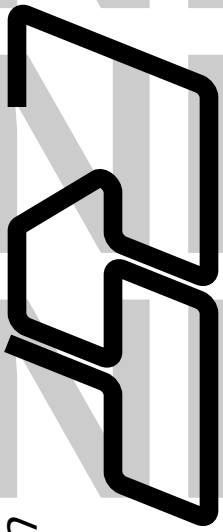
DAILY BOND

1997 FILE GUIDE



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

Center for Research in Security Prices

Graduate School of Business
The University of Chicago

CRSP Daily US Government Bond File Guide

Data Ending December 31, 1997

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 Addresses: 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

The documents and data are copyrighted materials of The University of Chicago, Graduate School of Business, Center for Research in Security Prices (CRSP) and its information providers. Reproduction or storage of materials retrieved from these are subject to the U.S. Copyright Act of 1976, Title 17 U.S.C.

The Center for Research in Security Prices, CRSP, CRSP Total Return Index Series, CRSPAccess97, CRSP Cap-Based Portfolio Series, PERMNO, PERMCO and CRSPID are additionally protected by registered trademark and other forms of proprietary rights. The Contents are owned or controlled by CRSP or the party credited as the provider of the Contents.

PROPRIETARY RIGHTS

PERMNO, PERMCO and CRSPID are symbols representing data, which is proprietary to The Center for Research in Security Prices.

CRSP DATA LICENSE

This is a legal agreement between you (either an individual or an entity) hereby referred to as the "user" and the University of Chicago, Graduate School of Business on behalf of the Center for Research in Security Prices, hereby referred to as "CRSP". By opening this product, you are agreeing to be bound by the terms of the following License Agreement. If you do not wish to accept these terms, return the unused, unopened data to CRSP within 30 days of receipt with any written materials to the Products and Services Department, CRSP, University of Chicago, GSB, 725 S. Wells Street, Suite 800, Chicago, IL 60607. *Opening this data without a signed agreement binds the user to restrictions of use in CRSP's standard subscription/contract terms for the product.*

The accompanying media contains data that are the property of CRSP, and its information providers and are licensed for use only, by you as the original licensee. Title to such media, data and documentation is expressly retained by CRSP.

Data and documentation are provided with restricted rights. CRSP grants the user the right to access the CRSP data and the CRSP Product File Guide(s) only for non-commercial, internal or academic research use. Usage of the data must be in accordance with the terms detailed in the Subscription Agreement, Contract or Agreement between CRSP and the user. The data is "in use" on a computer when it is loaded into the temporary memory (i.e. RAM) or is installed into the permanent memory (e.g. hard disk, CD ROM or any other storage device) of that computer or network in one location. If the data has been updated, Subscribers must promptly return the previous release of data on it's original medium to CRSP or destroy it.

Should you have any questions concerning this agreement, or if you desire to contact CRSP for any reason, please contact the Products and Services Office at CRSP, 725 S. Wells Street, Suite 800, Chicago, IL 60607 Telephone: 773.702.7467 Fax: 773.702.3036 e-mail: mail@crsp.uchicago.edu

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.

Copyright © 1998
Center for Research in Security Prices
University of Chicago

Version SDB-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 next page 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. 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

1.1 How to Use This Guide

Please read this documentation thoroughly before attempting to access the data

Document Organization

Introduction: outlines the content and development of the files and highlights recent changes to the files

Data Description: contains definitions and descriptions of all data items on the files.

Accessing the Data: contains technical information, including descriptions of sample FORTRAN and C programs that can be used to access the data and tables that display the data items vs. the respective program language (C and FORTRAN) variable usage.

Appendices: contain supplemental information

Index: contains an alphabetical listing of the variables in the data description, and the sample programs.

Notational Conventions

All data items and names that occur within FORTRAN or C programs are printed using a `constant – width (courier)` font. These names can be variable names, parameter names, subroutine names or keywords. For example, CUSIP refers to the CUSIP Agency identifier, while `CUSIP` refers to the variable that the programs use to store this identifier.

All names that refer to sample programs or include files are printed using an *italic Helvetica* `font`.

Names of FORTRAN common blocks are delimited by slashes `(/ /)`.

The text of this document is in Times New Roman, 10 point. *Italics* and **bold** styles are used to emphasize headings, names, definitions and related functions.

Accessing the Data

Section 2: Describes the data items available in the CRSP Daily US Government Bond Files and includes a diagram of Master and Cross-Sectional file structures.

Section 3: Describes the sample programs designed to read and process the data items. This section contains information on using the Section 2 data items in CRSP sample programs. It also contains brief implementation suggestions and complete record layouts.

Appendices

A. Special issues, stripped notes and bonds, and foreign targeted securities,

B. FORTRAN and C sample program, subprograms, and include file listings; and

C. Version specific technical information such as current parameter sizes.

1.2 Description of the CRSP US Government Bond Files

Development of CRSP US Government Bond Files

The CRSP US Government Bond Files were developed by the Center for Research in Security Prices at the Graduate School of Business, University of Chicago. The original monthly CRSP US Government Bond Master File was originally built by Lawrence Fisher, currently at Rutgers University, who originated the basic design and content of the Master Files. The monthly US Government Bond Master File tracks 5052 securities and contains over 93,000 price observations, beginning in December 1925. The Daily US Government Bond Master File tracks 3,176 securities and contains over 1.4 million price observations beginning June 14, 1961. The files provide a comprehensive machine-readable database of government security price information.

Development of the CRSP Daily US Government Bond File

The CRSP Daily US Government Bonds File contains over 1.4 million price observations for 3,176 securities beginning June 14, 1961. The prices were manually input through December 31, 1989. Beginning January, 1990 through September, 1996, the prices were obtained from the department of Commerce's electronic bulletin board (EBB). Beginning October, 1996 to the present, prices are supplied by GovPX.

The manually input prices were double-entered. Programs were written to compare the prices entered from both screens. Once compared, price corrections were double-entered; the corrections were also compared for consistency. Several iterations of this process took place to arrive at the final, "clean" version of the file. Logical filters were then written and run to further clean the data.

Descriptive information and amounts outstanding were shared with the CRSP Monthly US Government Bond File.

Description of CRSP Bond File Sources

Prices in the file prior to January of 1962 were obtained from a number of different sources (see description of SOURCR in Section 2.2). These sources include the *Wall Street Journal*, Salomon Brothers, Inc., and the Bank and Quotation Record.

Beginning with January of 1962, the majority of prices came from the Composite Closing Quotations for US Government Securities compiled by the Federal Reserve Bank of New York (FRBNY). In 1984, the quotation sheets were renamed the "Composite 3:30 P.M. Quotations for US Government Securities". The time at which the quotes were compiled was related to the fedwire deadline the FRBNY set for the transfer of securities. The deadline was set for 2:30 p.m. Eastern Time, but was regularly extended as much as three-quarters of an hour. The FRBNY trading desk began a "closing run" at 3:00 p.m. The reference to "closing quotations" from 1962 to 1984 probably refers to the "closing run" at the FRBNY. The close of the day on October 15th, 1996 the FRBNY discontinued publication of composite quotations.

The start of the day, October 16, 1996, our source for price quotations changed to GovPX, Inc (GovPX). GovPX receives its data from 5 inter-dealer bond brokers, who broker transactions among 37 primary dealers. Live, intra-day bids, offers and transactions in the active over-the-counter markets among these primary dealers are the source of GovPX's 5 p.m. End-Of-Day US Treasury prices. GovPX also began providing the following non-derived data: maturity date and coupon rates as of October 16, 1996. This data was formerly provided by the US Treasury Department.

The FRBNY described its listed bid price as "…the most widely quoted price from the range of quotations received". The ask price was determined by the FRBNY based on what they expect a typical bid-ask spread to be. The rule used to make this derivation was not public domain. GovPX describes its listed bid and ask prices as the "best price". To determine their "best price" they observe the prices from the 5 inter-dealer brokers and report the bid and ask prices that produce the smallest bid-ask spread.

The amount outstanding (TOTOUT) is obtained from the *Monthly Statement of the Public Debt of the United States published by the Treasury Department*. The amount publicly held (PUBOUT) is obtained from the quarterly US Treasury Bulletin. Money Rates are obtained from the Federal Reserve. The following non-derived data: issue date, coupon payable dates, bank eligibility, tax status and call status are obtained from the US Treasury Department.

Prior to 1990, CUSIP was obtained from Standard & Poor's CUSIP Directory. From January, 1990 through October 15th, 1996, CUSIP was obtained from the Composite 3:30 p.m. quotations for US Government Securities. GovPX, as of October 16, 1996, provides the CUSIP number. When in question, the CUSIP is verified by *Standard & Poor's CUSIP Directory*.

All data are checked for internal consistency with each release of the file. Secondary sources, such as the *Wall Street Journal*, are used to check suspect prices.

Differences Between Daily and Monthly Files

The CRSP Daily US Government Bond Files are a superset of the CRSP Monthly US Government Bond Files with three exceptions.

1. When-issued prices are included in the Daily Files. All prices before an issue's dated date can be identified as when-issued prices.

2. Government Certificate of Deposit, Commercial Paper, and Federal Funds rates are included in the daily files.

3. Bond indexes equivalent to the CRSP Monthly US Government Bond File Fama Files (4 total) have not yet been developed for the daily files.

The organization of the data has been changed significantly to reflect the increased amount of data. Certain derived data items are not stored, but can be accessed with utility functions that are provided. Other less frequent data are only stored on the observation dates. See Section 3 for information on accessing the daily data.

Accuracy of the US Government Bond Files

All data are checked for internal consistency, and secondary sources are used to check suspect prices.

Considerable resources are expended in checking and improving the quality of the 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 the differences in the data from update to update. The Annual CRSP US Government Bond Files contain updated data through the end of the previous calendar year. These updated files are available to subscribers each Spring.

1.3 Data Records Overview

The Daily US Government Bond Files are organized both as time series by issue and cross-sectionally by date.

Files based on organization by issue are referred to as Master Files (MBM) and include a file of raw monthly amount outstanding data and a file of interest payment dates and amounts.

Files based on organization by date are referred to as Cross-Sectional Files (MBX).

The sets of files are split into header information, raw daily data, and derived daily data. Header information contains CRSP identifiers, characteristics set by the US Treasury including interest dates and callable status and data ranges on quotes, number of amounts outstanding and number of interest payments.

See Section 2 for the available data items and their descriptions.

See Section 3 for specifics on data layouts and information on accessing the data.

1.4 Changes to the 1997 CRSP Daily US Government Bond File

- The CRSP US Government Bond Files are only available on CD this year. The CD has the volume label: `BDR1_199712`.
- The CRSP US Government Bond Files are in ASCII Character Format, SAS and Excel on CD. See Section 3 for details.
- The amounts for public debt and publicly held debt (`TOTOUT` and `PUBOUT`) have been significantly adjusted. The corrections are primarily between 1989 and the present.
- The US Treasury issued inflation indexed notes for the first time in 1997. The CRSP Bond Files contain data for these issues identified by a new `ITYPE` of 0. See Appendix A.4 for details.
- The Federal Reserve has created new categories of commercial paper rates with no history before 1997. CRSP may include these series in future releases of the CRSP Bond Files. Reports of current historical 30, 60 and 90 day Commercial Paper Rates were discontinued by the Federal Reserve after August 1997. The CRSP CP Rates File is set to 0 for these fields after that date.
- All new US Treasury Marketable Fixed-Rate Notes and Bonds issued on and after September 30, 1997 are eligible for STRIPS.

2. DATA DESCRIPTION

2.1 CRSP Daily US Government Bond File Structure

The diagrams below describe the structure of the CRSP Daily US Government Bond Files.

