, time series (**`CRSP_TIMESERIES`**), event arrays (**`CRSP_ARRAY`**), and headers (**`CRSP_ROW`**). Objects contain header information such as counts, ranges, or associated calendars (**`CRSP_CAL`**) plus arrays of data for zero or more observations. Some set types allow arrays of objects of one type. In this case, the number of available objects is determined by the `SETID`, and each of the objects in the list has independent counts, ranges, or associated calendars.

- **Arrays** are attached to each object. The array contains the set of observations and is the basic level of programming access. An observation can be a simple data type such as an integer for an array of volumes, or a complex structure such as for a name history. When there is an array of objects, there is a corresponding array of arrays with the data.

Data Objects

There are four basic types of information stored in CRSP databases. Each is associated with a CRSP object structure.

1. **Header Information**. These are identifiers with no implied time component.

2. **Event Arrays**. Arrays can represent status changes, random events, or observations. The time of the event and relevant information is stored for each observation. There is a count of the number of observations for each type of event data.

3. **Timeseries Arrays**. An observation is available for each period in an associated calendar. A beginning and ending point of valid data are available for each type of time series data. Data is stored for each period in the range – missing values are stored as placeholders if information is not available for a period.

4. **Calendar Arrays**. Each time series is tied to an array of relevant time periods. This calendar is used in conjunction with the time series arrays to attach times to the observations.

An observation can be a simple value or contain multiple components such as codes and amounts. Time series except Portfolios are based on calendars the frequency of the database. In a monthly database the time series are based on a month-end trading date calendar and in a daily database the time series are based on a daily trading date calendar. Portfolio calendars are dependent on the rebalancing methodology of the specific portfolio type. All calendars are attached automatically to each wanted time series object when the database is opened.

There are four primary building-block C structures called objects used in CRSPDBs. The following table contains each of the objects in all caps, followed by the components, lower case and indented, that each object type contains. All data items are defined in terms of the following objects:

C Object Structures Table

OBJECT or Field	Usage	DATA TYPE
CRSP_ARRAY	Structure for storing event-type data	
objtype	object type code identifies the structure as a CRSP_ARRAY, always = 3	int
arrtype	array type code defines the structure in the array. Base C types or CRSP-defined structures each have associated codes defined in the constants header file	int
subtype	subtype code defines a subcategory of array data. Subtypes further differentiate arrays with common array type fields.	int
size_of_array_width	width of the array in bytes	int
maxarr	maximum elements allocated in the array	int
num	number of elements in the array with available data for the current record	int
dummy	reserved, available as a secondary subtype code	int
arr	pointer to the array containing the actual data. The array can be a base C data type or a CRSP-defined structure. Its size and type are determined by arrtype, size_of_array_width, and maxarr.	void *
CRSP_ROW	Structure for storing header data	
objtype	object type code identifies the structure as a CRSP_ROW, always = 5	int
arrtype	array type code defines the structure in the array. Base C types or CRSP-defined structures each have associated codes defined in the constants header file	int
subtype	subtype code defines a subcategory of array data. Subtypes further differentiate arrays with common array type fields.	int
size_of_array_width	width of the array in bytes	int
arr	pointer to the array containing the actual data. The array can be a base C data type or a CRSP-defined structure. Its size and type are determined by arrtype and size_of_array_width. The maximum array size is always 1.	void *
CRSP_TIMESERIES	Structure for storing time series data	
objtype	object type code identifies the structure as a CRSP_TIMESERIES, always = 2	int
arrtype	array type code defines the structure in the array. Base C types or CRSP-defined structures each have associated codes defined in the constants header file	int
subtype	subtype code defines a subcategory of array data. Subtypes further differentiate arrays with common array type fields.	int
size_of_array_width	width of the array in bytes	int
maxarr	maximum elements allocated in the array	int
beg	first array index with valid data for the current record, or 0 if no valid range	int
end	last array index with valid data for the current record, or 0 if no valid range	int
caltype	code describing the type of time periods. caltype is always 2 indicating time periods are described by the last trading date in the period.	int

C Object Structures Table *(Con't)*

OBJECT or Field	Usage	DATA TYPE
cal	pointer to the calendar associated with the time series array. The calendar includes the matching period-ending dates for each array index.	CRSP_CAL *
arr	pointer to the array containing the actual data. The array can be a base C data type or a CRSP-defined structure. Its size and type are determined by arrtype, size_of_array_width, and maxarr.	void *
CRSP_CAL	Structure for storing calendar period data	
objtype	object type code identifies the structure as a CRSP_CAL, always = 1	int
calid	identifier assigned to each specific calendar by CRSP	int
type	generic group code of calendar, ie. daily or monthly	int
loadflag	code indicating the types of calendar arrays loaded. Currently = 2 for caldt only	int
ndays	index of last calendar period	int
name	text name of the calendar	char[80]
callist	reserved for array of alternate grouping identifiers for calendar periods	int *
caldt	array of calendar period ending dates, stored in YYYYMMDD. Calendars start at element 1 and end at element ndays	int *
daterng	reserved for array of periods described by the range of dates	CRSP_CAL_DATE_RANGE *
timerng	reserved for array of periods described by range of date and time	CRSP_CAL_TIMERANGE *
time	reserved for array of periods described by ending date and time	CRSP_CAL_TIME *
calmap	used to store array of first and last calendar period array elements in a linked calendar to elements in this calendar	CRSP_CAL_MAP *
basecal	used to point to a calendar linked in calmap	CRSP_CAL *

Set Structures and Usage

C top-level structures exist for each programming set type. These top level structures and built from general object and array structure definitions. Access functions for each set type identify a CRSPDB with a directory path and set identifier from a known set of numerical identifiers for the set available in the CRSPDB, and load data to a user-declared top-level structure of that type. Top level structures contain object and array pointers that have memory allocated to them by access open functions.

One set type and four set identifiers are currently supported for indices data. The identifier must be specified when opening or accessing data from the set.

Data Set Types and Identifiers

Data	Set Type	Set Identifiers	
CRSP Indices Data	IND	400	Monthly Groups
		420	Monthly Series
		440	Daily Groups
		460	Daily Series

Each set structure has three types of pointer definitions:

— **Module Pointers** point to CRSP_OBJECT_ELEMENT linked lists and are only needed internally to keep track of the objects in the module. These have the suffix _obj and can be ignored by ordinary programming.

— **Object Pointers** define a CRSP_ARRAY, CRSP_ROW, or CRSP_TIMESERIES object type. A suffix, _arr, _ts or _row is appended to the variable name. num, beg, and end are accessed from these variables.

— **Array Pointers** define the data item array. The second object pointer is a pointer to the array element of the object and is used for general access of the data item.

If a module has multiple types of objects, a group structure is created with definitions for those objects and is included in the main structure.

If a module has a variable number of objects of one type, an integer variable keeps track of the actual number. These variables end with the suffix types and are based on the set type.

Each of the top-level structures contains three standard elements:

— permno - the key loaded (actual key loaded is the indno)

— loadflag, a binary flag matching the set wanted parameters indicating which pointers have been allocated.

— setcode, a constant identifying the type of set (1=STK, 3=IND)

See the following sections for data objects and data items in the indices set structures.

C Language Data Objects for CRSP Indices Data

Indices data are accessed in C for individual series or portfolio group by an index identifier `indno`, assigned by CRSP. The index structure supports data for one series or group and includes header, rebalancing, and result information for the index.

Each index structure contains a fixed set of possible objects. Objects contain the header information needed to use the CRSP data structures as well as the data arrays. Data elements are described in the C data Structure Table under the array name.

Objects contain the header information needed to use the CRSP data structures. Data elements are described in the C Data Structure Table under the array name.

Time series `beg` and `end` are both equal to 0 if there is no data. Otherwise `beg > 0, beg <= end`, and `end < maxarr`. The 0^{th} element of a timeseries array is reserved for the missing value for that data type.

Multiple series in indexes usually refer to portfolio subgroups. Each of these will have the same `beg`, `end`, and calendar. In a `series setid`, the multiple series always has a count of 1. In a `group setid`, the count of series is found in the corresponding `xxxtypes` variables.

Index File C Objects Usage Table

Object	Name	objtype	arrtype	subtype	size_of_array_width	Range Elements On An Index Basis	Range Elements On Set Basis	Array Name
indhdr_row	indices header object	CRSP_ROW	CRSP_IND_HEADER_NUM = 200	0	300	none	none	ind.indhdr
rebal_arr[ ]	array of rebalancing arrays	CRSP_ARRAY	CRSP_IND_REBAL_NUM = 201	0	64	num for each series	maxarr, ind.rebaltypes	ind.rebal[j], j from 0 to ind.rebaltypes
list_arr[ ]	array of list arrays	CRSP_ARRAY	CRSP_IND_LIST_NUM = 202	0	24	num for each series	maxarr, ind.listtypes	ind.list[j], j from 0 to ind.listtypes
usdcnt_ts[ ]	array of used count time series	CRSP_TIMESERIES	CRSP_INTEGER_NUM = 2	CRSP_COUNT_NUM = 7	4	beg and end for each series	maxarr, cal, ind.indtypes	ind.usdcnt[j], j from 0 to ind.indtypes
totcnt_ts[ ]	array of total count time series	CRSP_TIMESERIES	CRSP_INTEGER_NUM = 2	CRSP_COUNT_NUM = 7	4	beg and end for each series	maxarr, cal, ind.indtypes	ind.totcnt[j], j from 0 to ind.indtypes
usdval_ts[ ]	array of used value time series	CRSP_TIMESERIES	CRSP_DOUBLE_NUM = 4	CRSP_WEIGHT_NUM = 4	8	beg and end for each series	maxarr, cal, ind.indtypes	ind.usdval[j], j from 0 to ind.indtypes
totval_ts[ ]	array of total value time series	CRSP_TIMESERIES	CRSP_DOUBLE_NUM = 4	CRSP_WEIGHT_NUM = 4	8	beg and end for each series	maxarr, cal, ind.indtypes	ind.totval[j], j from 0 to ind.indtypes
tret_ts[ ]	array of total return time series	CRSP_TIMESERIES	CRSP_FLOAT_NUM = 1	CRSP_RETURN_NUM = 2	4	beg and end for each series	maxarr, cal, ind.indtypes	ind.tret[j], j from 0 to ind.indtypes
aret_ts[ ]	array of capital appreciation time series	CRSP_TIMESERIES	CRSP_FLOAT_NUM = 1	CRSP_RETURN_NUM = 2	4	beg and end for each series	maxarr, cal, ind.indtypes	ind.aret[j], j from 0 to ind.indtypes
iret_ts[ ]	array of income return time series	CRSP_TIMESERIES	CRSP_FLOAT_NUM = 1	CRSP_RETURN_NUM = 2	4	beg and end for each series	maxarr, cal, ind.indtypes	ind.iret[j], j from 0 to ind.indtypes
tind_ts[ ]	array of total return index level time series	CRSP_TIMESERIES	CRSP_FLOAT_NUM = 1	CRSP_LEVEL_NUM = 3	4	beg and end for each series	maxarr, cal, ind.indtypes	ind.tind[j], j from 0 to ind.indtypes
aind_ts[ ]	array of capital appreciation index level time series	CRSP_TIMESERIES	CRSP_FLOAT_NUM = 1	CRSP_LEVEL_NUM = 3	4	beg and end for each series	maxarr, cal, ind.indtypes	ind.aind[j], j from 0 to ind.indtypes
iind_ts[ ]	array of income return index level time series	CRSP_TIMESERIES	CRSP_FLOAT_NUM = 1	CRSP_LEVEL_NUM = 3	4	beg and end for each series	maxarr, cal, ind.indtypes	ind.iind[j], j from 0 to ind.indtypes

C Language Data Structure for CRSP Indices Data

All CRSP-defined data types have names in all capitals beginning with `CRSP_` and are immediately followed by the definitions in the next level of indentation

Index and Date Ranges for all elements in a structure are the same as for the structure itself. There are four structure levels indicated by the indentation in the mnemonic column. The top level contains all other items and is used in all access functions. The second level indicates data grouped in modules. See Section 2.6 for data item definitions.

All character strings, indicated by `char[#],` are null terminated. The number of characters - 1 is the maximum string length allowed. Actual maximums may be lower. The top level `ind` structure is an example used by CRSP Indices sample programs. Other names can be used, and multiple `CRSP_IND_STRUCTs` can be declared in a program.

Index File C Variable Usage Table

Mnemonic	Name	Data Type	C Usage	Index Range	Date Usage	C Object Usage
ind	Master Indices Structure	CRSP_IND_STRUCT	ind			
indhdr	Indices Header Structure	CRSP_IND_HEADER *	ind.indhdr			ind.indhdr_row
indno	CRSP Permanent Index Number	int	ind.indhdr->indno			
indco	CRSP Permanent Index Group Number	int	ind.indhdr->indco			
primflag	Indno link to Primary Index	int	ind.indhdr->primflag			
portnum	Portfolio Number if Subset Series	int	ind.indhdr->portnum			
indname	Index Name	char[80]	ind.indhdr->indname			
typename	Index Group Name	char[80]	ind.indhdr->typename			
method	Index Creation Methodology Structure	CRSP_IND_METHOD	ind.indhdr->method			
methcode	Code of Basic Methodology Types	int	ind.indhdr->method.methcode			
primtype	Primary Methodology Group	int	ind.indhdr->method.primtype			
subtype	Secondary Methodology Group	int	ind.indhdr->method.subtype			
wgttype	Reweighting Type Flag	int	ind.indhdr->method.wgttype			
wgtflag	Reweighting Timing Flag3	int	ind.indhdr->method.wgtflag			
flags	Index Data Exception Flags Structure	CRSP_IND_FLAGS	ind.indhdr->flags			
flagcode	Code of Basic Exception Types	int	ind.indhdr->flags.flagcode			
addflag	Handling of New Issues	int	ind.indhdr->flags.addflag			
delflag	Handling of Ineligible Issues	int	ind.indhdr->flags.delflag			
delretflag	Return of Delisted Issues	int	ind.indhdr->flags.delretflag			
missflag	Handling Missing Data	int	ind.indhdr->flags.missflag			
partuniv	Partition Subset Screen Structure	CRSP_UNIV_PARAM	ind.indhdr->partuniv			
univcode	Code of Universe Subset Types	int	ind.indhdr->partuniv.univcode			
begdt	Beginning Date	int	ind.indhdr->partuniv.begdt			

Index File C Variable Usage Table *(Con't)*

Mnemonic	Name	Data Type	C Usage	Index Range	Date Usage	C Object Usage
enddt	Ending Data	int	ind.indhdr->partuniv.enddt			
wantexch	Valid Exchange Codes	int	ind.indhdr->partuniv.wantexch			
wantnms	Valid Nasdaq National Market Classification	int	ind.indhdr->partuniv.wantnms			
wantwi	Valid When-Issued Types	int	ind.indhdr->partuniv.wantwi			
wantinc	Valid Incorporation	int	ind.indhdr->partuniv.wantinc			
shrcd	Share Code Screen Structure	CRSP_UNIV_SHRCD	ind.indhdr->partuniv.shrcd			
sccode	Possible Share Code Restriction Types	int	ind.indhdr->partuniv.shrcd.sccode			
fstdig	Valid First Digits in Share Codes	int	ind.indhdr->partuniv.shrcd.fstdig			
secdig	Valid Second Digits in Share Codes	int	ind.indhdr->partuniv.shrcd.secdig			
induniv	Index Universe Screen Structure	CRSP_UNIV_PARAM	ind.indhdr->induniv			
univcode	Code of Universe Subset Types	int	ind.indhdr->induniv.univcode			
begdt	Beginning Date	int	ind.indhdr->induniv.begdt			
enddt	Ending Data	int	ind.indhdr->induniv.enddt			
wantexch	Valid Exchange Codes	int	ind.indhdr->induniv.wantexch			
wantnms	Valid Nasdaq National Market Classification	int	ind.indhdr->induniv.wantnms			
wantwi	Valid When-Issued Types	int	ind.indhdr->induniv.wantwi			
wantinc	Valid Incorporation	int	ind.indhdr->induniv.wantinc			
shrcd	Share Code Screen Structure	CRSP_UNIV_SHRCD	ind.indhdr->induniv.shrcd			
sccode	Possible Share Code Restriction Types	int	ind.indhdr->induniv.shrcd.sccode			
fstdig	Valid First Digits in Share Codes	int	ind.indhdr->induniv.shrcd.fstdig			
secdig	Valid Second Digits in Share Codes	int	ind.indhdr->induniv.shrcd.secdig			
rules	Index Rules Structure	CRSP_IND_RULES	ind.indhdr->rules			
rulecode	Code of Basic Rule Types	int	ind.indhdr->rules.rulecode			
buyfnct	Function Code for Buy Rules	int	ind.indhdr->rules.buyfnct			
sellfnct	Function Code for Sell Rules	int	ind.indhdr->rules.sellfnct			
statfnct	Function code for statistics calculations	int	ind.indhdr->rules.statfnct			
groupflag	Grouping Rules for Applying Statistics	int	ind.indhdr->rules.groupflag			
assign	Assignment Rules Structure	CRSP_IND_ASSIGN	ind.indhdr->assign			
assigncode	Code of Basic Assignment Type	int	ind.indhdr->assign.assigncode			
asperm	Indno of Associated Index for Breakpoints	int	ind.indhdr->assign.asperm			
asport	Portfolio Number in Associated Index	int	ind.indhdr->assign.asport			
rebalcal	Calid of Rebalancing Calendar	int	ind.indhdr->assign.rebalcal			
assigncal	Calid of Assignment Calendar	int	ind.indhdr->assign.assigncal			
calccal	Calid of Calculation Range Calendar	int	ind.indhdr->assign.calccal			