Figure 1 CRSP Daily US Government Bond File Structure

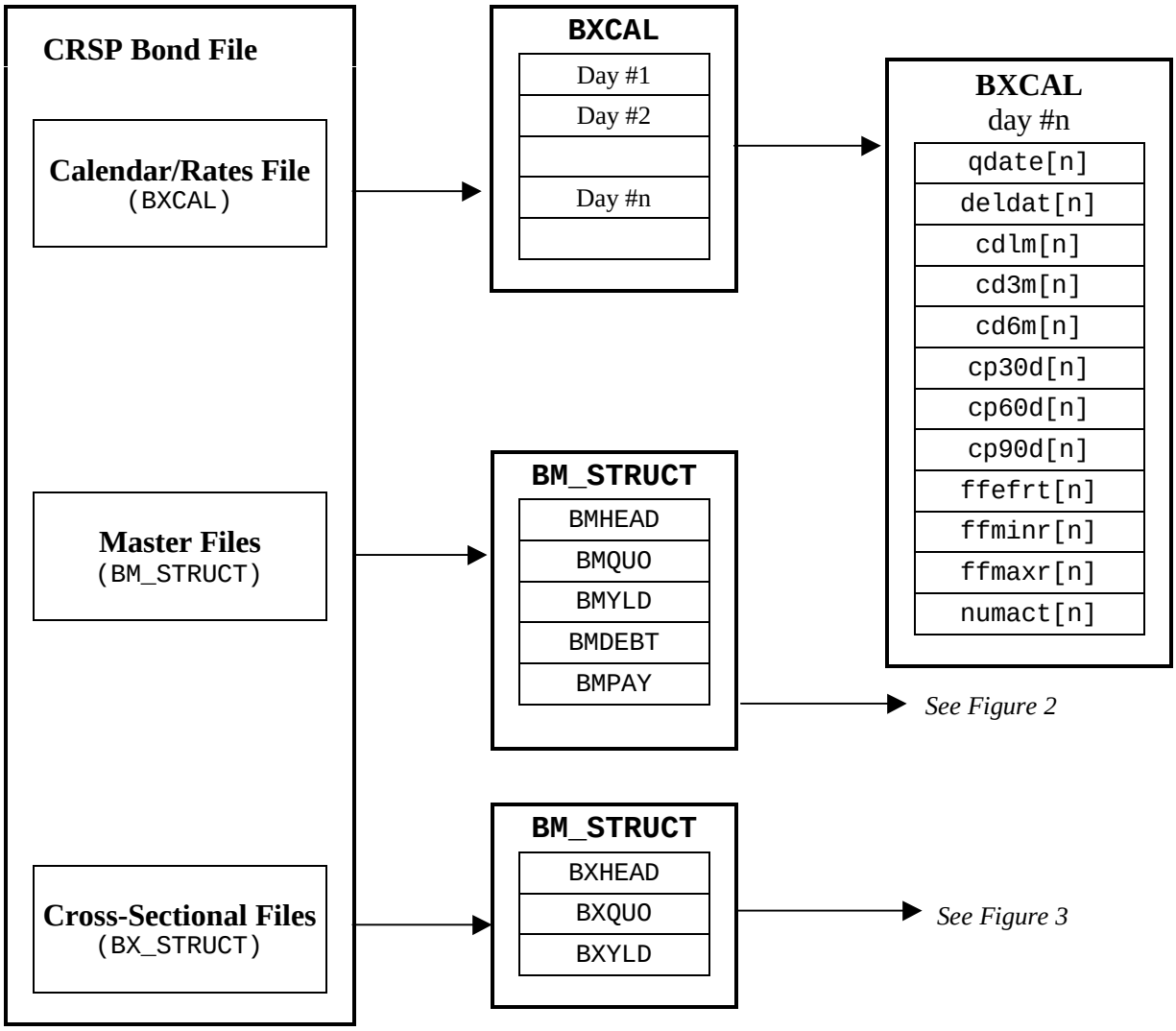


Figure 2 CRSP Daily US Government Bond Master File Structure

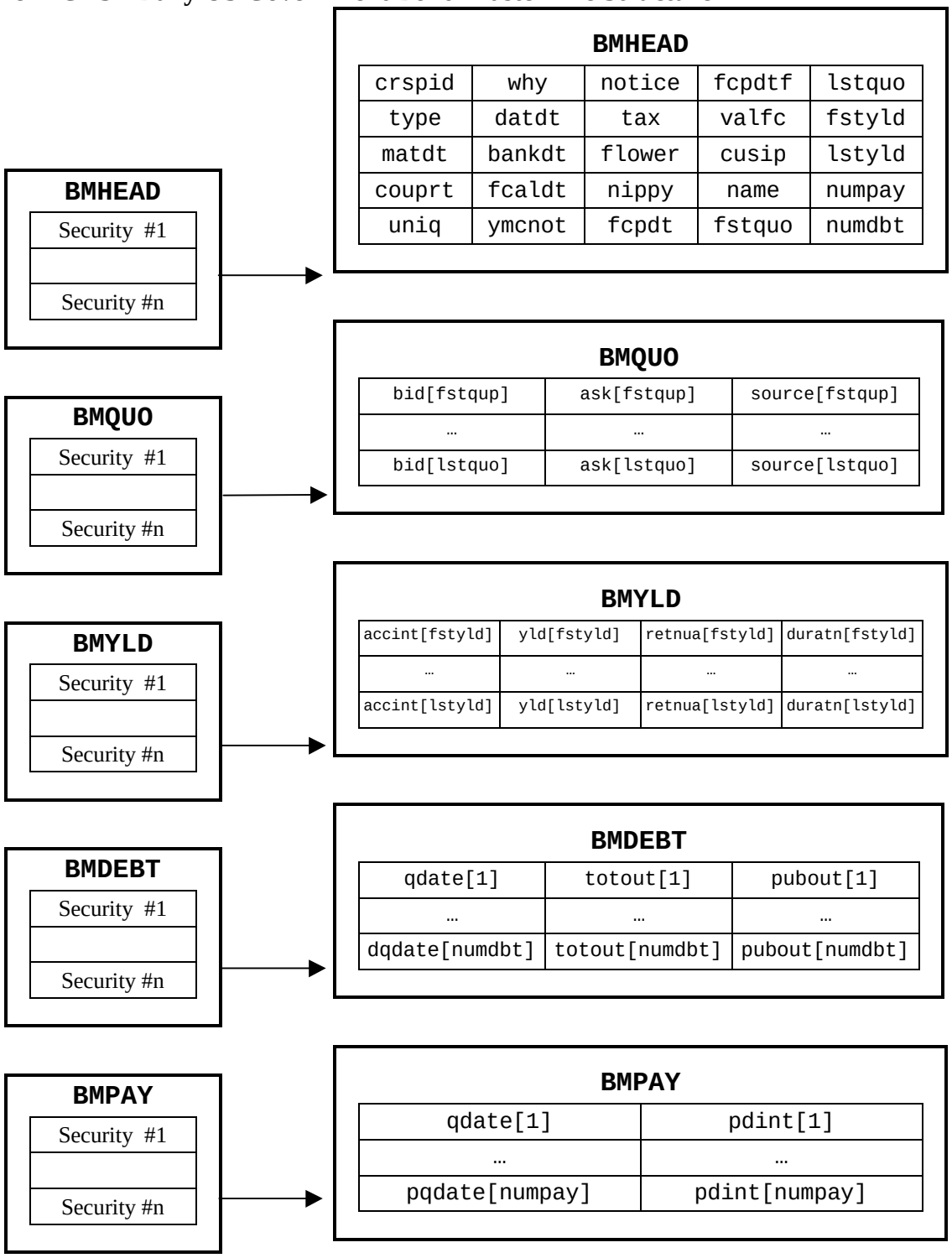


Figure 3 CRSP Daily US Government Bond Cross-Sectional File Structure

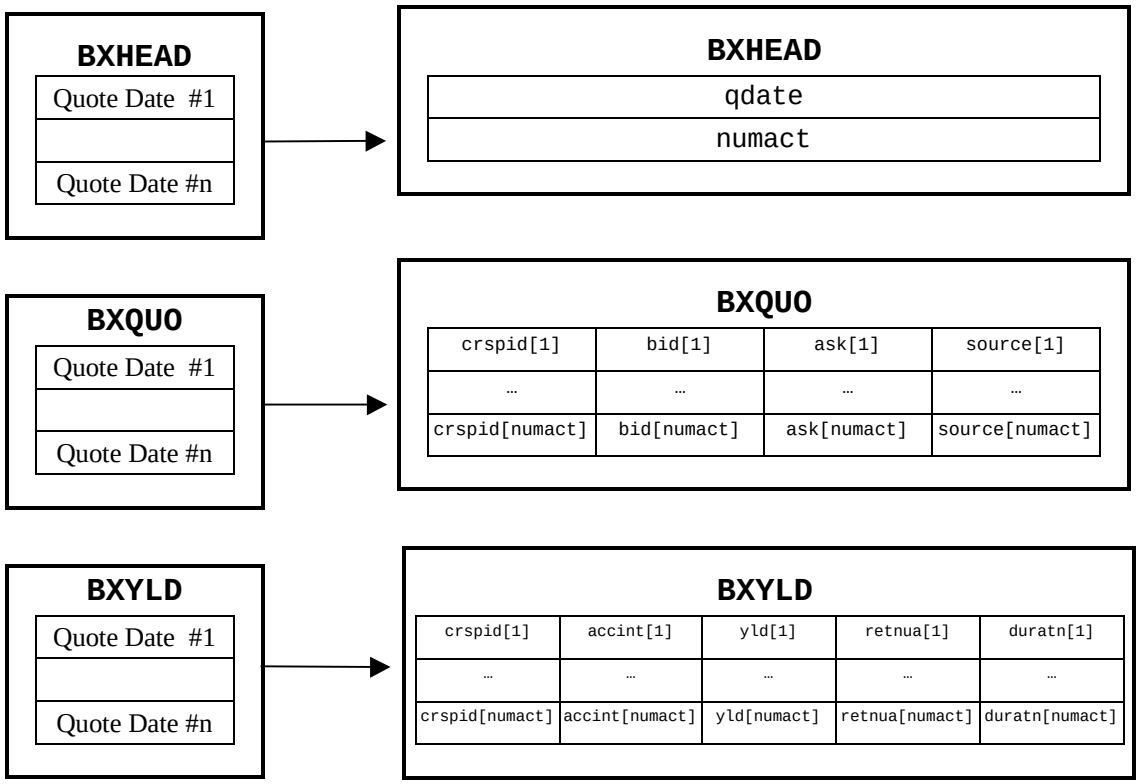


Figure 4 CRSP Fixed Term Indices File Layout

TERMTYPE[1]	QDATE[1]	CRSPID [1]	YEARSTM[1]	RETADJ[1]	YTM[1]	ACCINT[1]	DURATN[1]
_	_	_	_	_	_	_	_
TERMTYPE[1]	QDATE[N]	CRSPID [N]	YEARSTM[N]	RETADJ[N]	YTM[N]	ACCINT[1]	DURATN[N]
TERMTYPE[2]	QDATE[1]	CRSPID [1]	YEARSTM[1]	RETADJ[1]	YTM[1]	ACCINT[1]	DURATN[1]
_	_	_	_	_	_	_	_
TERMTYPE[N]	QDATE[N]	CRSPID [N]	YEARSTM[N]	RETADJ[N]	YTM[N]	ACCINT[N]	DURATN[N]

2.2 Variable Definitions

This section gives descriptions of the data items provided in the files. Each description is preceded with a line containing two items bolded:

1. The Variable Name
2. A Short Description of the Data Represented

The data items in this section are grouped logically according to six data types:

1. **CALENDAR** - Trading Calendar And Government Rates
2. **HEADER** - Issue Identification, Characteristics, And Data Ranges
3. **QUOTES** - Raw Pricing Data
4. **YIELDS** - Derived Yields, Duration, Returns, And Accrued Interest
5. **DEBT** - Amounts Outstanding
6. **PAYMENTS** - Interest Payments

Certain data types are available organized by issue and by date. The diagram in Section 2.1 graphically shows the organization in terms of Master and Cross-sectional files. More complete information on accessing the data items using variables in CRSP FORTRAN and C programs is contained in Section 3.

Information on the Fixed Term Indices File is available in Section 2.3.

CALENDAR - Calendar and Government Rates

The BXCAL structure contains the trading calendar and summary information for each date in the CRSP US Government Bond File. The three types of information include:

1. Trading calendar quote dates and delivery dates
2. Government rates for certificates of deposit, commercial paper, and federal funds
3. Counts of trading US Government securities

QDATE **Date of Quotation, in YYYYMMDD Format**

QDATE contains the trading quote dates for the Bond Files. These dates are stored in form YYYYMMDD (year, month, and date).

DELDAT **Delivery Date, in YYYYMMDD Format**

DELDAT contains the delivery date for a corresponding quote date. These dates are stored in the form YYYYMMDD (year, month, date).

The Federal Reserve Bank of New York the source from January 1962 through October 15, 1996, assumed cash transactions on delivery date. The delivery date usually fell two business days after the quotation date. GovPX, the source from October 16, 1996, reports delivery data the next business day after the end quote date.

CD1M **One-Month Certificate of Deposit Rate**

Certificate of deposit rate is the average of secondary market morning offering rates for time certificates of deposit of major money market banks. It is an unsecured note issued by companies for short-term borrowing purposes.

CD3M **Three-Month Certificate of Deposit Rate**

CD6M **Six-Month Certificate of Deposit Rate**

CP30D **30-Day Commercial Paper Rate**

Commercial paper rate is an average of posted 10 a.m. offering rates of five dealers. Rates are quoted on a discount basis. It is an unsecured note issued by companies for short-term borrowing purposes. Commercial paper is frequently sold by the issuer direct to the investor, the latter normally being institutions, viz. money-market fund, insurance companies, corporations, bank trust departments and pension funds. Commercial paper is also placed by intermediary banks or securities dealers.

CP60D **60-Day Commercial Paper Rate**

CP90D **90-Day Commercial Paper Rate**

FFEFRT **Federal Funds Effective Rate**

The effective rate is a weighted average of the rates on overnight Federal funds transactions arranged by federal funds brokers. It is the rate of interest charged on federal funds loaned by and to commercial banks. It is regarded by the Federal Reserve System regulator authorities as an important determinant of bank liquidity.

FFMINR **Federal Funds Minimum Trading Range**

FFMAXR **Federal Funds Maximum Trading Range**

NUMACT **Number of Active Issues**

The number of active US Government Bond issues that were quoted on a quotation date.

HEADER — Issue Identification, Characteristics, and Data Ranges

This structure contains header information for issues. There are three types of information included:

1. Identification assigned by CRSP or CUSIP to uniquely identify the issue.
2. Characteristics of the issue set by the treasury, such as interest dates and callable status.
3. Data ranges, including the date ranges of quotes, the number of amounts outstanding, and the number of interest payments.

CRSPID **CRSP Assigned Unique Issue Identification Number**

The CRSPID is in the format YYYYMMDD.TCCCCE, where:

YYYY	=	Maturity Year
MM	=	Maturity Month
DD	=	Maturity Day
T	=	Type Of Issue (TYPE)
CCCC	=	Integer Part of (COUPRT x 100)
E	=	Uniqueness Number (UNIQ)

For example, 19850515.504250 identifies a 41/4%callable bond which matures May 15, 1985. For callable notes and bonds, the YYYY portion of the CRSPID contains only the final maturity date of the issue and not the first eligible call date for that issue.

The variable CRSPID is a composite of other variables. Mathematical operations to retrieve parts of the CRSPID are unnecessary when using the Master File.

TYPE **Type of Issue**

0	Inflation Securities
1	Noncallable bond
2	Noncallable note
3	Certificate of indebtedness
4	Treasury Bill
5	Callable bond
6	Callable note
7	Tax Anticipation Certificate of Indebtedness
8	Tax Anticipation Bill
9	Other — this flags issues with unusual provisions. See Appendix A

MATDT **Maturity Date at Time of Issue, in YYYYMMDD Format**

COUPRT **Coupon Rate (percent per annum)**

UNIQ **Uniqueness Number**

Uniqueness number assigned to CRSPID if maturity date, coupon rate and type are not sufficient to distinguish between two securities; 0 otherwise.

WHY **Reason for End of Data on File**

0	Still quoted on last update of file.
1	Matured
2	Called for redemption
3	All exchanged
4	Sources no longer quote issue

DATDT **Date Dated by Treasury, in YYYYMMDD Format**

Coupon issues accrue interest beginning on the dated date. This may result in a modified first coupon payment if the dated date is not a regular interest payment date.

DATDT is 0 if it is not available or not applicable, as is the case with Treasury bills.

BANKDT **Bank Eligibility Date at Time of Issue, in YYYYMMDD Format.**

The earliest date at which a security is to become "bank eligible". A security is bank eligible if a bank may own it. Some 21/2%'s and 21/4%'s issued during and immediately after WWII limited negotiability because of prohibitions and restrictions on bank ownership.

0	no restrictions apply
YYYYMMDD	restrictions removed or scheduled to have been removed on this date

All remaining restrictions were removed on January 1, 1955. The last bank eligible CRSPID in the file is dated November 15, 1945 and matured on December 15, 1972.

FCALDT **First Eligible Call Date at Time of Issue, in YYYYMMDD Format.**

FCALDT is 0 if the security is not callable. All interest payment dates beginning with the first eligible call date are possible future call dates.

YMCNOT **Year and Month of First Call Notice, in YYYYMMDD Format**

YMCNOT is 0 if not called or not callable.

NOTICE **Notice Required on Callable Issues**

TAX **Taxability of Interest**

1 Fully taxable for federal income tax purposes.
2 Partially tax exempt, i.e. interest of first $3000 of bonds of this class, at par value, exempt from tax subject to surtax but not to normal tax.
3 Wholly tax exempt.

FLOWER **Payment of Estate Tax Code.**

1 No special status
2 Acceptable at par and accrued interest if owned by decedent at time of death; a flower bond
3 Acceptable at par and accrued interest if owned by decedent during entire 6 month period preceding death; a flower bond

NIPPY **Number of Interest Payments Per Year**

0 Treasury bill or certificate paying interest only at maturity
1 Annual interest
2 Semi-annual interest
3 Quarterly interest

All interest-bearing negotiable Treasury securities issued since the beginning of WWI have paid interest semi-annually. The last outstanding issue that paid interest quarterly was the Panama Canal Loan 3%'s due June 1, 1961.

FCPDT **First Coupon Payment Date, in YYYYMMDD Format**

FCPDT is 0 if not applicable. FCPDTF indicates whether the first coupon date is an estimate or a verified date.

FCPDTF **First Coupon Payment Date Flag**

0	Treasury bill or not applicable
-1	First coupon date is estimated from the normal coupon payment cycle
1	First coupon date has been verified on the Treasury Offering Circular

VALFC **Amount of First Coupon Per $100 Face Value**

CUSIP **CUSIP Number**

A CUSIP number (Committee on Uniform Securities Identification Procedures) is an identifying number assigned to a publicly-traded security. A nine-digit code is permanently assigned to each issue and is generally printed on the face of the security if it is in physical form. The first eight digits are included in the CRSP file. The ninth digit is a check digit derived from the first eight digits. Missing CUSIPs are assigned the value OXX. The earliest maturity on the file with a CUSIP is February 15, 1969.

NAME **Name of Government Security**

Name	ITYPE	Explanation
BILL	4	
T_A_BILL	8	Tax Anticipation
T_A_CTF	7	Tax Anticipation
BOND	1,5,9	
CNV_BOND	1	Convertible
CONSOL	9	Consol
CTF	3,7,9	Certificate of Deposit
NOTE	0,2,6,9	
1LL_BOND	5	First Liberty Loan
1LL_CV	5	1LL First Conversion
1LL_2CNV	5	1LL Second Conversion
2LL_BOND	5	Second Liberty Loan
2LL_CNV	5	2LL First Loan Conversion
3LL_BOND	1	Third Liberty
4LL_BOND	9	Fourth Liberty Loan
4LL_CALL	9	Fourth Liberty Loan called
PCL_BOND	1,5	Panama Canal Loan

FSTQUO **Day Number of Issue's First Quote on File**

The QDATE array can be used to translate day numbers into YYYYMMDD format dates.

LSTQUO **Day Number of Issue's Last Quote**

The QDATE array can be used to translate day numbers into YYYYMMDD format dates. An issue that matures typically stops trading on the first quote date with a delivery date greater than or equal to the issue's maturity date.

FSTYLD **Day Number of Issue's First Yield**

The QDATE array can be used to translate day numbers into YYYYMMDD format dates.

LSTYLD **Day Number of Issue's Last Yield**

The QDATE array can be used to translate day numbers into YYYYMMDD format dates. An issue that matures typically stops trading on the first quote date with a delivery date greater than or equal to the issue's maturity date.

NUMPAY **Number of Interest Payments**

Count of observations in BMPAY structure.

NUMDBT **Number of Amount Outstanding Observations**

Count of valid observations in the BMDEBT structure.

QUOTES — Raw Data

CRSP generated data such as yield and duration are calculated from secondary market cash transaction prices. CRSP derives its data from the bid and ask prices. CRSP data are calculated based on cash transactions on the quotation date. CRSP's primary data sources assume cash transactions on delivery date. Quotes from the Federal Reserve Bank of New York usually have a delivery date two business days after the quotation date. Quotes from GovPX usually have a delivery date one business day after the quotation date. The delivery date usually falls two business days after the quotation date. CRSP takes this into account when verifying the internal consistency of the files.

When-issued prices are included in the file when quoted. Any price with a quote date before an issues' dated date is classified when-issued.

Quotes are present in Master file and Cross-Sectional versions of the file. In the Master file , the quotes are sorted by issue, then date. For any issue, header variables FSTQUO and LSTQUO can be used to delimit the day numbers of the range. In the Cross-Sectional file, the quotes are sorted by date, then issue. For any quote date, calendar variable NUMACT contains the number of quotes available.

BID & ASK **Prices**

The bid price is the price at which a buyer is willing to purchase a security. The ask price is the price at which the seller is offering to sell the security.

Arrays BID and ASK contain day-end bid and ask information, when available for each quote date prior to maturity. If BID and ASK are not available, whatever quote information is available is used and coded using the following conventions:

Information in Data Source	BID	ASK
Bid and Ask	Bid	Ask
Mean of Bid and Ask	Mean	Mean
Bid only	Bid	-Bid
Ask only	-Ask	Ask
Sale (last trading price)	Sale	0
No price Sale	0	0

SOURCR **Primary Data Source**

R Federal Reserve Bank of New York
S Salomon Brothers
W Wall Street Journal (Associated Press: 6/14/61-8/20/87, Bloomberg: 8/28/87-7/2/90, Bear-Stearns: 12/4/90-present)
M No quote was available
X GovPX, Inc.

YIELDS — Derived Data

For callable bonds that have been called, or are likely to be called, the original maturity date is no longer valid for computing duration and yield. In these cases the anticipated call date is used as the working maturity date.

The following note applies to the variables promised daily yield (YIELD) and duration (DURATN).

Status	Yield and Duration Computed to
Called	Next call date
Callable and priced at a premium	Next call date
Callable and priced at a discount	Maturity date
Not callable	Maturity date

Users should be cautious in interpreting yields based on issues close to maturity. Quotes on these instruments are not always reliable due to infrequent trading.

Yields are present in Master file and Cross-Sectional versions of the file. In the Master file, the yields are sorted by issue, then date. For any issue, header variables FSTYLD and LSTYLD can be used to delimit the day numbers of the range. In the Cross-Sectional file, the yields are sorted by date, then issue. For any quote date, calendar variable NUMACT contains the number of yields available.

ACCINT **Total Accrued Interest At End of Day**

Accrued interest on U.S. Treasury marketable securities is calculated on the basis of the number of days between interest payment dates for a $100 bond or note. Interest is accrued either from the last interest payment date or the dated date (when an interest payment has not yet occurred) to the quotation date.

YIELD **Promised Daily Yield**

YIELD is the promised yield daily rate, also called daily yield to maturity.

At any date, the promised yield of a security is the single interest or discount rate which makes the sum of the present values of the principle at maturity and future interest payments be precisely equal to the flat price of the security. The flat price is the nominal price, e.g., mean of BID and ASK, plus the accrued interest on the date in question. If a price is missing, the YIELD for that month is set to -99.

RETNUA **Unadjusted Return**

RETNUA is price change plus interest, divided by last day's price. It is set to a large negative number for days in which a return cannot be calculated, i.e. if the price is missing for either this day or last day. Missing returns are set to -99.

$$RETNUA = \frac{XNUM}{XDEN}, \text{ where}$$

When BID and ASK available:

$$XDEN = \frac{BID\ (I-1) + ASK\ (I-1)}{2} + ACCINT\ (I-1)$$

$$XNUM = \frac{BID\ (I) + ASK\ (I)}{2} - \frac{BID\ (I-1) + ASK\ (I-1)}{2} + YINT$$

$$YINT = PDINT\ (I) + ACCINT\ (I) - ACCINT\ (I-1)$$

For all other cases:

```
XNUM = BID(I) - BID(I-1) + YINT
XDEN = BID(I-1) + ACCINT(I-1)
YINT = PDINT(I) + ACCINT(I) - ACCINT(I-1)
```

DURATN **Duration (Macaulay's Duration)**

Duration is the weighted average number of days until the cash flows occur, where the present values, discounted by yield to maturity, of each payment are used as the weights[1]. Also known as Macaulay's Duration.

If, P_{t_0}, P_{t_2}, …, P_{t_n} are the present values at time t_0 of payment promised at perhaps unequally spaced time intervals $t_1, t_2, ..., t_n$ then the duration of that promised stream measured at t_0 is:[2]

$$D_{t_0} = \frac{\sum_{j=1}^{j=n}(t_j - t_0)P_{t_j}}{\sum_{j=1}^{j=n}P_{t_j}} = \frac{\sum_{j=1}^{j=n}t_j P_{t_j}}{\sum_{j=1}^{j=n}P_{t_j}} - t_0$$

[1] *Some Theoretical Problems of Interest Rates, Bond Yields and Stock Prices in the United States Since 1856.* Frederick R. MacAulay, National Bureau of Economic Research, 1938, 44-53.

2 Coping with the Risk of Interest-Rate Fluctuations: Returns to Bondholders from Naive and Optimal Strategies, Lawrence Fisher and Roman L. Weil, Journal of Business, vol. 44, 415.

DEBT — Amounts Outstanding

Amounts outstanding are present in the Master file, sorted by issue and date. The header variable NUMDBT contains the number of records available for an issue. These values are typically reported monthly. Total amounts outstanding are obtained from the *Monthly Statement of the Public Debt of the United States*. The amounts publicly held are obtained from the quarterly *Treasury Bulletin*. The *Treasury Bulletin* was reported monthly before 1983.

DQDATE **Effective Date of Amount Outstanding Values in YYYYMMDD Format**

TOTOUT **Face Value Outstanding**

Amount (face value) issued and still outstanding in millions of dollars. Set to 0 for unknown values up to December 31, 1961 and set to -1 for unavailable values after December 31, 1961.

PUBOUT **Publicly Held Face Value Outstanding**

Amount (face value) held by the public in millions of dollars. This is the total amount outstanding (TOTOUT) minus the amount held in U.S. Government accounts and Federal Reserve Banks. This amount is not available for Treasury Bills and is always set to 0. For other issues, set to 0 for unknown values up to December 31, 1961 and set to -1 for unavailable values after December 31, 1961. After December 31, 1982, these numbers are reported quarterly instead of monthly and the reported values are carried forward for the next two months.

PAYMENTS — Interest Payments

Payments are present in the Master file, sorted by issue and date. The values are derived from the frequency and amount of coupon payments, the first coupon date, value of first coupon, and maturity date. Payments are only stored for the time range of an issue's quotes. Bills have no payment records.

PQDATE **Interest Payment Dates, in YYYYMMDD Format**

PDINT **Interest Paid**

PDINT is the coupon payable on the interest payment date.

2.3 CRSP Fixed Term Indices Files

The CRSP Daily US Government Bond Fixed Term Indices Files contain 1, 2, 5, 7, 10, 20 and 30 year Fixed Term Indices. These issues are sorted by termtype, which distinguishes the length of maturity. A valid issue that best represents each term is chosen at the end of each month for each of the above referenced fixed terms. A valid issue is one that is at least one half year prior to the target maturity date and is fully taxable. The selection process filters a representative bond from each of the fixed term groups. The first selection criteria are; a non-callable, non-flower bond that is closest to the target maturity of its group and fully taxable. If more than one issue remains, and/or none are available which fit the above criteria, they are then respectively filtered on the basis of; flower bonds acceptable at par, and accrued interest if owned by descendent at time of death.

These values were designed to plot a sophisticated yield curve and the user may reference the yields with returns, prices and durations.