Index File C Variable Usage Table *(Con't)*

Mnemonic	Name	Data Type	C Usage	Index Range	Date Usage	C Object Usage
rebal	Rebalancing History	CRSP_IND_REBAL **	ind.rebal[j][i]	j between 0 and ind.rebaltypes, i between 0 and ind.rebal_arr[j]->num-1	data valid from ind.rebal[j][i].rbbegdt and ind.rebal[j][i].rbenddt	array of ind.rebal_arr
rbbegdt	Rebalancing Beginning Date	int	ind.rebal[j][i].rbbegdt			
rbenddt	Rebalancing Ending Date	int	ind.rebal[j][i].rbenddt			
usdcnt	Count used as of Rebalancing	int	ind.rebal[j][i].usdcnt			
maxcnt	Maximum Count During Period	int	ind.rebal[j][i].maxcnt			
totcnt	Available Count as of Rebalancing	int	ind.rebal[j][i].totcnt			
endcnt	Count at end of Period	int	ind.rebal[j][i].endcnt			
minid	Identifier at Minimum Value	int	ind.rebal[j][i].minid			
maxid	Identifier at Maximum Value	int	ind.rebal[j][i].maxid			
minstat	Smallest Statistic in Period	double	ind.rebal[j][i].minstat			
maxstat	Largest Statistic in Period	double	ind.rebal[j][i].maxstat			
medstat	Median Statistic in Period	double	ind.rebal[j][i].medstat			
avgstat	Average Statistic in Period	double	ind.rebal[j][i].avgstat			
list	Issue List Arrays	CRSP_IND_LIST **	ind.list[j][i]	j between 0 and ind.listtypes, i between 0 and ind.list_arr[j]->num-1	member of list from ind.list[j][i].begdt to ind.list[j][i].enddt	array of ind.list_arr
permno	Issue Identifier	int	ind.list[j][i].permno			
begdt	First Date Included	int	ind.list[j][i].begdt			
enddt	Last Date Included	int	ind.list[j][i].enddt			
subind	Subcategory within List	int	ind.list[j][i].subind			
weight	Weight During Range	double	ind.list[j][i].weight			
usdcnt	Portfolio Used Count Arrays	int **	ind.usdcnt[j][i]	j between 0 and indtypes-1, i between ind.usdcnt_ts[j]->beg and ind.usdcnt_ts[j]->end	value on date ind.usdcnt_ts[j]->cal->caldt[i]	array of ind.usdcnt_ts
totcnt	Portfolio Eligible Count Arrays	int **	ind.totcnt[j][i]	j between 0 and indtypes-1, i between ind.totcnt_ts[j]->beg and ind.totcnt_ts[j]->end	value on date ind.totcnt_ts[j]->cal->caldt[i]	array of ind.totcnt_ts
usdval	Portfolio Used Weight Arrays	double **	ind.usdval[j][i]	j between 0 and indtypes-1, i between ind.usdval_ts[j]->beg and ind.usdval_ts[j]->end	value on date ind.usdval_ts[j]->cal->caldt[i]	array of ind.usdval_ts
totval	Portfolio Eligible Weight Arrays	double **	ind.totval[j][i]	j between 0 and indtypes-1, i between ind.totval_ts[j]->beg and ind.totval_ts[j]->end	value on date ind.totval_ts[j]->cal->caldt[i]	array of ind.totval_ts
tret	Total Returns Arrays	float **	ind.tret[j][i]	j between 0 and indtypes-1, i between ind.tret_ts[j]->beg and ind.tret_ts[j]->end	value on date ind.tret_ts[j]->cal->caldt[i]	array of ind.tret_ts
aret	Capital Appreciation Arrays	float **	ind.aret[j][i]	j between 0 and indtypes-1, i between ind.aret_ts[j]->beg and ind.aret_ts[j]->end	value on date ind.aret_ts[j]->cal->caldt[i]	array of ind.aret_ts
iret	Income Return Arrays	float **	ind.iret[j][i]	j between 0 and indtypes-1, i between ind.iret_ts[j]->beg and ind.iret_ts[j]->end	value on date ind.iret_ts[j]->cal->caldt[i]	array of ind.iret_ts

Index File C Variable Usage Table *(Con't)*

Mnemonic	Name	Data Type	C Usage	Index Range	Date Usage	C Object Usage
tind	Total Return Index Level Arrays	float **	ind.tind[j][i]	j between 0 and indtypes-1, i between ind.tind_ts[j]->beg and ind.tind_ts[j]->end	value on date ind.tind_ts[j]->cal->caldt[i]	array of ind.tind_ts
aind	Capital Appreciation Index Level Arrays	float **	ind.aind[j][i]	j between 0 and indtypes-1, i between ind.aind_ts[j]->beg and ind.aind_ts[j]->end	value on date ind.aind_ts[j]->cal->caldt[i]	array of ind.aind_ts
iind	Income Return Index Level Arrays	float **	ind.iind[j][i]	j between 0 and indtypes-1, i between ind.iind_ts[j]->beg and ind.iind_ts[j]->end	value on date ind.iind_ts[j]->cal->caldt[i]	array of ind.iind_ts

C Sample Programs and Functions

There are two sample programs provided that can process a CRSPAccess97 Stock Database. If an indices-only database is used, the programs can be adapted to remove stock processing. If a CRSPAccess97 Indices database is overlaid on top of a Stock database, additional portfolio types and indices become available. The sample program code contains additional comment information. See system-dependent C programming instructions at the end of this section for instructions to run sample programs on supported systems. See C usage diagrams above and data item descriptions in Section 2.6 for possible data usage.

stk_samp1.c *stk_samp1.c* reads a stock database sequentially in `permno` order. It loads one index series before processing the stock data. As is, *stk_samp1.c* accepts parameters for database directory, stock set identifier, indices set identifier, index identifier `indno`, and output file name.

stk_samp2.c *stk_samp2.c* reads a stock database using an input file of `permnos`. It loads one set of indices before processing the input list. As is, stk_samp2.c accepts parameters for database directory, stock set identifier, indices set identifier, index identifier `indno`, input file name, and output file name.

C Header Files and Data Structures

Header files contain all CRSP needed structure definitions, constants, and function prototypes. Two C header files are sufficient to define all CRSP structures, constants, and functions.

crsp.h defines all structures and constants used by the CRSP C access and utility functions, and the function definitions. *crsp.h* includes several other header files. The primary definitions needed for the indices data are in *crsp_objects.h, crsp_const.h, crsp_ind_objects.h*, and *crsp_ind_const.h.*

crsp_init.h declares internal variables needed to store initialization and error information. This should only be included in the main program and not in any function modules.

The following list is a more complete summary of indices header files that are included by *crsp.h*. All header files are kept in the `CRSP_INCLUDE` directory.

Header File	Description
crsp_objects.h	defines all object structures and data array structures for all supported types
crsp_const.h	defines generic crsp constants
crsp_ind.h	top level indices header file includes all needed header files for CRSP indices access
crsp_ind_objects.h	defines top level `CRSP_IND_STRUCT` structure for indices data
crsp_ind_const.h	define indices constants and wanted parameters
crsp_access_ind.h	defines index access function prototypes
crsp_util_ind.h	defines index utility function prototypes
crsp_sysio.h	defines system-specific constants
crsp_maint.h	defines internal data structures

CRSPAccess97 C Index File Access Functions

The following tables list the available functions to access CRSPAccess97 index data. Standard usage is to use an open function, followed by successive reads and a close. Different databases and sets can be processed simultaneously if there is a matching structure defined for each one.

C Indices Access Functions

crsp_ind_close	Closes The Modules of an Indices Set
crsp_ind_free	Deallocates Memory and Reinitialize an Indices Set Structure
crsp_ind_init	Initializes Internal Indices Structures from init File
crsp_ind_open	Opens an Existing Indices Set in an Existing CRSPDB
crsp_ind_read	Loads Wanted Indices Data for an indno

crsp_ind_close Closes The Modules Of an Indices Set

Prototype:	int crsp_ind_close (int crspnum, int setid, CRSP_IND_STRUCT *indptr)
Description:	closes the modules of an index set
Arguments:	int crspnum - identifier of CRSP database, as returned by open int setid - identifier of the index set code to close, as used in open CRSP_IND_STRUCT *indptr - pointer to Indices structure shell
Return Values:	CRSP_SUCCESS: if successfully closed Indices set CRSP_FAIL: if error closing a file or illegal parameter
Side Effects:	All Indices module files are closed, memory allocated by them in the index structure is freed. If these are the last modules open in the database, the root is also closed. If indptr is NULL, no structure memory allocation is affected.
Preconditions:	The crspnum and setid must be taken from a previous crsp_ind_open call.
Call Sequence:	Called by external programs; must be preceded by a call to crsp_ind_open.