The Fixed Term Indices Daily Files begin June 14, 1961. The specific maturities are as follows:

Termtype	Index
3012	30 year
2012	20 year
1012	10 year
712	7 year
512	5 year
212	2 year
112	1 year

Indices Variable Items

ACCINT **Total Accrued Interest At End of Day**

Accrued interest on U.S. Treasury marketable securities is calculated on the basis of the number of days between interest payment dates for a $100 bond or note. Interest is accrued either from the last interest payment date or the dated date (when an interest payment has not yet occurred) to the quotation date.

BID & ASK **Prices**

The bid price is the price at which a buyer is willing to purchase a security. The ask price is the price at which the seller is offering to sell the security.

Arrays BID and ASK contain day-end bid and ask information when available for each quote date prior to maturity.

Information in Data Source	BID	ASK
No price	0	0
Sale	Sale	0
Bid only	Bid	-Bid
Ask only	-Ask	Ask
Bid and Ask	Bid	Ask
Mean of Bid and Ask	Mean	Mean

CRSPID **CRSP Assigned Unique Issue Identification Number**

The CRSPID is in the format YYYYMMDD.TCCCCE, where:

YYYY	=	Maturity Year
MM	=	Maturity Month
DD	=	Maturity Day
T	=	Type Of Issue (TYPE)
CCCC	=	Integer Part of (COUPRT x 100)
E	=	Uniqueness Number (UNIQ)

For example, 19850515.504250 identifies a 41/4%callable bond which matures May 15, 1985. For callable notes and bonds, the YYYY portion of the CRSPID contains only the final maturity date of the issue and not the first eligible call date for that issue.

DURATN **Duration (Macaulay's Duration)**

Duration is the weighted average number of days until the cash flows occur, where the present values, discounted by yield to maturity, of each payment are used as the weights[3]. Also known as Macaulay's Duration.

If P_{t_0}, P_{t2},… , P_{t_n} are the present values at time t_0 of payment promised at perhaps unequally spaced time intervals t_1, t_2, ..., t_n then the duration of that promised stream measured at t_0 is:[4]

$$D_{t_0} = \frac{\sum_{j=1}^{j=n}(t_j - t_0)P_{t_j}}{\sum_{j=1}^{j=n}P_{t_j}} = \frac{\sum_{j=1}^{j=n}t_j P_{t_j}}{\sum_{j=1}^{j=n}P_{t_j}} - t_0$$

QDATE **Date of Quotation, in YYYYMMDD Format**

QDATE contains the Trading Quote Dates for the Bond Files. These dates are stored in the form YYYYMMDD (year, month, and date).

RETADJ **One Month Holding Period Return**

RETADJ is the one month holding period return expressed as a percentage .

RETADJ(I)=100**RETNUA(I)*

TERMTYPE **Index Identification Number**

Fixed term index identification number links all results in the fixed term indices file. The identification is typically in the form YYMM, where YY is the number of years to maturity of issues selected in the index and MM is the number of months an issue is held once selected before another is chosen.

YEARSTM **Years to Maturity**

Number of years left to maturity. In the fixed term index files, YEARSTM contains the time left to maturity of the selected issue as of the quote date, expressed annually as a decimal amount.

YTM **Annualized Yield**

YTM is the annualized YIELD to maturity expressed as a percent per annum. See YIELDS: YIELD.

YTM(I)=100**[YLD(I)**365*]*

[3] *Some Theoretical Problems of Interest Rates, Bond Yields and Stock Prices in the United States Since 1856.* Frederick R. Macaulay, National Bureau of Economic Research, 1938, 44-53.

4 Coping with the Risk of Interest-Rate Fluctuations: Returns to Bondholders from Naive and Optimal Strategies, Lawrence Fisher and Roman L. Weil, Journal of Business, vol. 44, 415.

3. ACCESSING THE DATA

This section provides general information needed to access the CRSP Daily US Government Bond Files. The data files are available in three formats: ASCII, Excel, and SAS.

1. The ASCII files, closely structured to the format formerly provided on the CD, work with the included C and FORTRAN sample programs and subroutines and can be used to load into various other programs. These files were used to create the Excel and SAS files. See Section 3.2 for details about the ASCII file specifications for the Master Bond (`MBM`) File, the associated Header File, Cross Sectional (`MBX`) File and the Fixed Term Indices File. Section 3.1 contains descriptions of the sample programs and subroutines.

2. The Excel 5.0/95 Workbook files may contain multiple worksheets per file. The large master and cross sectional files were not converted into Excel because of their size. See Section 3.2 for details about the Excel file and work sheet layout.

3. The SAS files contain the entire Master File. They were combined and are distributed in one large transport file created in SAS PROC CPORT, to support SAS's many different platforms and data engines. Sample SAS code is provided to create Cross Sectional Files from the Master Files. See Section 3.2 for detail on the SAS File layout.

CD-ROM Layout

The top level of the CD contains the directory containing the ASCII character data (data), the documentation (doc), data converted into MS Excel 5.0/95 (excel), data converted into SAS (sas), source code containing uncompiled FORTRAN (forsrc) and C (src) sample programs and subroutines, and two text files; a copy of the accompanying CRSP Data License (`data_lic.txt`) and a copy of CRSP's Copyright Notice (`copyright.txt`).

The `BXDLYIND` ASCII character file in the data directory contains multiple series. In this cases, the combined file is stored in the top level data directory and a subdirectory (`data\bxdlyind\`) exists with the individual series.

There are two subdirectories in the `\doc\` directory. `\doc\word\` subdirectory contains the documentation in Microsoft Word 97 (`.doc`) and `\doc\adobe\` subdirectory contains the documentation in Acrobat Adobe (`.pdf`) format.

3.1 Description of Programs

CRSP has provided both FORTRAN and C subroutines and sample programs that can be used to access the bond data in Master or Cross-Sectional File format. The FORTRAN programs can read sequentially the character files provided and C programs can read sequentially or randomly the character files provided. In addition, there are C programs that can convert the data files to binary and C and FORTRAN programs that can read sequentially or randomly the binary files created.

The following table shows how data items can be accessed in the FORTRAN programs for Master or Cross-Sectional files. The table is ordered by data item names as described in Section 2. Usage shows whether the data item is being accessed in Master or Cross-Sectional Files. The calendar is available in both groups of files. Common block names are not used when directly accessing a variable in a program.

Table 1 Data Items vs. FORTRAN Variable Usage

Group	Data Item Name	FORTRAN Data Type	Usage	FORTRAN variable with Common Block	Index I Between
CALENDAR	QDATE	INTEGER	Calendar	/BXCAL/QDATE[I]	1 and /BXCAL/NQDAT
	"	"	Cross-Sectional	/BXCAL/XQDATE	n/a
	DELDAT	INTEGER	Calendar	/BXCAL/DELDAT[I]	1 and /BXCAL/NQDAT
	CD1M	REAL	Calendar	/BXCAL/CD1M[I]	1 and /BXCAL/NQDAT
	CD3M	REAL	Calendar	/BXCAL/CDM3M[I]	1 and /BXCAL/NQDAT
	CD6M	REAL	Calendar	/BXCAL/CD6M[I]	1 and /BXCAL/NQDAT
	CP30D	REAL	Calendar	/BXCAL/CP30D[I]	1 and /BXCAL/NQDAT
	CP60D	REAL	Calendar	/BXCAL/CP60D[I]	1 and /BXCAL/NQDAT
	CP90D	REAL	Calendar	/BXCAL/CP90D[I]	1 and /BXCAL/NQDAT
	FFEFRT	REAL	Calendar	/BXCAL/FFEFRT[I]	1 and /BXCAL/NQDAT
	FFMINR	REAL	Calendar	/BXCAL/FFMINR[I]	1 and /BXCAL/NQDAT
	FFMAXR	REAL	Calendar	/BXCAL/FFMAXR[I]	1 and /BXCAL/NQDAT
	NUMACT	INTEGER	Calendar	/BXCAL/NUMACT [I]	1 and /BXCAL/NQDAT
	"	"	Cross-Sectional	/BXHEAD/XNUM	n/a
HEADER	CRSPID	CHARACTER*15	Master	/BMHEAD/CRSPID	n/a
	"	"	Cross-Sectional	/BMHEAD/CRSPID[I]	1 and /BXHEAD/XNUM
	TYPE	INTEGER	Master	/BMHEAD/TYPE	n/a
	MATDT	REAL*8	Master	/BMHEAD/MATDT	n/a
	COUPRT	INTEGER	Master	/BMHEAD/COUPRT	n/a
	UNIQ	INTEGER	Master	/BMHEAD/UNIQ	n/a
	WHY	INTEGER	Master	/BMHEAD/WHY	n/a
	DATDT	INTEGER	Master	/BMHEAD/DATDT	n/a
	BANKDT	INTEGER	Master	/BMHEAD/BANKDT	n/a
	FCALDT	INTEGER	Master	/BMHEAD/FCALDT	n/a
	YMCNOT	INTEGER	Master	/BMHEAD/YMCNOT	n/a
	NOTICE	INTEGER	Master	/BMHEAD/NOTICE	n/a
	TAX	INTEGER	Master	/BMHEAD/TAX	n/a
	FLOWER	INTEGER	Master	/BMHEAD/FLOWER	n/a
	FCPDT	INTEGER	Master	/BMHEAD/FCPDT	n/a
	FCPDTF	INTEGER	Master	/BMHEAD/FCPDTF	n/a
	VALFC	REAL*8	Master	/BMHEAD/VALFC	n/a
	CUSIP	CHARACTER*8	Master	/BMHEAD/CUSIP	n/a
	NAME	CHARACTER*8	Master	/BMHEAD/NAME	n/a
	FSTQUO	INTEGER	Master	/BMHEAD/FSTQUO	n/a
	LSTQUO	INTEGER	Master	/BMHEAD/LSTQUO	n/a
	FSTYLD	INTEGER	Master	/BMHEAD/FSTYLD	n/a
	LSTYLD	INTEGER	Master	/BMHEAD/LSTYLD	n/a
	NUMPAY	INTEGER	Master	/BMHEAD/NUMPAY	n/a
	NUMDBT	INTEGER	Master	/BMHEAD/NUMDBT	n/a

Table 1 Data Items vs. FORTRAN Variable Usage *(Con't)*

Group	Data Item Name	FORTRAN Data Type	Usage	FORTRAN Variable with Common Block	Index I Between
QUOTES	BID	REAL*8	Master	/BMQUO/BID[I]	/BMHEAD/FSTQUO and
	"	"	Cross-Sectional	/BXQUO/BID[I]	/BMHEAD/LSTQUO1 and /BXHEAD/XNUM
	ASK	REAL*8	Master	/BMQUO/ASK[I]	/BMHEAD/FSTQUO and /BMHEAD/LSTQUO
	"	"	Cross-Sectional	/BXQUO/ASK[I]	/BXQUO/ASK[I] 1 and /BXHEAD/XNUM
	SOURCE	CHARACTER*1	Master	/BMQUO/SOURCE[I]	/BMHEAD/FSTQUO and /BMHEAD/LSTQUO
	"	"	Cross-Sectional	/BXQUO/SOURCE[I]	1 and BXHEAD/XNUM
YIELDS	ACCINT	REAL*8	Master	/BMYLD/ACCINT[I]	/BMHEAD/FSTYLD and /BMHEAD/LSTYLD
	"	"	Cross-Sectional	/BXYLD/ACCINT[I]	1 and /BXHEAD/XNUM
	YLD	REAL*8	Master	/BMYLD/YLD[I]	/BMHEAD/FSTYLD and /BMHEAD/LSTYLD
	"	"	Cross-Sectional	/BXYLD/YLD[I]	1 and /BXHEAD/XNUM
	RETNUA	REAL*8	Master	/BMYLD/RETNUA[I]	/BMHEAD/FSTYLD and /BMHEAD/LSTYLD
	"	"	Cross-Sectional	/BXYLD/RETNUA[I]	1 and /BXHEAD/XNUM
	DURATN	REAL*8	Master	/BMYLD/DURATN[I]	/BMHEAD/FSTYLD and /BMHEAD/LSTYLD
	"	"	Cross-Sectional	/BXYLD/DURATN[I]	1 and /BXHEAD/XNUM
DEBT	DQDATE	INTEGER	Master	/BMDEBT/DQDATE[I]	1 and /BMHEAD/NUMDBT
	TOTOUT	INTEGER	Master	/BMDEBT/TOTOUT[I]	1 and /BMHEAD/NUMDBT
	PUBOUT	INTEGER	Master	/BMDEBT/PUBOUT[I]	1 and /BMHEAD/NUMDBT
PAYMENTS	PQDATE	INTEGER	Master	/BMPAY/PQDATE[I]	1 and /BMHEAD/NUMPAY
	PDINT	REAL*8	Master	/BMPAY/PDINT[I]	1 and /BMHEAD/NUMPAY

The following table shows how data items can be accessed in the C programs for Master or Cross-Sectional files. The table is ordered by data item names as described in Section 2. Usage shows whether the data items is being accessed in Master or Cross-Sectional Files. The calendar is available in both groups of files.

Table 2 Data Items vs. C Variable Usage

Group	Data Item Name	C Data Type	Usage	C Variable with Structure	Index i Between
CALENDAR	QDATE	int	Calander	bxcal.qdat [i]	1 and nbx_cal
	"	"	Cross-Sectional	bx_struct.bxhead.qdate	n/a
	DELDAT	int	Calander	bxcal.deldat [i]	1 and nbx_cal
	CD1M	float	Calander	bxcal.cd1m [i]	1 and nbx_cal
	CD3M	float	Calander	bxcal.cdm3m [i]	1 and nbx_cal
	CD6M	float	Calander	bxcal.cd6m [i]	1 and nbx_cal
	CP30D	float	Calander	bxcal.cp30d [i]	1 and nbx_cal
	CP60D	float	Calander	bxcal.cp60d [i]	1 and nbx_cal
	CP90D	float	Calander	bxcal.cp90d [i]	1 and nbx_cal
	FFERT	float	Calander	bxcal.ffefrt [i]	1 and nbx_cal
	FFMINR	float	Calander	bxcal.ffminr [i]	1 and nbx_cal
	FFMAXR	float	Calander	bxcal.ffmaxr [i]	1 and nbx_cal
	NUMACT	int	Calander	bxcal.numact [i]	1 and nbx_cal
	"	"	Cross-Sectional	bx_struct.bxhead.numact	n/a
HEADER	CRSPID	Char[16]	Master	bm_struct.bmhead.crspid	n/a
	"	"	Cross-Sectional	bm_struct.bmquo.crspid [i]	0 and <bx_struct.bxhead.numact
	"	"	Cross-Sectional	bm_struct.bmyld.crspid [i]	0 and <bx_struct.bxhead.numact
	TYPE	int	Master	bm_struct.bmhead.type	n/a
	MATDT	int	Master	bm_struct.bmhead.matdt	n/a
	COUPRT	double	Master	bm_struct.bmhead.couprt	n/a
	UNIQ	int	Master	bm_struct.bmhead.uniq	n/a
	WHY	int	Master	bm_struct.bmhead.why	n/a
	DATDT	int	Master	bm_struct.bmhead.datdt	n/a
	BANKDT	int	Master	bm_struct.bmhead.bankdt	n/a
	FCALDT	int	Master	bm_struct.bmhead.fcaldt	n/a
	YMCNOT	int	Master	bm_struct.bmhead.ymcnot	n/a
	NOTICE	int	Master	bm_struct.bmhead.notice	n/a
	TAX	int	Master	bm_struct.bmhead.tax	n/a
	FLOWER	int	Master	bm_struct.bmhead.flower	n/a
	FCPDT	int	Master	bm_struct.bmhead.fcpdt	n/a
	FCPDTF	int	Master	bm_struct.bmhead.fcpdtf	n/a
	VALFC	double	Master	bm_struct.bmhead.valfc	n/a
	CUSIP	char[9]	Master	bm_struct.bmhead.cusip	n/a
	NAME	char[9]	Master	bm_struct.bmhead.name	n/a
	FSTQUO	int	Master	bm_struct.bmhead.fstquo	n/a
	LSTQUO	int	Master	bm_struct.bmhead.lstquo	n/a
	FSTYLD	int	Master	bm_struct.bmhead.fstyld	n/a
	LSTYLD	int	Master	bm_struct.bmhead.lstyld	n/a
	NUMPAY	int	Master	bm_struct.bmhead.numpay	n/a
	NUMDBT	int	Master	bm_struct.bmhead.numdbt	n/a
QUOTES	BID	double	Master	bm_struct.bmquo.bid[i]	bm_struct.bmhead.fstquo and bm_struct.bmhead.lstquo
	"	"	Cross-Sectional	bx_struct.bxquo.bid [i]	0 and <bx_struct.bxhead.numact
	ASK	double	Master	bm_struct.bmquo.ask [i]	bm_struct.bmhead.fstquo and bm_struct.bmhead.lstquo
	"	"	Cross-Sectional	bx_struct.bxquo.ask[i]	0 and <bx_struct.bxhead.numact
	SOURCE	char	Master	bm_struct.bmquo.source [i]	bm_struct.bmhead.fstquo and bm_struct.bmhead.lstquo
	"	"	Cross-Sectional	bx_struct.bxquo.source [i]	0 and <bx_struct.bxhead.numact
YIELDS	ACCINT	double	Master	bm_struct.bmyld.accint [i]	bm_struct.bmhead.fstyld and bm_struct.bmhead.lstyld
	"	"	Cross-Sectional	bx_struct.bxyld.accint [i]	0 and <bx_struct.bxhead.numact
	YLD	double	Master	bm_struct.bmyld.yld [i]	bm_struct.bmhead.fstyld and bm_struct.bmhead.lstyld
	"	"	Cross-Sectional	bx_struct.bxyld.yld[i]	0 and <bx_struct.bxhead.numact
	RETNUA	double	Master	bm_struct.bmyld.retnua [i]	bm_struct.bmhead.fstyld and bm_struct.bmhead.lstyld
	"	"	Cross-Sectional	bx_struct.bxyld.retnua [i]	0 and <bx_struct.bxhead.numact
	DURATN	double	Master	bm_struct.bmyld.duratn [i]	bm_struct.bmhead.fstyld and bm_struct.bmhead.lstyld
	"	"	Cross-Sectional	bx_struct.bxyld.duratn [i]	0 and <bx_struct.bxhead.numact
DEBT	DQDATE	int	Master	bm_struct.bmdebt.qdate [i]	0 and <bm_struct.bmhead.numdbt
	TOTOUT	int	Master	bm_struct.bmdebt.totout [i]	0 and <bm_struct.bmhead.numdbt
	PUBOUT	int	Master	bm_struct.bmdebt.pubout [i]	0 and <bm_struct.bmhead.numdbt
PAYMENTS	PQDATE	int	Master	bm_struct.bmpay.qdate [i]	0 and <bm_struct.bmhead.numpay
	PDINT	double	Master	bm_struct.bmdebt.pdint [i]	0 and <bm_struct.bmhead.numpay

FORTRAN Sample Programs

The sample programs give short examples of how to access the CRSP Daily US Government Bond Data with the bond access routines using FORTRAN. The first two give basic examples of the FORTRAN sequential access to the character files, while the last four illustrate both sequential and random access to the binary files, using C access routines which are described later in this chapter. To use a sample program, copy it to your directory, edit the program to meet your needs and run according to the instructions inside the program.

BMSAMP Program *BMSAMP* reads the character calendar file and the character master file. *BMSAMP* first calls subroutine *BXCGTC* to read the character calendar file into the common block `/BXCAL/`. *BMSAMP* then makes successive calls to *BMGETC,* each call reading all data for one issue from data files into the common blocks `/BMHEAD/` (header information), `/BMQUO/` (quotes information), `/BMYLD/` (yield information), `/BMDEBT/` (debt information) and `/BMPAY/` (payment information).

BXSAMP Program *BXSAMP* reads the character calendar file and the character cross-sectional file. *BXSAMP* first calls subroutine *BXCGTC* to read the character calendar file into the common block `/BXCAL/`. *BXSAMP* then makes successive calls to *BXGETC*, each call reading all data for one quote date from the data files into the common blocks `/BXHEAD/` (header information), `/BXQUO/` (quotes information), `/BXYLD/` (yield information).

BMBFOR Program *BMBFOR* reads sequentially the Daily US Government Bond master binary files using C access functions. *BMBFOR* calls subroutine *BMBRDK* to read a `BM_STRUCT` structure. It also calls *BMBOPE* to open the files and load the index and *BMBCLO* to close the files.

BMBRAN Program *BMBRAN* reads randomly the Daily US Government Bond master binary files using C access functions. *BMBRAN* calls subroutine *BMBRDK* to read a `BM_STRUCT` structure. It also calls *BMBOPE* to open the files and *BMBCLO* to close the files.

BXBFOR Program *BXBFOR* reads sequentially the Daily US Government Bond cross-sectional binary files using C access functions. *BXBFOR* calls subroutine *BXBRDK* to read a `BX_STRUCT` structure. It also calls *BXBOPE* to open the files and load the index and *BXBCLO* to close the files.

BXBRAN Program *BXBRAN* reads sequentially the Daily US Government Bond cross-sectional binary files using C access functions. *BXBRAN* calls subroutine *BXBRDK* to read a `BX_STRUCT` structure. It also calls *BXBOPE* to open the files and load the index and *BXBCLO* to close the files.

FORTRAN Access Subroutines

CRSP Daily US Government Bond File FORTRAN access subroutines are used by FORTRAN programs to actually retrieve CRSP Daily US Government Bond data for processing. These subroutines should be included in an object library. You should link the library with each program that uses any of the access functions.

BMGETC (*, *) Subroutine BMGETC first calls BMRES to erase the previous record's data and then reads all data for one issue from the data files into the common blocks /BMHEAD/ (header information), /BXQUO/ (quotes information), /BMYLD/ (yield information), /BMDEBT/ (debt information) and /BMPAY/ (payment information). BMGETC first reads a header record and then reads LSTQUO - FSTQUO + 1 quotes records, LSTYLD - FSTYLD + 1 yield records, NUMDBT debt records and NUMPAY payment records. BMGETC makes sure that the CRSPID from the header and the data records are the same. The first alternate return is taken from the file. The second alternate return is taken if there is an error.

BXGETC (THEDAY, NUMREC, *,*)

Subroutine BXGETC first calls BXRES to erase the previous record's data and then reads all data for one quote date from the data files into the common blocks /BXHEAD/ (header information), /BXYLD/ (yield information). BXGETC has two parameters:

THEDAY - the quote date
NUMREC - the number of issues having the THEDAY quote date

BXGETC reads NUMREC quotes records and then NUMREC yield records. BXGETC makes sure that the parameter THEDAY and the quote date of the data records are the same and that the CRSPID of the quotes data is the same as the CRSPID of the yield data. The first alternate return is taken at the end of the file. The second alternate return is taken if there is an error.

BXCGTC Subroutine BXCGTC reads the character calendar file into the /BXCAL/ common block.

FORTRAN Utility Subroutines

CRSP Daily US Government Bond FORTRAN utility subroutines are used by FORTRAN programs to actually obtain different CRSP derived variables. These subroutines should also be included into the object library. You should link the library with each program that uses any of the utility functions.

FORTRAN Utility Subroutines

Subroutine	Type	Description
BMRES	BM	reset master structure
BXRES	BX	reset cross-sectional structure
BXCLJL	CAL	convert calendar date to Julian date
FPDINT	BM	derive paid interest for a date
IDBT	CAL	find index in debt array for a date
INDCAL	CAL	find index in a calendar for a date
INDCID	BX	find index in a CRSPID list for a CRSPID
IPAY	BM	find index in payment structure for a date
IQDAY	CAL	find DD day for a calendar index
JAHRMO	CAL	find year and month for a calendar index
NDDATE	CAL	find Julian day number of delivery date for a calendar index
NDHFYR	CAL	return number of days in last half year
NDIFDT	CAL	find difference in days between 2 dates
NDZERO	CAL	find zero'th day of a month
NFQDAT	CAL	find YYMMDD date from calendar index
NPOUT	BM	find publicly held value for calendar index
NQDATE	CAL	Julian day number for calendar index
NQTOQD	CAL	find number of days between given index and previous
NTOUT	BM	find total debt for calendar index
PCYIELD	BM	calculate yield to maturity compounded to given frequency
RETADJ	BM, BX	express holding period return as a percentage
YTM	BM, BX	calculate annualized yield to maturity

BMRES Subroutine BMRES resets the vectors belonging to the previous master structure. It initializes the /BMQUO/, /BMYLD/, /BMDEBT/ and /BMPAY/ common blocks.

BXRES Subroutine BXRES resets the vectors belonging to the previous master structure. It initializes the /BXHEAD/, /BMQUO/ and /BMYLD/ common blocks.

BXCLJL (IDTCAL, IDTJUL, *)

Subroutine BXCLJL converts a calendar date to its linear (Julian) date equivalent. IDTCAL is the integer YYYYMMDD date which BXCLJL should convert, IDTJUL is the converted (Julian) date which BXCLJL returns. The alternative return is used if IDTCAL is an illegal date.

INTEGER FPDINT (IDXCAL)

Function FPDINT takes as a parameter IDXCAL - index in the calendar, calls the IPAY function to get the index in the BMPAY vector corresponding to the calendar data and returns the paid interest for that date. FPDINT returns -1 if the date was not found.

INTEGER IDBT (IDXCAL)

Function IDBT takes as a parameter IDXCAL - index in the calendar, searches in the BMDEBT vector and returns the index in the BMDEBT vector corresponding to the calendar data. IDBT returns -1 if the date was not found.

INTEGER INDCAL (DATE, CODE, ARRAY, MAXARR)

Function INDCAL can be used to locate the index of a date in a given date array. DATE is the value to be located in array ARRAY with MAXARR sorted values. CODE is one of -1, 0, 1, depending of what action is taken when the exact given date is not found. If CODE = 0 and the exact date is not found, 0 is returned. If CODE = -1 and the exact date is not found, the index of the first date less than DATE is returned, or 0 is returned if DATE is less than any date in the array. If CODE = 1 and the exact date is not found, the index of the first date greater than DATE will be returned, or 0 is returned if DATE is greater than any date in the array.

INTEGER INDCID (CRSPID, CODE, ARRAY, MAXARR)

Function INDCID can be used to locate the index of a CRSPID in a given CRSPIDs array. CRSPID is the value to be located in array ARRAY with MAXARR sorted values. CODE is one of -1, 0, 1, depending of what action is taken when the CRSPID is not found. If CODE = 0 and the CRSPID is not found, 0 is returned. If CODE = -1 and the CRSPID is not found, the index of the previous CRSPID in the array is returned, or 0 is returned if CRSPID is the first one in the array. If CODE = 1 and the CRSPID is not found, the index of the next CRSPID in the array will be returned, or 0 is returned if CRSPID is the last one in the array.

INTEGER IPAY (IDXCAL)

Function IPAY takes as a parameter IDXCAL - index in the calendar, searches in the BMPAY vector and returns the index in the BMPAY vector corresponding to the calendar data. IPAY returns -1 if the date was not found.