crsp_ind_free Deallocates Memory and Reinitializes an Indices Set Structure

Prototype:	int crsp_ind_free (int crspnum, int setid, CRSP_IND_STRUCT *indptr)
Description:	deallocates memory and reinitializes an index set structure
Arguments:	int crspnum - identifier of CRSPDB database, as returned by open int setid - identifier of the index set code to close CRSP_IND_STRUCT *indptr - pointer to index structure
Return Values:	CRSP_SUCCESS: if successfully deallocated and reset Indices structures CRSP_FAIL: if error deallocating memory or error in parameters
Side Effects:	The index structures are reset so all pointers are null and all settings are 0. All memory allocated to existing object is freed. There is no effect if indptr is NULL.
Preconditions:	The crspnum must be known from a previous crsp_ind_open call. The setid is a predefined identifier for the index daily or monthly series or group set of index data.
Call Sequence:	called by external programs or by crsp_ind_close

crsp_ind_init Initializes Internal Indices Structures from Init File

Prototype:	int crsp_ind_init (CRSP_IND_STRUCT *indptr)
Description:	initializes an index structure by setting all pointers to NULL and all counts to zero. Initializes CRSP internal structures if no previous initialization has been done.
Arguments:	CRSP_IND_STRUCT *indptr:- pointer to the index structure to be initialized. This argument can be NULL to initialize a CRSP internal database without resetting an existing structure.
Return Values:	CRSP_SUCCESS: if index internals successfully initialized CRSP_FAIL: if error opening or reading initialization file
Side Effects:	Internal structures will be initialized, including the array of known sets. They will be stored in internal structures in this module and used by other CRSP functions. All the pointers in the index structure indptr will be set to null. If a structure is already initialized with crsp_ind_open. crsp_ind_free should be used or memory will be lost.
Preconditions:	the initialization must be in the following format: # of known Indices sets [one row per Indices module with default buffersize, fillfactor, filename] for each Indices set: # name maxnam, maxdis, maxdel, maxshr, maxndi, maxports [optionally one row per module, with actual buffersize, fillfactor, filename]

crsp_ind_open Opens An Existing Index Set in an Existing CRSPDB

Prototype:
```
int crsp_ind_open (char *root, int setid, CRSP_IND_STRUCT *indptr, int wanted,
    char *mode, int bufferflag)
```

Description: opens an existing index set in an existing crspdb. This opens database files, allocates needed memory to a structure, and initializes internal structures to index data can be used.

Arguments:

char *root - path of root directory. If root is NULL, the CRSP_DSTK or CRSP_MSTK environment variables are used.

int setid - the set identifier
-monthly index series
-monthly index groups
-daily index series
-daily index groups

CRSP_IND_STRUCT *indptr - pointer to index structure to be associated with this database. If indptr is not NULL, then space for a CRSP_IND_STRUCT is allocated by this function.

int wanted - mask indicating which modules will be used The list below shows the wanted values for the index modules. the wanted values can be summed or summary wanted values, used to open multiple modules. Only modules that are selected in the wanted parameter have memory allocated in the index structure and only those modules can be accessed in further access functions to the database.

IND_HEAD	1	header structure and index description
IND_REBALS	2	rebalancing information for index groups
IND_LISTS	4	issue lists
IND_USDCNTS	8	portfolio used counts
IND_TOTCNTS	16	portfolio total eligible counts
IND_USDVALS	32	portfolio used weights
IND_TOTALS	64	portfolio eligible weights
IND_TRETURNS	120	total returns
IND_ARETURNS	256	capital appreciation returns
IND_IRETURNS	512	income returns
IND_TLEVELS	1024	total return index levels
IND_ALEVELS	2048	capital appreciation index levels
IND_ILEVELS	4096	income return index levels

Symbols are available for common groups of modules. IND_ALL selects all the index data.

IND_INFO	-IND_HEAD	IND_REBALS	IND_LISTS	
IND_RETURNS	-IND_TRETURNS	IND_ARETURNS	IND_IRETURNS	
IND_LEVELS	-IND_TLEVELS	IND_ALEVELS	IND_ILEVELS	
IND_COUNTS	-IND_HEAD	IND_TOTCNTS	IND_USDVALS	IND_TOTVALS
IND_RESULTS	-IND_HEAD	IND_USDCNTS	IND_USDVALS	IND_TRETURNS
IND_ARESULTS	-IND_HEAD	IND_USDCNTS	IND_USDVALS	IND_ARETURNS
IND_IRESULTS	-IND_HEAD	IND_USDCNTS	IND_USDVALS	IND_IRETURNS
IND_STD	-IND_HEAD	IND_COUNTS	IND_TRETURNS	IND_ARETURNS
IND_ALL	-IND_INFO	IND_RETURNS	IND_LEVELS	IND_COUNTS

char *mode - usage while open. Possible string values are "r"=read, "rw"=read/write)

int bufferflag - level of buffering: 0 : no buffering, 1 : use default, n : use factor of default

Return Values:

crspnum: if opened successfully, this crspnum is used in further access functions to the database.

CRSP_FAIL: if error opening or loading files, if bad parameters, root already opened exclusively, indices set already opened rw, wanted not a subset of set's modules, set does not exist in root, set already opened and structure allocated, error allocating memory for internal or Indices structures.

Side Effects: This will load root and Indices initialization files if needed, open the root including loading the configuration structure and index structures to memory, opening the address file, and if necessary allocating memory to file buffers, loading the free list, and logging information to the log file. Files will be opened for all wanted modules. Associated calendars will be loaded if necessary. Wanted Indices structures will be allocated.

Preconditions: None; the root may already be open under a different set in r mode

Call Sequence: crsp_ind_open must precede any calls to CRSP index read functions

crsp_ind_read Loads Wanted Indices Data for an indno

Prototype:	`int crsp_ind_read (int crspnum, int setid, int *key, int keyflag, CRSP_IND_STRUCT *indptr, int wanted)`
Description:	loads wanted index data for an `indno`

Arguments:	`int crspnum` - `crspdb` root identifier returned by `crsp_ind_open`
	`int setid` - the set identifier used in `crsp_ind_open`
	`int *key` - specific `indno` of data to load
	`int keyflag` – `CRSP_EXACT` constant to search for the `indno` in `*key`, or positional constant:
	`CRSP_FIRST` - the first key in the database
	`CRSP_PREV` - the next or previous key at the end of beginning of the file
	`CRSP_LAST` - the last key in the database
	`CRSP_SAME` - the same key
	`CRSP_NEXT` - the next key
	`CRSP_IND_STRUCT *indptr – structure to load data`
	`int wanted` - mask of flags indicating which module data to load. See `crsp_ind_open` for module codes.

Return Values:	Data from the wanted modules will be loaded to the proper location in the structure. The data loaded in the module buffers may be changed.
Preconditions:	The index set must be previously opened. The `setid, crspnum` and `indptr` are the same as opened and the `wanted` must be a subset of the `wanted open.`

C Access on Supported Systems

Windows NT Systems

CRSP supports Windows NT 3.51 and 4.0 on Intel x86 and Alpha machines. All C content was compiled and tested using the Microsoft Visual C++ 5.0 compiler.

CRSP access depends on environment variables set during installation of CRSPAccess97 Stock and Indices databases. The environment variables can be set in the Control Panel/System menu or in the `autoexec.bat` file. Environment variables can be used in command prompt windows with the name enclosed in `%` characters. The set command can be used in a command prompt window to show available environment variables. (e.g. `>set | find "crsp"`)

Important CRSP files and directories have the following names:

`%crsp_bin%`- folder containing executable sample programs and batch files. This folder should be in the `PATH` so programs can be run from any folder.

`%crsp_lib%`- folder containing CRSP Object Library and Internal Files.

`%crsp_lib%\crsp_lib.lib`- CRSP Object Library

`%crsp_include%`- location of CRSP C Header Files Referred to by `INCLUDE` Statements

`%crsp_sample%`- folder containing CRSP Sample Programs

`%crsp_mstk%`- folder containing Monthly CRSP Stock and Indices Databases

`%crsp_dstk%-` folder containing Daily CRSP Stock and Indices Databases

Following is an example of modifying a sample C program using Microsoft Visual C++:

Copy the sample program to a local directory using the Explorer (File Manager) utility or the command prompt copy command, or use the Developer Studio to open the file and save to a new location with Save As.

Sample programs can be found in the `%crsp_sample%` directory. The command `echo %crsp_sample%` can be used to get the explicit directory needed. The explicit paths for `%crsp_include%` and `%crsp_lib%` may be needed to set up projects in the Visual C++ Developers Studio,

To use Visual C++ Developer Studio:

- Create new project by choosing the New option from the File Menu.
- Choose Project Workspace from New window.
- Choose Console Application, from New Project Workspace window.

  use browse to choose a location which would be the location of your new project. Type the project name in the Name box. Choose create. This creates a project shell.

  Choose Configurations from the Build Menu.
- Choose Add in the Configurations window. From Add Project Configuration window type a new target name in the Configuration box. Choose OK, then choose Close to close the Configurations window.

Choose File into Project from the Insert Menu. From Insert Files into Project window, type the path and filename of the sample program in the File Name box and choose OK.

Repeat Step 3, but type the explicit path of `%crsp_lib%` followed by `crsp_lib.lib` in the File Name Box instead of the sample program path.

Choose Settings from the Build Menu. In Project Settings window:

- In C/C++ tab, choose General in Category box,

  * type "`,WINNT=2`" at the end of line in the Preprocessor definitions box,

  * Choose None in Debug Info box,

  * Choose Default in Optimizations box.