INTEGER IQDAY (IDXCAL)

Function IQDAY takes as a parameter IDXCAL and returns the day (DD) of the quotation date which has index IDXCAL. Returns -1 if IDXCAL is out of range.

INTEGER JAHRMO (IDXCAL)

Function JAHRMO takes as a parameter IDXCAL and returns the year and month (YYYYMM) of the quotation date which has index IDXCAL. Returns -1 if IDXCAL is out of range.

INTEGER NDDATE (IDXCAL)

Function NDDATE takes as a parameter IDXCAL and returns the day number of the of the delivery date which has index IDXCAL. NDDATE calls the BXCLJL function to get the day number. Returns -1 if IDXCAL is out of range or if BXCLJL fails.

INTEGER NDHFYR (IDXCAL)

Function NDHFYR takes as a parameter IDXCAL and returns the number of days in the last half year corresponding to the quotation date which has index IDXCAL. NDHFYR calls the NDIFDT function to get the difference between the quotation date. Returns -1 if IDXCAL is out of range.

INTEGER NDIFDT (IDAT1, IDAT2)

Function NDIFDT converts two calendar dates to linear (Julian) dates and returns the difference. IDAT1 and IDAT2 are integer YYYYMMDD dates. NDIFDT calls the BXCLJL function to calculate the linear (Julian) dates.

INGETER NDZERO (IDXCAL)

Function NDZERO takes as a parameter IDXCAL and returns the zero'th day of the month of the quotation date which has index IDXCAL. NQDATE calls the BXCLJL function to get the linear date. Returns -1 if IDXCAL is out of range or if BXCLJL fails.

INTEGER NFQDAT (IDXCAL)

Function NFQDAT takes as a parameter IDXCAL and returns the quotation date (YYMMDD) which has index IDXCAL. Returns -1 if IDXCAL is out of range.

INTEGER NPOUT (IDXCAL)

Function NPOUT takes as a parameter IDXCAL - index in the calendar, calls the IDBT function to get the index in the BMDEBT vector corresponding to the calendar data and returns the publicly held face value outstanding for that date. NPOUT returns -1 if the date was not found.

INTEGER NQDATE (IDXCAL)

Function NQDATE takes as a parameter IDXCAL and returns the day number of the quotation date which has index IDXCAL. NQDATE calls the BXCLJL function to get the day number. Returns -1 if IDXCAL is out of range or if BXCLJL fails.

INTEGER NQTOQD (IDXCAL)

Function NQTOQD takes as a parameter IDXCAL and returns the number of days between the previous quotation date and the quotation date which has index IDXCAL. NQTOQD calls the NQDATE function to get the linear (Julian) quotation dates. Returns -1 if IDXCAL is out of range.

INTEGER NTOUT (IDXCAL)

Function NTOUT takes as a parameter IDXCAL - index in the calendar, calls the IDBT function to get the index in the BMDEBT vector corresponding to the calendar data and returns the face value outstanding for that date. NTOUT returns -1 if the date was not found.

PCYLD (PCYARR, FREQ)

Subroutine PCYLD calculates the yield to maturity. PCYLD has two parameters:

PCYARR - an array of floats which will be loaded with the calculated values
FREQ - the frequency

If a yield is missing, the value will be -99

RETADJ (ADJARR) Subroutine RETADJ calculates the holding period return expressed as a percentage. RETADJ has a parameter:

ADJARR - an array of floats which will be loaded with the calculated values.

If RETNUA, the unadjusted return, is missing, the value will be -999

YTM (YTMARR) Subroutine YTM calculates the annualized yield to maturity. YTM has a parameter:

YTMARR - an array of floats which will be loaded with the calculated values

If a yield is missing, the value will be -999

FORTRAN Include Files

The Daily US Government Bond sample programs and subroutines use include files to replace long, often-used blocks of code with single statements. If an include file is modified, all programs and subroutines that use the include file must be recompiled. All declarations needed to use the CRSP data with FORTRAN programs are automatically made by adding the include statements at the beginning of any main programs or subprograms that will use CRSP data or CRSP access or utility routines. The contents of these files are printed in Appendix B.

BMINCL Include file *BMINCL* contains constants definitions and common blocks definitions to be used in any program or subroutine which access the master files.

BXINCL Include file *BXINCL* contains constants definitions and common blocks definitions to be used in any program or subroutine which access the cross-sectional files.

CALINC Include file *CALINC* contains constants definitions and common blocks definitions to be used in any program or subroutine which access the calendar file.

BMBPRM Include file *BMBPRM* contains constants definitions to be used by programs or subroutines which access the master files using C functions.

BXBPRM Include file *BXBPRM* contains constants definition to be used by programs or subroutines which access the cross-sectional files using C functions.

C Sample Programs

The sample programs give short examples of how to access the CRSP Daily US Government Bond data with the bond access routines using C. The first four give basic examples of the C sequential and random access to the binary files, while the last four illustrate both sequential and random access to the binary files. The last two are the programs that generate the binary files from the character files. To use a sample program, copy it to your directory, edit the program to meet your needs and run according to the instructions inside the program.

Character Files

Program:	***bmc_read_rand***
Description:	reads the character calendar file and then reads randomly the character master bond files.
Methodology:	*bmc_read_rand* first calls procedure `bxc_cal_load` to load the calendar in the `bx_cal` array and then reads sequentially the input file and calls the `bxc_rdkey` for each read `CRSPID`. The function `bxc_rdkey` loads all wanted data in the `bms` structure for the desired `CRSPID`.
Parameters:	`bndpath` - the path of the directory where the daily bond files are `inpfilename` - the input file name (including the path).
Return Values:	
Notes: The wanted `CRSPIDs` are read from a text file.	

Program:	***bmc_read_seq***
Description:	Reads the character calendar file and then reads sequentially the character master bond files.
Methodology:	*bmc_read_seq* first calls procedure `bxc_cal_load` to load the calendar in the `bx_cal` array and then reads bond data one `CRSPID` by one till the end of files. The function `bmc_rdkey` loads all wanted data in the `bms` structure for the next `CRSPID.`
Parameters:	`bndpath` - the path of the directory where the daily bond files are
Return Values:	
Notes:	

Program:	***bxc_read_rand***
Description:	Reads randomly the character cross-sectional bond files.
Methodology:	*bxc_read_rand* reads sequentially the input file and calls the `bxc_rdkey` for each read date. The function `bxc_rdkey` loads all wanted data in the `bxs` structure for the desired date.
Parameters:	`bndpath` - the path of the directory where the daily bond files are `inpfilename` - the input file name(including the path)
Return Values:	
Notes: The wanted dates are read from a text file.	

Program:	***bxc_read_seq***
Description:	Reads sequentially the character cross-sectional bond files.
Methodology:	*bxc_read_seq* reads bond data in a loop till the end of files. The function `bxc_rdkey` loads all wanted data in the `bxs` structure for the next date.
Parameters:	`bndpath` - the path of the directory where the daily bond files are
Return Values:	
Notes: The wanted dates are read from a text file.	

Binary Files

Program:	***bmb_read_rand***
Description:	Reads the binary calendar file and then reads randomly the binary master bond files.
Methodology:	*bmb_read_rand* first calls procedure `bxb_cal_load` to load the calendar in the `bx_cal` array and then reads sequentially the input file and calls the `bxb_rdkey` for each read `CRSPID`. The function `bxb_rdkey` loads all wanted data in the `bms` structure for the desired `CRSPID`.
Parameters:	`bndpath` - the path of the directory where the daily bond files are `inpfilename` - the input file name(including the path)
Return Values:	
Notes:	The wanted `CRSPIDs` are read from a text file.

Program:	***bmb_read_seq***
Description:	Reads the binary calendar file and then reads sequentially the binary master bond files.
Methodology:	*bmb_read_seq* first calls procedure `bxb_cal_load` to load the calendar in the `bx_cal` array and then reads bond data one `CRSPID` by one till the end of files. The function `bmb_rdkey` loads all wanted data in the `bms` structure for the next `CRSPID`.
Parameters:	`bndpath` - the path of the directory where the daily bond files are
Return Values:	
Notes:	

Program:	***bxb_read_rand***
Description:	Reads randomly the binary cross-sectional bond files.
Methodology:	*bxb_read_rand* reads sequentially the input file and calls the `bxb_rdkey` for each read date. The function `bxb_rdkey` loads all wanted data in the `bxs` structure for the desired date.
Parameters:	`bndpath` - the path of the directory where the daily bond files are `inpfilename` - the input file name(including the path)
Return Values:	
Notes:	The wanted dates are read from a text file.

Program:	***bxb_read_seq***
Description:	Reads sequentially the binary cross-sectional bond files.
Methodology:	*bxb_read_seq* reads bond data in a loop till the end of files. The function `bxb_rdkey` loads all wanted data in the `bxs` structure for the next date.
Parameters:	`bndpath` - the path of the directory where the daily bond files are
Return Values:	
Notes:	

Conversion Programs from Character to Binary

Program:	***bmc_bmb_conv***
Description:	Reads the character calendar file and then reads sequentially the character master bond files and writes into binary files the loaded structure.
Methodology:	*bmc_bmb_conv* first calls procedure `bxc_cal_load` to load the calendar in the `bx_cal array` and then reads bond data one `CRSPID` by one till the end of files and write the data. The function `bmc_rdkey` loads all wanted data in the `bms` structure for the next `CRSPID` and the function `bmb_wrkey` writes the structure into the binary files. The program also calls the `bxb_cal_write` to write the calendar array into the binary calendar file.
Parameters:	`bndpath` - the path of the directory where the daily bond files are
Return Values:	
Notes: Converts the master files from character to binary.	

Program:	***bxc_bxb_conv***
Description:	Reads sequentially the character cross-sectional bond files and writes the data into the binary cross-sectional files.
Methodology:	*bxc_bxb_conv* reads character data one date by one until the end of files and writes it into the binary files. The function `bxc_rdkey` loads all wanted data in the `bxs` structure for the next date and the function `bxb_wrkey` write the structure into the files.
Parameters:	`bndpath` - the path of the directory where the daily bond files are
Return Values:	
Notes: Converts the cross-sectional files from character to binary.	

C Access Routines

Functions Called by C Programs

CRSP bond daily C access subroutines are used by C programs to actually retrieve CRSP bond daily data for processing. These subroutines should be included into an object library. Link the library with each program that uses any of the access functions.

Character Files

Function:	**bmc_rdkey**
Prototype:	int bmc_rdkey (bm_str, key, wanted)
Description:	Reads the data from the character master bond files for the given key.
Parameters:	bm_str - pointer to the BM_STRUCT structure to be loaded key - the desired CRSPID for random access or MFIRST, MPREV, MLAST, MSAME, MNEXT wanted - the desired information; should be QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination.
Return Values:	success(0) the key found error(-1) for: -no read access -key not found or no previous for same -could not read a needed file EOFL (-2)

Function:	**bxc_rdkey**
Prototype:	int bxc_rdkey (bx_str, key, wanted)
Description:	Reads the data from the character cross-sectional files for the given key.
Parameters:	bx_str - pointer to the BX_STRUCT structure to be loaded key - the desired qdate for random access or XFIRST, XPREV, XLAST, XSAME, XNEXT wanted - the desired information; should be QUOTES, YIELDS, ALLBX or any combination
Return Values:	success(0) the key found error(-1) for: -no read access -key not found or no previous for same -could not read a needed file EOFL (-2)

Function:	**bxc_cal_load**
Prototype:	int bxc_cal_load (bndpath)
Description:	Function to load the character calendar from the BXCALIND.DAT file into the bx_cal array.
Parameters:	bndpath - the path of the directory where the calendar file is.
Return Values:	success(nbx_cal) - the number of dates error(-1)

Function:	bmc_open
Prototype:	int bmc_open (bndpath,wanted)
Description:	Opens wanted character master bond files and loads the index file in an array.
Parameters:	bndpath - the path of the directory where the bond data files are located wanted - the desired information should be QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination
Return Values:	success(0) error(-1)

Function:	bxc_open
Prototype:	int bxc_open (bndpath,wanted)
Description:	Opens wanted character cross-sectional files and loads the index file in an array.
Parameters:	bndpath - the path of the directory where the bond data files are located wanted - the desired information should be QUOTES, YIELDS, ALLBX or any combination
Return Values:	success(0) error(-1)

Function:	bmc_close
Prototype:	int bmc_close (wanted)
Description:	Opens wanted character in master files sequentially.
Parameters:	wanted - the desired information should be QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination
Return Values:	success(0) couldn't close(-1)

Function:	bxc_close
Prototype:	int bxc_close (wanted)
Description:	Opens wanted character in cross-sectional files sequentially.
Parameters:	wanted - the desired information should be QUOTES, YIELDS, ALLBX, or any combination
Return Values:	success(0) couldn't close(-1)

Binary Files

Function:	**bmb_rdkey**
Prototype:	int bmb_rdkey (bm_str, key, wanted)
Description:	Reads the data from the binary master bond files for the given key.
Parameters:	bm_str - pointer to the BM_STRUCT structure to be loaded key - the desired CRSPID for random access or MFIRST, MPREV, MLAST, MSAME, MNEXT wanted - the desired information; should be QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination
Return Values:	success(0) error(-1) for: -no read access -key not found or no previous for same -could not read a needed file EOFL (-2)

Function:	**bxb_rdkey**
Prototype:	int bxb_rdkey (bx_str, key, wanted)
Description:	Reads the data from the binary cross-sectional files for the given key.
Parameters:	bx_str - pointer to the BX_STRUCT structure to be loaded key - the desired CRSPID for random access or XFIRST, XPREV, MLAST, MSAME, MNEXT Wanted - the desired information; should be QUOTES, YIELDS, ALLBX or any combination
Return Values:	success(0) error(-1) for: -no read access -key not found or no previous for same -could not read a needed file EOFL (-2)

Function:	**bmb_wrkey**
Prototype:	int bmb_wrkey (bm_str)
Description:	Writes the data from the bm_str structure into the binary master bond files.
Parameters:	bm_str - pointer to the BM_STRUCT structure to be loaded
Return Values:	success(0) the key found error(-1)

Function:	**bxb_wrkey**
Prototype:	int bxb_wrkey (bx_str)
Description:	Writes the data from bx_str structure into the binary cross-sectional files.
Parameters:	bx_str - pointer to the BX_STRUCT structure to be loaded
Return Values:	success(0) error(-1)

Function:	bxb_cal_load
Prototype:	int bxb_cal_load (bndpath)
Description:	Function to load the binary calendar from the BXCALIND.DAT file into the bx_cal array.
Parameters:	bndpath - the path of the directory where the calendar file is
Return Values:	success(nbx_cal) - the number of dates error(-1)

Function:	bmb_open
Prototype:	int bmb_open (bndpath,wanted)
Description:	Opens wanted binary master bond files and loads the index file in an array.
Parameters:	bndpath - the path of the directory where the bond data files are wanted - the desired information; should be QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination
Return Values:	success(0) error(-1)

Function:	bmb_close
Prototype:	int bmb_close (wanted)
Description:	Close wanted binary master files sequentially.
Parameters:	wanted - the desired information; should be QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination
Return Values:	success(0) error(-1) couldn't close

Function:	bxb_open
Prototype:	int bxb_open (bndpath,wanted)
Description:	Opens wanted binary cross-sectional files and loads the index file in an array.
Parameters:	bndpath - the path of the directory where the bond data files are wanted - the desired information; should be QUOTES, YIELDS, ALLBX or any combination
Return Values:	success(0) error(-1) couldn't close

Function:	bxb_close
Prototype:	int bxb_close (wanted)
Description:	Close wanted binary cross-sectional files sequentially..
Parameters:	wanted - the desired information; should be QUOTES, YIELDS, ALLBX or any combination
Return Values:	success(0) error(-1) couldn't close

Functions Called by FORTRAN Programs

Function:	**bmbrdk**
Prototype:	`int bmbrdk (pbmhead, pbmquo, pbmyld, pbmpay, pbmdebt, key, wanted, ret)`
Description:	Reads the data from the master binary files for a given key and loads them into a `BM_STRUCT` structure and then into FORTRAN common blocks to be used by FORTRAN programs.
Parameters:	`pbmhead` - pointer to the /`BMHEAD`/ common block `pbmquo` - pointer to the /`BMQUO`/ common block `pbmyld` - pointer to the /`BMYLD`/ common block `pbmpay` - pointer to the /`BMPAY`/ common block `pbmdebt` - pointer to the /`BMDEBT`/ common block `key` - the desired `CRSPID` for random access or `MFIRST`, `MPREV`, `MLAST`, `MSAME`, `MNEXT` `wanted` - the desired information; should be `QUOTES`, `YIELDS`, `PAYMTS`, `DEBTS`, `ALLBM` or any combination `ret` - the return code
Return Values:	success(0) the key found, or error(-1) for: -no read access -key not found or no previous for same -could not read a needed file `EOFL` (-2)

Function:	**bxbrdk**
Prototype:	`int bxbrdk (pbxhead, pbxquo, pbxyld, key, wanted, ret)`
Description:	Reads the data from the cross-sectional binary files for a given key and load them into a `BX_STRUCT` structure and then into FORTRAN common blocks to be used by FORTRAN programs.
Parameters:	`pbxhead` - pointer to the /`BXHEAD`/ common block `pbxquo` - pointer to the /`BXQUO`/ common block `pbxyld` - pointer to the /`BXYLD`/ common block `key` - the desired `CRSPID` for random access or `XFIRST`, `XPREV`, `XLAST`, `XSAME`, `XNEXT` `wanted` - the desired information; should be `QUOTES`, `YIELDS`, `ALLBX` or any combination `ret` - the return code
Return Values:	success(0) the key found, or error(-1) for: -no read access -key not found or no previous for same -could not read a needed file `EOFL` (-2)

Function:	bxbcal
Prototype:	int bxbcal (pbxcal, nbxcal, bndpath, bndlen, ret)
Description:	Reads the data from the binary calendar file and load them into FORTRAN common block to be used by FORTRAN programs.
Parameters:	pbxcal - pointer to the /BXCAL/ common block nbxcal - the number of dates bndpath - the path of the directory where the file is bndlen - the length of the path ret - the return code
Return Values:	success(0) the key found, or error(-1)

Function:	bmbope
Prototype:	int bmbope (bndpath, bndlen, wanted, mode, ret)
Description:	Opens all bond master binary data files and loads the index file in the array. It calls the C function bmb_open.
Parameters:	bndpath - the path of the directory where the bond data files are bndlen - the length of the path wanted - the desired information; should be QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination mode - the mode to open the files should be "R" (read) or "W" (write) ret - the return code
Return Values:	success(0) the key found, or error(-1)

Function:	bmbclo
Prototype:	int bmbclo (wanted, ret)
Description:	close bond master binary data files sequentially.
Parameters:	wanted - the desired information; should be QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination ret - the return code
Return Values:	success(0), or error(-1)

Function:	bxbope
Prototype:	int bxbope (bndpath, bndlen, wanted, mode, ret)
Description:	Opens all bond cross-sectional binary data files and loads the index file in the array calling the C function bxb_open.
Parameters:	bndpath - the path of the directory where the bond data files are bndlen - the length of the path wanted - the desired information; should be QUOTES, YIELDS, ALLBX or any combination mode - the mode to open the files should be "R" (read) or "W" (write) ret - the return code
Return Values:	success(0), or error(-1)

Function:	**bxbclo**
Prototype:	int bxbclo (wanted, ret)
Description:	Close bond cross-sectional binary data files sequentially.
Parameters:	wanted - the desired information; should be QUOTES, YIELDS, ALLBX or any combination ret - the return code
Return Values:	success(0), or error(-1)

C Utility Routines

CRSP Daily US Government Bond C utility subroutines are used by C programs to actually obtain different CRSP derived variables. These subroutines should also be included in the object library. Link the library with each program that uses any of the utility functions.

Subroutine	Type	Description
bxcljl	cal	convert calendar date to Julian date
fpdint	bm	derive paid interest for a date
idbt	cal	find index in debt array for a date
indcal	cal	find index in a calendar for a date
indcid	bx	find index in a CRSPID list for a CRSPID
ipay	bm	find index in payment structure for a date
iqday	cal	find DD day for a calendar index
jahrmo	cal	find year and month for a calendar index
nddate	cal	find Julian day number of delivery date for a calendar index
ndhfyr	cal	return number of days in last half year
ndifdt	cal	find difference in days between 2 dates
ndzero	cal	find zero'th day of a month
nfqdat	cal	find YYMMDD date from calendar index
npout	bm	find publicly held value for calendar index
nqdate	cal	Julian day number for calendar index
nqtoqd	cal	find number of days between given index and previous
ntout	bm	find total debt for calendar index
pcyield	bm	calculate yield to maturity compounded to given frequency
retadj	bm, bx	express holding period return as a percentage
ytm	bm, bx	calculate annualized yield to maturity

Utility:	**bxcljl**
Prototype:	int bxcljl (idtcal)
Description:	Function bxcljl converts a calendar date to its linear (Julian) date equivalent.
Parameters:	idtcal - date in format YYYYMMDD
Return Values:	success: the linear date error(-1)

Utility:	**fpdint**
Prototype:	int fpdint (bm_str, idxcal)
Description:	Function fpdint takes as a parameter idxcal - index in the calendar, calls the ipay function to get the index in the bmpay vector corresponding to the calendar data and returns the paid interest for that date.
Parameters:	bm_str - pointer to a BM_STRUCT structure which must be loaded before this function to be called idxcal - the index in the calendar
Return Values:	success: an index in the BMPAY structure error(-1) fpdint returns -1 if the date was not found.

Utility:	idbt
Prototype:	int idbt(bm_str, idxcal)
Description:	Function idbt takes as a parameter idxcal - index in the calendar, searches in the bmdebt vector and returns the index in the bmdebt vector corresponding to the calendar data.
Parameters:	bm_str - pointer to a BM_STRUCT structure which must be loaded before this function to be called idxcal - the index in the calendar
Return Values:	success: an index in the bmdebt structure error(-1)

Utility:	indcal
Prototype:	int indcal (key, code, array, maxarr)
Description:	indcal can be used to locate the index of a YYYYMMDD date in a calendar array.
Parameters:	key - pointer to a string containing a YYYYMMDD calendar date to find code -1, 0, 1 for handling non-exact matches -1, if date is not found, returns index of previous valid date 0, if date is not found, returns 0 1, if date is not found, returns index of next valid date array - pointer to an array of YYYYMMDD calendar dates maxarr - number of calendar dates in array
Return Values:	success: index in array of YYYYMMDD calendar dates error(0) if not found or out of range according to code

Utility:	indcid
Prototype:	int indcid (key, code, array, maxarr)
Description:	indcid can be used to locate the index of a CRSPID in a CRSPID array.
Parameters:	key - pointer to a string containing a CRSPID to find code -1, 0, 1 for handling non-exact matches -1, if CRSPID is not found, returns index of previous CRSPID 0, if CRSPID is not found, returns 0 1, if CRSPID is not found, returns index of next CRSPID array - pointer to an array of CRSPIDs maxarr - number of CRSPIDs in array
Return Values:	success: index in array of CRSPIDs error(0) if not found or out of range according to code

Utility:	ipay
Prototype:	int ipay (bm_str, idxcal)
Description:	Function ipay takes as a parameter idxcal - index in the calendar, searches in the bmpay vector and returns the index in the bmpay vector corresponding to the calendar data.
Parameters:	bm_str - pointer to a BM_STRUCT structure which must be loaded before this function to be called idxcal - the index in the calendar
Return Values:	success: an index in the bmpay structure error(-1)

Utility:	**iqday**
Prototype:	`int iqday (idxcal)`
Description:	Function `iqday` takes as a parameter `idxcal` and returns the day (DD) of the quotation date which has index `idxcal`.
Parameters:	`idxcal` - the index in the calendar
Return Values:	success: the day of the quotation date error(-1)

Utility:	**jahrmo**
Prototype:	`int jahrmo (idxcal)`
Description:	Function `jahrmo` takes as a parameter `idxcal` and returns the year and month (YYYYMM) of the quotation date which has index `idxcal`.
Parameters:	`idxcal` - the index in the calendar
Return Values:	success: year and month of the quote date YYYYMM error(-1)

Utility:	**nddate**
Prototype:	`int nddate (idxcal)`
Description:	Function `nddate` takes as a parameter `idxcal` and returns the day number of the of the delivery date which has index `idxcal`. `nddate` calls the `bxcljl` function to get the day number.
Parameters:	`idxcal` - the index in the calendar
Return Values:	success: the day number of the delivery date error(-1)

Utility:	**ndhfyr**
Prototype:	`int ndhfyr (idxcal)`
Description:	Function `ndhfyr` takes as a parameter `idxcal` and returns the number of days in the last half year corresponding to the quotation date which has index `idxcal`. `ndhfyr` calls the `ndifdt` function to get the difference between the quotation date.
Parameters:	`idxcal` - the index in the calendar
Return Values:	success: the linear number of dates in half year error(-1)

Utility:	**ndifdt**
Prototype:	`int ndifdt (idat1, idat2)`
Description:	Function `ndifdt` converts two calendar dates to linear (Julian) dates and returns the difference.
Parameters:	`idat1` - first date in format YYYYMMDD `idat2` - second date in format YYYYMMDD
Return Values:	success: the difference between `idat1` and `idat2` error(-1)
`ndifdt` calls the `BXCLJL` function to calculate the linear (Julian) dates.	