- In C/C++ tab, choose Precompiled Headers in Category box,

  * Choose not using precompiled header.

- In C/C++ tab, choose preprocessor,

  * Use explicit path of `%crsp_include%` to add include directory in the Additional Include directories.

- In C/C++ tab, choose all other choices, one at a time, disabling default options.

- In Link tab,

  * Can set an executable name as desired in Output File Name box,

  * Disable all other options.

- To Compile, choose Rebuild All from the Build Menu.

Alternately, use the command prompt from your local directory.

The location of C run-time libraries must be set in the `LIB` environment variable.

```
cl /I %crsp_include% /D WINNT=2 stk_sampl.c %crsp_lib%crsp_lib.lib
```

Sample programs can also be compiled and linked using the `nmake` utility. The file *c_samp.mak* in the `%crsp_sample%` directory is a description file to maintain the two stock sample programs. To run, copy the file to your program directory and run the utility with the command

```
nmake c_samp.mak
```

To run your sample program using arguments from a command prompt,

Make sure any input files exist and output files do not already exist. If no arguments it will print a usage message.

`stk_sampl %crsp_mstk% 20 420 1000080 file.out` (sample uses monthly data and loads CRSP value-weighted with dividends data, creating a header row for each `permno` in `file.out`).

See Microsoft documentation on the Visual C++, `NMAKE`, `CL`, and `LINK` commands.

See installation instructions in Appendix A for Windows NT to load data and configure the system to support CRSP data.

Unix Systems

CRSP supports Unix on Sun Sparcstations running Solaris and Digital Alpha machines running Digital Unix.

CRSP C Supported Unix Systems

	Sun Solaris	Digital Unix	HP/UX
CPU	Sun Sparc	Alpha	PARISC
Operating System	Solaris 2.5	Digital Unix	HP/UX
C Compiler	SparcCompiler C 4.0	DEC C 4.3	gnu C Compiler

CRSP access depends on environment variables set during installation. Environment variables can be used on Unix with the name preceded by `$`. All file names and environmental 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 Sample Programs and Batch Files. This directory is in the `PATH` so programs can be run from any directory.

`$CRSP_LIB`- directory containing CRSP Object Library and Internal Files.

`$CRSP_LIB/crsplib.a`- CRSP Object Library

`$CRSP_INCLUDE`- directory containing CRSP header files referred to by `#INCLUDE` statements

`$CRSP_SAMPLE`- directory containing CRSP Sample Programs

`$CRSP_MSTK`- directory containing monthly CRSP Stock and Indices Databases

`$CRSP_DSTK`- directory containing daily CRSP Stock and Indices Databases

Following is an example of how to modify and to run a sample C program with Sun Solaris and Digital UnixUnix:

- `cp $CRSP_SAMPLE/stk_sampl.c .` (copy the program to a local directory)
- `chmod 660 stk_sampl.c` (make sure you have write access to modify the program)
- `vi stk_sampl.c` (or other available editor, to insert alternate processing)
- `cc stk_sampl.c -o stk_sampl -DUNIX=l -I$CRSP_INCLUDE $CRSP_LIB/ crsplib.a` to compile the program.
- `stk_sampl $CRSP_MSTK 20 420 1000080 file.out` to run the program (sample uses monthly data and loads CRSP value-weighted with dividends data, creating a header row for each `PERMNO` in `file.out`). Make sure any input files exist and output files do not already exist. If no arguments the program will print a usage message.

Following is an example of how to modify and to run a sample C program with HP/UX:

- `cp $CRSP_SAMPLE/stk_sampl.c .` (copy the program to a local directory)
- `chmod 660 stk_sampl.c` (make sure you have write access to modify the program)
- `vi stk_sampl.c` (or other available editor, to insert alternate processing)
- `gcc stk_sampl.c -o stk_sampl -DUNIX=l -DUNIX=2 -I$CRSP_INCLUDE $CRSP_LIB/ crsplib.a` to compile the program.
- `stk_sampl $CRSP_MSTK 20 420 1000080 file.out` to run the program (sample uses monthly data and loads CRSP value-weighted with dividends data, creating a header row for each `PERMNO` in `file.out`). Make sure any input files exist and output files do not already exist. If no arguments the program will print a usage message.

Sample programs can also be compiled and linked using the `make` utility. The directory `$CRSP_SAMPLE` contains sample `make` description files for Sun Solaris and Digital Unix, *c_samp.mk* and *c_samp.mk2* respectively. 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, and run with the command

- `make -f filename`

See installation instructions in Appendix A for UNIX to load data and configure the system to support CRSP data.

OpenVMS Systems

CRSP supports OpenVMS on Digital Alpha machines. C Functions were compiled with DEC C 5.3.

CRSP access depends on logicals set during installation. The `show logical crsp*` command can be used in a terminal window to find available CRSP logicals.

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

`CRSP_BIN:` - directory containing Executable Programs and Batch Files.

`CRSP_LIB:` - directory containing CRSP Object Library and Internal Files.

`CRSP_LIB:libcrsp.olb` - CRSP Object Library

`CRSP_INCLUDE:` - directory containing CRSP header files referred to by `INCLUDE` Statements

`CRSP_SAMPLE:` - directory containing CRSP Sample Programs

`CRSP_MSTK:` - directory containing Monthly CRSP Stock and Indices database

`CRSP_DSTK:` - directory containing Daily CRSP Stock and Indices database

Following is an example of modifying and running a sample C program with OpenVMS:

```
copy crsp_sample:stk_sampl.c [ ]
```
(copy the program to a local directory)

```
eve stk_sampl.c
```
(or other available editor, to insert alternate processing)

```
cc stk_sampl /include=CRSP_INCLUDE:/define=(VMS=l)/float=IEEE_FLOAT
```

```
link stk_sampl + crsp_lib:libcrsp/lib + sys$1ibrary:vaxcrtlt.olb/lib
```

```
stk_sampl :== "$curdir:stk_sampl.exe"
```
(`curdir` is full path to location of executable, needed to run the program with arguments)

```
stk_sampl crsp_mstk: 20 420 1000080 file.out
```
(sample arguments use monthly data and loads CRSP value-weighted with dividends data, creating a header row for each `PERMNO` in file.out. A usage message will be displayed if the program is run without arguments.)

Sample programs can also be compiled and linked using the `MMS` utility. A sample description file, *C_SAMP.MMS*, exists in the `CRSP_SAMPLE:` directory. To use the sample description file, copy it to your program directory, modify it to include your program, and run with the command

```
mms / description=c_samp
```

See installation instructions in Appendix A to load data and configure the system to support CRSP data.

APPENDIX

A. INSTALLATION AND DATABASE ADMINISTRATION

CRSPAccess97 databases are distributed on CD-ROM. The code and data for all supported systems are distributed with each product set. Setup programs are provided on the first disk. See the machine-specific installation sections for details of installation on a particular system. Utility programs to convert CRSP databases to previous CRSP binary formats and create subset databases are described at the end of this section.

Installation consists of the following basic steps:

1. Copy program libraries and executables to a root directory on target machine
2. Copy data to a directory on the target machine
3. Set CRSP environment variables needed to use the CRSP programs

The installer must determine locations for program files, data, and log files before installation. Multiple database installations can exist if desired and disk space allows. Documentation exists in the doc subdirectory of the CD, but is not loaded by the setup programs. Program files are identical in each product and should only be copied once. Log directories are shared by all databases.

Use the CRSPAccess97 Products table in the CRSPAccess97 Release Notes supplement to determine the disk space needed for each CRSP product.

A.1 Using Multiple CRSP Products

CRSPAccess97 programs can support a daily and monthly database and indices. Each database includes security data with capitalization portfolio information plus a value-weighted, equal-weighted, and composite index. CRSP daily or monthly indices can be added to a corresponding stock database to add additional portfolio types and indices. Stand alone, fixed format ASCII Indices files are also included with the CRSPAccess97 Indices File / Portfolio Assignments Product. **Install the Indices last if both Stock and Indices are available**.

The programs supplied with each product are identical and should only be loaded once.

The log directory is shared by all databases and needs to be created only once. The log directory should be set in a scratch area where all intended users have write access.

CD-ROM Layout

- top level contains setup programs, CD descriptions, copyright information, and three folders
- root - contains programs, libraries, header files, and sample programs. Root is only included on the first CD of a multiple CD product
  - `bin` contains executable programs and scripts; there is a subdirectory for each supported operating system / platform
    - `wnt40x86` - Windows NT 4.0 on Intel x86
    - `w9540x86` - Windows 95 4.0 on Intel x86
    - `vms65axp` - OpenVMS 6.2 on AXP
    - `sol25spa` - Solaris 2.5 on Sparc
    - `unx40axp` - Digital Unix 4.0 on AXP
    - `wnt40axp` - Windows NT 4.0 on AXP
    - `hpu10par` - HP/UX Unix 10.2 on PARISC
  - `lib` contains object libraries and initialization files; there is a subdirectory for each supported operation system / platform
    - `wnt40x86` - Windows NT 4.0 on Intel x86
    - `w9540x86` - Windows 95 4.0 on Intel x86
    - `vms65axp` - OpenVMS 6.2 on AXP
    - `sol25spa` - Solaris 2.5 on Sparc
    - `unx40axp` - Digital Unix 4.0 on AXP
    - `wnt40axp` - Windows NT 4.0 on AXP
    - `hpu10par` - HP/UX Unix 10.2 on PARISC
  - `include` - FORTRAN and C header files
  - `sample` - FORTRAN and C sample programs
  - `src` - C source code
  - `forsrc` - FORTRAN source code
- `data` - contains actual database files. Bottom level directories exist as needed based on the product and part of the set.
  - `ieeebig` - IEEE big-endian database files (Sun, HP). Subdirectories are named as the product code and the year and month of data
    - `diYYYYMM`
    - `miYYYYMM`
    - `dxYYYYMM`
    - `daYYYYMM`
    - `maYYYYMM`
    - `mzYYYYMM`
  - `ieeelit` - IEEE little-endian database files (PC, Alpha) and text files
    - `inYYYYMM`
    - `diYYYYMM`
    - `miYYYYMM`
    - `dxYYYYMM`
    - `daYYYYMM`
    - `maYYYYMM`
    - `mzYYYYMM`
- `doc` - CRSP documentation in different formats
  - `bin` - Adobe Acrobat `.pdf` versions of the Indices File Guide and Release Notes.
  - `html` - html versions of the Indices File Guide and the Release Notes. (The file contains relative links beginning with the `caicovno.html` file, which contains the table of contents.)
  - `word` - Microsoft Word 97 versions of the Indices File Guide and Release Notes.

A.2 Installation

Windows NT Installation on Intel x86 Computers

The setup program will copy program and data files to your machine and can also be used to set CRSP environment variables. To run the setup program:

1. login to an account with administrator permissions to set CRSP for all users on the machine.
2. load the first disk of your product to your CD drive,
3. double click on the `setup.exe` application, and
4. follow the instructions in the setup menus.

When the setup program asks for a database folder, use the `browse...` option to choose a folder other than the default. A folder can be typed in or selected with the menus. If the folder does not exist the program will ask if you wish to create it. In these screens, the `Back` option is used to skip copying data and does not return to a previous screen. If the program asks for a second CD ROM, make sure the computer registers the new CD ROM before continuing.

If the root has been loaded from one product, the section should be skipped in succeeding products. Copy program libraries and executables to a root directory on target machine.

Install the Indices last if both Stock and Indices are available.

An administrator can set permissions on directories as needed. Write access is not needed for any program or data directories. The log directory should have full write access.

Environment Variables

Environment variables must be set for the CRSP programs to work properly. Environment variables can be set at the system or user level. At the system level all accounts will have access to the CRSP data. A user with a local copy of the database can set the environment variables at the user level so other users are not affected.

The setup application can be used for changing environment variables in the initial setup or if files are moved. In the initial setup, environment variables can be set based on the destination directories used for program files or data. Run the application and skip the sections up to the environment setup to make changes.

The Environment tab of the Control Panel / System Window can be used to modify environment variables. The `PATH` may need setting by hand. Use the delete option to CRSP environment variables that are no longer needed. The `set` command in a command prompt window can also be used to show the current assignments.

The following list of environment variables are used in CRSPAccess97.

Environment variable	**Usage**
`CRSP_ROOT`	Top level program directory. Most other CRSP environment variables are set based on `CRSP_ROOT`
`CRSP_LOG`	Log directory used for user
`CRSP_MSTK`	CRSP Monthly Database directory
`CRSP_DSTK`	CRSP Daily Database directory
`CRSP_INCLUDE`	Programming header files; include subfolder of root
`CRSP_SAMPLE`	Sample programs; sample subfolder of root
`CRSP_LIB`	Object libraries; `acclib` subfolder of root
`CRSP_BIN`	Executables and scripts; `accbin` subfolder of root
`CRSP_ENV_ULOG`	Usage logs produced by users; `=CRSP_LOG`
`CRSP_ENV_ELOG`	Error logs produced by users; `=CRSP_LOG`
`CRSP_ENV_EMSG`	Location of error messages file; `=CRSP_LIB`
`CRSP_ENV_LMSG`	Location of log messages file; `=CRSP_LIB`
`CRSP_ENV_ROOT`	Location of base CRSPAccess97 database
`CRSP_ENV_PATH`	Location of default CRSPAccess97 database
`CRSP_ENV_INIT`	Location of initializations file; `CRSP_LIB`
`PATH`	The `CRSP_BIN` folder must be included in the `PATH` list of directories to run CRSP programs

Windows 95 Support

The setup application provided with Windows NT will work on a Windows 95 machine but cannot set environment variables. The environment variables must be included in the `autoexec.bat` file on the root hard drive. The `accbin` folder in the CRSP `root` folder contains a file called `crsp_symbols.bat`. This file must be edited to reflect the local CRSP file locations, and either called by the `autoexec.bat` file on the system disk or added directly to the `autoexec.bat` file. See the Windows NT section for Intel x86 for descriptions of the environment variables.

Windows NT Support on Digital Alpha

The setup application is not supported on Alpha Windows NT. Files must be copied by hand with the NT Explorer and environment variables must be set directly in the Control Panel.

Copying Files

Files are copied with the NT Explorer Application:

1. Create a new folder in the target location for CRSP root files (skip 1-5 if not the first CRSP product). The folder must be visible on the left portion of Explorer.
2. Double-click on the `root` folder of the CD. Drag the `include` and `sample` folders to the new root folder.
3. Double-click on the `lib` folder. Drag the `wnt40axp` folder to the new `root` folder. Highlight and select rename with the right mouse button. Type the folder name `acclib`.
4. Select the `bin` folder under root on the CD. Drag the `wnt40axp` folder to the new `root` folder. Highlight and select `rename` with the right mouse button. Type the folder name `accbin`.
5. Create a new folder in a scratch area for CRSP `log` files.
6. Create a new folder in the target location for the CRSP `data` files.
7. Navigate the CD down the data and `ieeelit` folders. Double click on the data directory.
8. Use `Select All` and Copy commands in the `Edit` menu.
9. Select the target folder and choose the `Paste` command from the `Edit` menu.

The IX product contains extra files for the daily or monthly stock databases. To overlay the additional files, select the data folder with the first two letters matching the existing stock database. Select and copy the files in the folder and paste into the existing database directory. Files will match existing files in this folder. Choose `Yes` to `All` to overwrite.

If the root has been loaded from one product, the section should be skipped in succeeding products. Copy program libraries and executables to a root directory on target machine.

Install the Indices last if both Stock and Indices are available.

Environment Variables

Environment variables are entered by hand with the Environment tab of the System window of the Control Panel. You must be logged into an account with Administrator permissions to set system level variables. System level variables are accessible by all users on the computer. User level variables take precedence over system variables, but only for the user setting the variables.

Environment variable	Set To (end paths with \ unless otherwise noted)
CRSP_ROOT	Path of `root` folder created in step 1
CRSP_LOG	Path of `log` folder created in step 5 (no ending \)
CRSP_MSTK	Path of CRSP Monthly Database directory
CRSP_DSTK	Path of CRSP Daily Database directory
CRSP_INCLUDE	Path of include subfolder of root
CRSP_SAMPLE	Path of sample subfolder of root (not ending \)
CRSP_LIB	Path of `acclib` subfolder of root
CRSP_BIN	Path of `accbin` subfolder of root
CRSP_ENV_ULOG	Same as CRSP_LOG
CRSP_ENV_ELOG	Same as CRSP_LOG
CRSP_ENV_EMSG	Same as CRSP_LIB
CRSP_ENV_LMSG	Same as CRSP_LIB
CRSP_ENV_ROOT	Path of base CRSPAccess97 database
CRSP_ENV_PATH	Path of default CRSPAccess97 database
CRSP_ENV_INIT	Same as CRSP_LIB
PATH	Add a semicolon and the CRSP_BIN path to the end of the current PATH (no ending \ on CRSP_BIN path)

Solaris Installation on Sun Sparcstation

The setup shellscript will copy program and data files to your machine. If overlaying indices on top of an existing stock database, use the `chmod` command to make sure you have permission to overwrite files. There is a program file script included that can create a machine-specific `shellscript` that can be called by any user to set CRSP environment variables. To run the setup script:

1. load the first disk of your product to your CD drive. The system should automatically mount it
2. in a terminal window, `cd` to a local directory. Use the `df` command to find the path of the CDROM
3. copy the `setup.sh` file from the top level directory of the CDROM to your local directory, for example

```
cp /cdrom/ixx1-199712/setup.sh .
```

4. execute the script:

```
setup.sh
```

5. answer prompts as needed. Directories be specified without trailing slashes.
6. if the program prompts for another CD, eject the current CD and insert the requested CD from outside the terminal window running the program. The `eject cdrom` command at a second terminal window or the eject disk button in the file manager can be used to eject a CD.

If the root has been loaded from one product, the section should be skipped in succeeding products.

Digital Unix Installation on Digital Alpha

The setup shellscript will copy program and data files to your machine. If overlaying indices on top of an existing stock database, use the `chmod` command to make sure you have permission to overwrite files. There is a program file script included that can create a machine-specific shellscript that can be called by any user to set CRSP environment variables. To run the setup script:

1. load the first disk of your product to your CD drive. You must know the location on your system of the CDROM drive.

2. Mount the CD with the command (device `/dev/rz4c` is an example here and may be different on your machine)

```
mount –t cdfs –o noversion /dev/rz4c /cd
```

3. in a terminal window, `cd` to a local directory. The path of the CDROM is `/cd`.

4. copy the setup.sh file from the top level directory of the CDROM to your local directory, for example

```
cp /cd/setup.sh
```

5. execute the script:

```
setup.sh
```

6. answer prompts as needed. Directories should not be specified with trailing slashes.

7. if the program prompts for another CD, eject the current CD and insert the requested CD from outside the terminal window running the program. The `umount /cd` command at a second terminal can be used to eject a CD. Load the new CD with the same command as in step 2.

8. Use `umount /cd` to dismount the CD

If the root has been loaded from one product, the section should be skipped in succeeding products.

HP/UX Installation

The setup shellscript will copy program and data files to your machine. If overlaying indices on top of an existing stock database, use the `chmod` command to make sure you have permission to overwrite files. There is a program file script included that can create a machine-specific shellscript that can be called by any user to set CRSP environment variables. To run the setup script:

1. Load the first disk of your product to your CD drive. You must know the location on your system of the CDROM drive.

2. Mount the CD with the command (device `/dev/dsk/c02d0` is an example here and may be different on your machine)

   ```
   mount/dev/dsk/c0t2d0 /cdrom
   ```

3. In a terminal window, `cd` to a local directory. The path of the CDROM is `/cdrom`

4. Copy the `setup.sh` file from the top level directory of the CDROM to your local directory, for example

   ```
   cp /cdrom/setup.sh
   ```

5. Execute the script:

   ```
   setup.sh
   ```

6. Answer prompts as needed. Directories should be specified **without** trailing slashes.

7. If the program prompts for another CD, eject the current CD and insert the requested CD from outside the terminal window running the program. The `umount /cdrom` command at a second terminal can be used to eject a CD. Load the new CD with the same command as in step 2.

8. Use `umount /cdrom` to dismount the CD

If the root has been loaded from one product, the section should be skipped in succeeding products.

Unix Environment Variables

Environment variables must be set for the CRSP programs to work properly on any of the Unix platforms. CRSP provides shellscripts that prompts for directories and creates either Unix `ksh` or `csh` scripts that can be used to set CRSP definitions.

1. `cd` to the root directory where program files have been loaded.

2. `cd accbin`

3. if you are running csh shell
   `source crsp_setup.csh`

   if you are running ksh shell,
   `crsp_setup.sh`

4. the script will prompt for data, root, and log directories. Follow the instructions on the prompts in terms of trailing slashes in directory names.

5. The script will create to new scripts, `crsp.cshrc` in `csh` or `crsp.kshrc` in `ksh`.

6. The new scripts can be incorporated into all accounts by the system administrator, or sourced into initialization files directly by individual users.

One method of incorporating CRSP initializations for `csh` users is:

Move the file `crsp.cshrc` to the `/local/bin` or other common directory

Add `source /local/bin/crsp.cshrc` to the `.cshrc` file of each account using CRSP

One method of incorporating CRSP initialization for `ksh` users is:

Move the file `crsp.kshrc` to the `/local/bin` or other common directory

Add `. /local/bin/crsp.kshrc` to the `.kshrc` file of each account

Add the line `ENV=$HOME/.kshrc` to the `/etc/.login` file

`env | grep CRSP` can be used to check the CRSP environment variables set

The script produced by the CRSP setup script also adds the `$CRSP_BIN` directory to the users' path.

The following list of environment variables are used in CRSPAccess97.

Environment variable	Usage
CRSP_ROOT	Top level program directory. Most other CRSP environment variables are set based on CRSP_ROOT
CRSP_LOG	Log directory used for user
CRSP_MSTK	CRSP Monthly Database directory (if available)
CRSP_DSTK	CRSP Daily Database directory (if available)
CRSP_INCLUDE	Programming header files
CRSP_SAMPLE	Sample programs
CRSP_LIB	Object libraries
CRSP_BIN	Executables and scripts
CRSP_ENV_ULOG	Usage logs produced by users
CRSP_ENV_ELOG	Error logs produced by users
CRSP_ENV_EMSG	Location of error messages file
CRSP_ENV_LMSG	Location of log messages file
CRSP_ENV_ROOT	Location of base CRSPAccess97 database
CRSP_ENV_PATH	Location of default CRSPAccess97 database
CRSP_ENV_INIT	Location of initializations file
PATH	The CRSP_BIN folder must be included in the PATH list of directories to run CRSP programs

OpenVMS Installation on Digital Alpha

The setup command file will copy program and data files to your machine. There are also command files included that can be called on a user, group, or system level to set CRSP logicals and symbols. To run the setup command file:

1. Load the first disk of your product to your CD drive.
2. Mount the CD with the command (the LABEL is the first 11 characters of your external label, and LABEL#1 is the same label followed by the characters #1)

```
mount /media=cd cd LABEL /undefined_fat=(stream_lf:500) - /shared
/bind=LABEL#1
```

3. In a different terminal, set default to a local directory. The path of the CD ROM is `cd:`.
4. Copy the setup.com file from the top level directory of the CD ROM to your local directory, for example

```
copy cd:[000000]setup.com []
```

5. Execute the command file:

```
@setup
```

6. Answer prompts as needed. The CD location is `CD:[000000]`. The root directory must be specified with a full directory path, ending with a period following the last subdirectory, and no closing bracket. Data directories must have the closing bracket.
7. If the program prompts for another CD, return to the terminal that mounted the first CD and execute the command `dismount cd:`. Insert the requested CD and mount with the same command as step 2 but using the new label based on the first eleven characters of the new CD's external label.
8. Use `dismount cd:` when finished to dismount the CD

If the root has been loaded from one product, the section should be skipped in succeeding products. Copy program libraries and executables to a root directory on target machine.

Install the Indices last if both Stock and Indices are available.

Logicals and Symbols

Logicals and symbols must be set for the CRSP programs to work properly. CRSP provides command files that can be used at a system, group, or process level to set directories and program symbols. They are located in the `[.BIN]` subdirectory of the root directory where program files have been loaded.

`crsp_logicals_install.com` sets all directories of databases and programs. It is run with six arguments:

1. One of `P` or `G` or `S` - this determines whether logicals will be loaded to the process, group or system logical names table. The account must have `GRPNAM` permission to load to the group table, and `SYSNAM` permission to load to the system table
2. The root directory in the form `disk:[root.]`
3. A log directory in the form `disk:[log.]`. The directory and two subdirectories must exist. The names of the two subdirectories are `[.usage]` and `[.errs].` The log disk and directories should be in a scratch disk with write access to any CRSP user.
4. A CRSP Daily Database directory. A monthly database directory can be substituted if no daily database directory exists on the system.

5. A CRSP Monthly Database directory. A daily database directory can be substituted if no monthly database directory exists on the system.

6. A . (period)

`crsp_symbols_install.com` defines program names so they can be run with arguments. This command file is run with no arguments.

The two initialization files should be called by a system initialization or a user `login.com` so they are available to targeted users at login. A single user can run them in a local process to set up a personal copy of a database. The command show `logicals crsp*` can be used to verify the logicals set.

The following list of logicals are used in CRSPAccess97.

Logical	Usage
CRSP_ROOT	Top level program directory. Most other CRSP environment variables are set based on CRSP_ROOT. CRSP_ROOT is set as a concealed logical.
CRSP_LOG	Log directory used for user files. CRSP_LOG is set as a concealed logical
CRSP_MSTK	CRSP Monthly Database directory
CRSP_DSTK	CRSP Daily Database directory
CRSP_INCLUDE	Programming header files
CRSP_SAMPLE	Sample programs
CRSP_LIB	Object libraries
CRSP_BIN	Executables and scripts
CRSPDB_ULOGDIR	Usage logs produced by users
CRSPDB_ELOGDIR	Error logs produced by users
CRSPDB_EMSGDIR	Location of error messages file
CRSPDB_LMSGDIR	Location of log messages file
CRSPDB_ROOTDIR	Location of base CRSPAccess97 database
CRSPDB_PATHDIR	Location of default CRSPAccess97 database
CRSPDB_INITDIR	Location of initializations file

A.3 CRSP Database Utilities

CRSP provides two utilities that can be used to manipulate CRSPAccess97 Indices databases.

ind_partial

This program creates a new CRSPAccess97 index database from an existing database or appends indices from one existing database to another. It can use an `indno` list or a data type restriction to subset the original database. It takes parameters on input and output databases, input and output set identifiers, data wanted in the new database, and optionally a file containing `INDNOs` to copy to the new database. Standard stock databases contain stock and indices sets.

Parameter	Values
Input `CRSPDB` Directory Path	The directory where the database is stored. Standard environment names can be used such as `$CRSP_DSTK` on UNIX, `crsp_dstk:` on OpenVMS, or `%crsp_dstk%` for Windows NT.
Output `CRSPDB` Directory Path	The directory where the new database will be stored. This can be an empty directory or an existing directory. If it is an empty directory, a new database will be created. If there is already a `CRSPDB` in that directory, the selected `indnos` will be added to that database.
Input Index `setid`	The database set type. Use one of: 400 if a monthly groups 420 if monthly series 440 if daily groups 460 if daily series
Output Index `setid`	The database set type. Input and output index `setids` should be the same.
Set Wanted	A binary flag to determine the index modules that will be supported in the new database. Use `8191` to support all current modules. A module that is not loaded at this time cannot be added later to that database.

Data Wanted	A binary flag to determine which modules will be copied to the new database. Use `8191` to copy all data to the new database. Data wanted must be a subset of set wanted. Individual wanted codes can be summed to load multiple modules. Individual modules codes are:
	`1 = headers`
	`2 = rebalancing information for index groups`
	`4 = issue lists`
	`8 = portfolio used counts`
	`16 = portfolio total eligible counts`
	`32 = portfolio used weights`
	`64 = portfolio eligible weights`
	`128 = total returns`
	`256 = capital appreciation returns`
	`512 = income returns`
	`1024 = total return index levels`
	`2048 = capital appreciation index levels`
	`4096 = income return index levels`

`indno` List File	The name of a file with a list of `indnos`, one to a line. This parameter is optional. If it is used, only the `indnos` in the input file will have data copied to the new database. If the parameter is not used, all `indnos` in the input database will be copied.

crsp_ind_headall

This program creates header files for an index database. It is useful primarily for a subset database. If the files are created in the same directory as the database, and the `CRSP_MSTK` or `CRSP_DSTK` environment points to the database, the index search utilities will function with that database.

Parameters are an input database and `setid` and one output file. The output file includes `indno`, `setid`, and index description.

Parameter	Values
Input `CRSPDB` directory path	The directory where the database is stored. Standard environment names can be used such as `$CRSP_DSTK` on UNIX, `crsp_dstk:` on OpenVMS, or `%crsp_dstk%` for Windows NT.
Input Stock `setid`	The database set type. Use one of: 400 if a monthly goups 420 if monthly series 440 if daily groups 460 if daily series

A.4. *ts_print* Maintenance

The *ts_print* data utility creates temporary files in the user log directory setup on the system during installation. It might become necessary to clean this directory periodically if circumstances arise where *ts_print* is unable to clean these files.

The log directory is defined during installation. The environment variable name on Unix or Windows NT systems is `CRSP_ENV_ULOG`. The logical directory name on OpenVMS is `CRSPDB_ULOGDIR:`. Any user with the proper permissions can erase all files in this directory if necessary.

B. TEXT FILES

B.1 ASCII Character File Names

The following table lists ASCII text file names loaded from the CD. Refer to the CD layout for directory location.

Name	Description	Record Length
DSIA.DAT	Daily NYSE Indices/Cap. Deciles	466
MSIA.DAT	Monthly NYSE Indices/ Cap. Deciles	466
QSIA.DAT	Quarterly NYSE Indices/Cap. Deciles	466
ASIA.DAT	Annual NYSE Indices/ Cap. Deciles	466
DSIB.DAT	Daily AMEX Indices/Cap. Deciles	466
MSIB.DAT	Monthly AMEX Indices/Cap. Deciles	466
QSIB.DAT	Quarterly AMEX Indices/Cap. Deciles	466
ASIB.DAT	Annual AMEX Indices/Cap. Deciles	466
DSIC.DAT	Daily NYSE/AMEX Indices/Cap. Deciles	466
MSIC.DAT	Monthly NYSE/AMEX Indices/Cap. Deciles	466
QSIC.DAT	Quarterly NYSE/AMEX Indices/Cap. Deciles	466
ASIC.DAT	Annual NYSE/AMEX Indices/Cap. Deciles	466
DSIO.DAT	Daily NASDAQ Indices/Cap. Deciles	466
MSIO.DAT	Monthly NASDAQ Indices/Cap. Deciles	466
QSIO.DAT	Quarterly NASDAQ Indices/Cap. Deciles	466
ASIO.DAT	Annual NASDAQ Indices/Cap. Deciles	466
DSIX.DAT	Daily NYSE/AMEX/NASDAQ Indices	466
MSIX.DAT	Monthly NYSE/AMEX/NASDAQ Indices	466
QSIX.DAT	Quarterly NYSE/AMEX/NASDAQ Indices	466
ASIX.DAT	Annual NYSE/AMEX/NASDAQ Indices	466
DSBC.DAT	Daily NYSE/AMEX Indices/Beta Deciles	466
DSSC.DAT	Daily NYSE/AMEX Indices/Std. Deviation Deciles	466
DSBO.DAT	Daily Nasdaq Indices/Beta Deciles	466
DSSO.DAT	Daily Nasdaq Indices/Std. Dev.iation Deciles	466
MHISTN.DAT	Monthly Cap-Based NYSE Results	92
MHISTA.DAT	Monthly Cap-Based NYSE/AMEX Results file	92
MHISTQ.DAT	Monthly Cap-Based NYSE/AMEX/NASDAQ Nationalresults file	92
REBALN.DAT	Quarterly Cap-Based NYSE Results	100
REBALA.DAT	Quarterly Cap-Based NYSE/AMEX Results file	100
REBALQ.DAT	Quarterly Cap-Based NYSE/AMEX/NASDAQ National Market rebalancing file	100
DSP500.DAT	Daily CRSP Index file on the S&P 500®	130
MSP500.DAT	Monthly CRSP Index file on S&P 500®	130
MCTI.DAT	Monthly CTI file	288
QCTI.DAT	Quarterly CTI file	288
ACTI.DAT	Annual CTI file	288

B.2 File Version Specifics

This section contains version specific information for the CRSPAccess97 Indices Text Files with data ending December 31, 1997. The table below contains the number of records, the applicable dates and records, and the file sizes in character format for each file. The calendar is always the same for all files with the same frequency. All file sizes are in kilobytes and rounded to the nearest kilobyte.

File Name	Number of Records	Calendar Range	Index Range	Size Character
NYSE Only				
DSIA.DAT	8939	620702-971231	1-8939	4166
MSIA.DAT	865	251231-971231	1-865	403
QSIA.DAT	289	251231-971231	1-289	135
ASIA.DAT	73	251231-971231	1-73	34
AMEX Only				
DSIB.DAT	8939	620702-971231	1-8939	4166
MSIB.DAT	865	620731-971231	440-865	403
QSIB.DAT	289	620958-971231	148-289	135
ASIB.DAT	73	621231-971231	38-73	34
NYSE And AMEX Combined				
DSIC.DAT	8939	620702-971231	1-8939	4166
MSIC.DAT	865	251231-971231	1-865	403
QSIC.DAT	289	251231-971231	1-289	135
ASIC.DAT	73	251231-971231	1-73	34
NASDAQ Only				
DSIO.DAT	8939	731214-971231	2610-8939	4166
MSIO.DAT	865	731229-971231	565-865	403
QSIO.DAT	289	731229-971231	189-289	135
ASIO.DAT	73	731229-971231	48-73	34
NYSE, AMEX And NASDAQ Combined				
DSIX.DAT	8939	620702-971231	1-8939	4166
MSIX.DAT	865	251231-971231	1-865	403
QSIX.DAT	289	251231-971231	1-289	135
ASIX.DAT	73	251231-971231	1-73	34
NYSE And AMEX Risk-Class Portfolios				
DSBC.DAT	8939	620702-971231	1-8939	4166
DSSC.DAT	8939	620702-97331	1-8939	4166
Nasdaq Risk-Class Portfolios				
DSBO.DAT	8939	731214-971231	2610-8939	4166
DSSO.DAT	8939	731214-971231	2610-8939	4166
CRSP Cap-Based Portfolios				
MHISTN.DAT	865 x 17 = 14705	251231-971231	1-865	1353
MHISTA.DAT	865 x 17 = 14705	730731-971231	1-865	1353
MHISTQ.DAT	865 x 17 = 14705	820630-971231	1-865	1353
REBALN.DAT	288 x 10 = 2880	2603-9712	1-288	288
REBALA.DAT	288 x 10 = 2880	6209-9712	1-288	288
REBALQ.DAT	288 x 10 = 2880	8206-9712	1-288	288
CRSP Indices For The S&P 500® Universe				
DSP500.DAT	8939	620702-971231	1-8939	1162
MSP500.DAT	865	251231-971231	1-865	112
CRSP US Treasury and Inflation Series				
MCTI.DAT	865	251231-971231	1-865	249
QCTI.DAT	289	251231-971231	1-289	83
ACTI.DAT	73	251231-971231	1-73	21

INDEX

I

L

M

N

O

P

R

S

T

U

V

W

Y