Utility:	ndzero
Prototype:	int ndzero (idxcal)
Description:	Function ndzero takes as a parameter idxcal and returns the zero'th day of the month of the quotation date which has index idxcal. nqdate calls the bxcljl function to get the linear date.
Parameters:	idxcal - the index in the calendar
Return Values:	success: the day number of the zero'th day of the month error(-1)

Utility:	nfqdat
Prototype:	int nfqdat (idxcal)
Description:	Function nfqdat takes as a parameter idxcal and returns the quotation date (YYMMDD) which has index idxcal.
Parameters:	idxcal - the index in the calendar
Return Values:	success: the quotation date YYMMDD error(-1)

Utility:	npout
Prototype:	int npout (bm_str, idxcal)
Description:	Function npout takes as a parameter idxcal - index in the calendar, calls the idbt function to get the index in the bmdebt vector corresponding to the calendar data and returns the publicly held face value outstanding for that date. npout returns -1 if the date was not found.
Parameters:	bm_str - pointer to a BM_STRUCT structure which must be loaded before this function to be called idxcal - the index in the calendar
Return Values:	success: an index in the bmdebt structure error(-1)

Utility:	nqdate
Prototype:	int nqdate (idxcal)
Description:	Function nqdate takes as a parameter idxcal and returns the day number of the quotation date which has index idxcal. nqdate calls the bxcljl function to get the day number.
Parameters:	idxcal - the index in the calendar
Return Values:	success: the day number of the quotation date error(-1)

Utility:	nqtoqd
Prototype:	int nqtoqd (idxcal)
Description:	Function nqtoqd takes as a parameter idxcal and returns the number of days between the previous quotation date and the quotation date which has index idxcal. nqtoqd calls the nqdate function to get the linear (Julian) quotation dates.
Parameters:	idxcal - the index in the calendar
Return Values:	success: the number of days between the last the quotation date and this quotation date error(-1)

Utility:	**ntout**
Prototype:	`int ntout (bm_str, idxcal)`
Description:	Function `ntout` takes as a parameter `idxcal` - index in the calendar, calls the `idbt` function to get the index in the `bmdebt` vector corresponding to the calendar data and returns the face value outstanding for that date. `ntout` returns -1 if the date was not found.
Parameters:	`bm_str` - pointer to a `BM_STRUCT` structure which must be loaded before this function to be called `idxcal` - the index in the calendar
Return Values:	success: an index in the BMDEBT structure error(-1)

Utility:	**pcyield**
Prototype:	`void pcyield (bm_str, pcyarr, freq)`
Description:	`pcyield` calculates the yield to maturity and loads the `pcyarr` with the results.
Parameters:	`bm_str` - pointer to a `BM_STRUCT` structure which must be loaded before this function to be called `pcyarr` - pointer to an array of floats `freq` - the frequency
Return Values:	success: none error: if a yield is missing, the value will be -99

Utility:	**retadj**
Prototype:	`void retadj (bm_str, adjarr, freq)`
Description:	`retadj` calculates the holding period return expressed as a percentage and loads the `adjarr` with the results.
Parameters:	`bm_str` - pointer to a `BM_STRUCT` structure which must be loaded before this function to be called `adjarr` - pointer to an array of floats
Return Values:	success: none error: If `RETNUA` is missing, the value will be -999

Utility:	**ytm**
Prototype:	`void ytm (bm_str, ytmarr, freq)`
Description:	`ytm` calculates the annualized yield to maturity and loads the `ytmarr` with the results.
Parameters:	`bm_str` - pointer to a `BM_STRUCT` structure which must be loaded before this function to be called `ytmarr` - pointer to an array of floats
Return Values:	success: none error: If a yield Is missing, the value will be -999

C Input/Output Routines

These functions are specific to file access (open, close, read sequentially, read randomly). They should be modified according to the compiler's requirements.

Routine:	`file_open`
Prototype:	`int file_open (filepath, filename)`
Description:	opens a file.
Parameters:	`filepath` - the path of the directory where the file is `filename` - the name of the file
Return Values:	success the file descriptor(>0) error(-1)

Routine:	`file_read`
Prototype:	`int file_read (fdes, buffer, offset, size)`
Description:	reads and stores the data temporary in a character buffer.
Parameters:	`fdes` - file descriptor `buffer` - character buffer where the data is read `offset` - the offset from the beginning of the file from where to read `size` - the number of bytes to be read
Return Values:	success(0) error(-1)

Routine:	`file_write`
Prototype:	`int file_write (fdes, buffer, size)`
Description:	Writes a buffer into a file.
Parameters:	`fdes` - file descriptor `buffer` - the address of the buffer `size` - the number of bytes to be read
Return Values:	success(0) error(-1)

Routine:	`file_next`
Prototype:	`int file_next (fdes, buffer, size)`
Description:	Reads sequentially the next record and stores the data temporary in a character buffer.
Parameters:	`fdes` - file descriptor `buffer` - character buffer where the data is read `size` - the number of bytes to be read
Return Values:	Success(the number of bytes) EOF or error(-1)

Routine:	`file_close`
Prototype:	`int file_close(fdes)`
Description:	Close a file.
Parameters:	`fdes` - file descriptor
Return Values:	success(0) error(-1)

C Include Files

The following include files were used in the Daily US Government Bonds sample programs, function and procedures. If an include file is modified, all programs, procedures and functions that used the include file must be recompiled. All declarations needed to use the CRSP data with C programs are automatically made by adding the include statements at the beginning of any main programs or subprograms that will use CRSP data or CRSP access or utility routines. The contents of these files are printed in Appendix C.

bnd_const.h Include file *bnd_const.h* contains the constants definitions for programs which access the data in Daily US Government Bond Files.

bnd_struct.h Include file *bnd_struct.h* contains the structures definitions for programs which access the data in Daily US Government Bond Files.

3.2 Daily US Government Bond File Specifications.

The tables below detail the exact specifications of the formatted CRSP files. Each table represents one file on the CD. The table names match the names in the CD layout descriptions in Appendix C. The following column specific information is true for all of the tables listed below. The "Character Positions" show where in the character record each field is positioned. The "FORTRAN Format" and "C Format" are the formats that appear on the CD. The "Associated Name" refers to the data item defined in the "Description of Variables" section of this manual.

Records are all fixed-length. File names beginning with `BX` are sorted by `QDATE`, then `CRSPID`. Fields are delimited by a pipe (`|`). The sort order is the same for ASCII or EBCDIC files.

Daily US Government Bond Master File Specifications

These files are sorted by `CRSPID`, then quote date where available.

Header File Specifications

This table details the exact specifications of the formatted CRSP daily bond header file. The CRSP Daily Bond Header File contains one record for each issue, sorted by `CRSPID`. This file has a 156 character record.

`BMHEADER` Data Records

Character Positions	FORTRAN Format	C Format	Data Type	Associated Name
1 - 15	A15	%15s	Character	CRSPID
17 - 24	A8	%8s	Character	CUSIP
26 - 33	A8	%8s	Character	NAME
35 - 42	I8	%8d	Integer	MATDT
44	I1	%1d	Integer	TYPE
46 - 52	F7.3	%7.3f	Double	COUPRT
54	I1	%1d	Integer	UNIQ
56	I1	%1d	Integer	WHY
58 - 65	I8	%8d	Integer	DATDT
67 - 74	I8	%8d	Integer	BANKDT
76 - 83	I8	%8d	Integer	FCALDT
85 - 90	I6	%6d	Integer	YMCNOT
92	I1	%1d	Integer	NOTICE
94	I1	%1d	Integer	TAX
96	I1	%1d	Integer	FLOWER
98	I1	%1d	Integer	NIPPY
100 - 107	I8	%8d	Integer	FCPDT
109	I1	%1d	Integer	FCPDTF
111 - 119	F9.6	%9.6f	Double	VALFC
121 - 125	I5	%5d	Integer	NUMPAY
127 - 131	I5	%5d	Integer	NUMDBT
133 - 137	I5	%5d	Integer	FSTQUO
139 - 143	I5	%5d	Integer	LSTQUO
145 - 149	I5	%5d	Integer	FSTYLD
151 - 155	I5	%5d	Integer	LSTYLD

Quotes File Specifications

This table details the exact specifications of the formatted CRSP daily bond quotes file. The CRSP Daily Bond Quotes File contains one record for each quote, sorted by CRSPID then QDATE. This file has a 52 character record.

BMQUOTES Data Record

Character Positions	FORTRAN Format	C Format	Data Type	Associated Name
1 - 15	A15	%15s	Character	CRSPID
17 - 24	I8	%8d	Integer	QDATE
26 - 36	F11.6	%11.6f	Double	BID
38 - 48	F11.6	%11.6f	Double	ASK
50	A1	%1s	Character	SOURCE

Yield File Specifications

This table details the exact specifications of the formatted CRSP daily bond yields file. The CRSP daily bond yields file contains one record for each quote, sorted by CRSPID then QDATE. This file has a 74 character record.

BMYIELD Data Records

Character Positions	FORTRAN Format	C Format	Data Type	Associated Name
1 - 15	A15	%15s	Character	CRSPID
17 - 24	I8	%8d	Integer	QDATE
26 - 38	E13.6	%13.6E	Double	ACCINT
40 - 52	E13.6	%13.6E	Double	YLD
54 - 66	E13.6	%13.6E	Double	RETNUA
68 - 73	F6.1	%6.1f	Double	DURATN

Debt File Specifications

This table details the exact specifications of the formatted CRSP daily bond debt file. The CRSP Daily Bond Debt File contains one record for each amount outstanding observation sorted by CRSPID then PQDATE. This file has a 38 character record.

BMDEBT Data Records

Character Positions	FORTRAN Format	C Format	Data Type	Associated Name
1 - 15	A15	%15s	Character	CRSPID
17 - 24	I8	%8d	Integer	QDATE
26 - 30	I5	%5d	Integer	TOTOUT
32 - 36	I5	%5d	Integer	PUBOUT

Coupon Payments File Specifications

This table details the exact specifications of the formatted CRSP daily bond payments file. The CRSP Daily Bond Payments File contains one record for each amount outstanding observation, sorted by CRSPID then DQDATE. This file has a 36 character record.

BMPAYMTS Data Records

Character Positions	FORTRAN Format	C Format	Data Type	Associated Name
1 - 15	A15	%15s	Character	CRSPID
17 - 24	I8	%8d	Integer	QDATE
26 - 34	F9.6	%9.6f	Double	PDINT

Address File Specifications

This table details the exact specifications of the formatted CRSP daily bond master address file. The CRSP Daily Bond Master Address File contains one record for each issue, and contains header information used by CRSP sample programs to read other master files randomly. This file has a 95 character record.

BMADDRS Data Records

Character Positions	FORTRAN Format	C Format	Data Type	Associated Name
1 - 15	A15	%15s	Character	CRSPID
17 - 25	I9	%9d	Integer	DBTLOC
27 - 35	I9	%9d	Integer	DBTSIZ
37 - 45	I9	%9d	Integer	PAYLOC
47 - 55	I9	%9d	Integer	PAYSIZ
57 - 65	I9	%9d	Integer	QUOLOC
67 - 75	I9	%9d	Integer	QUOSIZ
77 - 85	I9	%9d	Integer	YLDLOC
87 - 95	I9	%9d	Integer	YLDSIZ

Daily US Government Bond Cross-Sectional File Specifications

These files are sorted by QDATE, then CRSPID where available.

Calendar File Specification

This table details the exact specifications of the formatted CRSP Daily Bond Calendar/Rates File. The CRSP Daily Bond Calendar/Rates File contains one record for each Quote Date, sorted by QDATE. This file has an 87 character record.

BXCALIND Data Records

Character Positions	FORTRAN Format	C Format	Data Type	Associated Name
1 - 8	I8	%8d	Integer	QDATE
10 - 17	I8	%8d	Integer	DELDAT
19 - 24	F6.2	%6.2f	Real	CD1M
26 - 31	F6.2	%6.2f	Real	CD3M
33 - 38	F6.2	%6.2f	Real	CD6M
39 - 44	F6.2	%6.2f	Real	CP30D
46 - 51	F6.2	%6.2f	Real	CP60D
53 - 59	F6.2	%6.2f	Real	CP90D
61 - 66	F6.2	%6.2f	Real	FFEFRT
68 - 73	F6.2	%6.2f	Real	FFMINR
75 - 80	F6.2	%6.2f	Real	FFMAXR
82 - 87	I6	%6d	Integer	NUMACT

Quotes File Specifications

This table details the exact specifications of the formatted CRSP Daily Bond Cross-Sectional Quotes File. The CRSP Daily Bond Cross-Sectional Quotes File contains one record for each quote, sorted by QDATE then CRSPID. This file has a 52 character record.

BXQUOTES Data Records

Character Positions	FORTRAN Format	C Format	Data Type	Associated Name
1 - 8	I8	%8d	Integer	CRSPID
10 - 24	A15	%15s	Character	QDATE
26 - 36	F11.6	%11.6f	Double	BID
38 - 48	F11.6	%11.6f	Double	ASK
50	A1	%1s	Character	SOURCE

Yield File Specifications

This table details the exact specifications of the formatted CRSP Daily Bond Cross-Sectional Yields File. The CRSP Daily Bond Cross-Sectional Yields File contains one record for each quote, sorted by QDATE then CRSPID. This file has a 74 character record.

BXYIELD Data Records

Character Positions	FORTRAN Format	C Format	Data Type	Associated Name
1-8	I8	%8d	Integer	CRSPID
10-24	A15	%15s	Character	QDATE
26-38	E13.6	%13.6E	Double	ACCINT
40-52	E13.6	%13.6E	Double	YLD
54-66	E13.6	%13.6E	Double	RETNUA
68-73	F6.1	%6.1f	Double	DURATN

Address File Specifications

This table details the exact specifications of the formatted CRSP Daily Bond Cross-Sectional Address File. The CRSP Daily Bond Cross-Sectional Address File contains one record for each issue, and contains header information used by CRSP sample programs to read other Cross-Sectional files randomly. This file has a 49 character record.

BXADDRS Data Records

Character Positions	FORTRAN Format	C Format	Data Type	Associated Name
1 - 8	I8	%8d	Integer	CRSPID
10 - 18	I9	%9d	Integer	QUOLOC
20 - 28	I9	%9d	Integer	QUOSIZ
30 - 38	I9	%9d	Integer	YLDLOC
40 - 48	I9	%9d	Integer	YLDSIZ

Daily Fixed Term Indices File Specifications

This table details the exact specifications of the formatted CRSP Daily Bond Fixed Term Indices. The CRSP Fixed Term Indices File contains one record for each maturity term type for each quote date. This file has a 102 character record.

BXDLYIND Data Records

Character Positions	FORTRAN Format	C Format	Data Type	Associated Name
1-4	I4	%4d	Integer	TERMTYPE
6-13	I8	%8d	Integer	QDATE
15-29	A15	%15s	Character	CRSPID
31-35	F6.3	%6.3f	Integer	YEARSTM
38-48	F11.6	%11.6f	Double	RETADJ
50-60	F11.6	%11.6f	Double	YTM
62-72	F11.6	%11.6f	Double	ACCINT
74-79	F6.1	%6.1f	Double	DURATN
81-90	F10.6	%10.6f	Double	BID
92-101	F10.6	%10.6f	Double	ASK

Excel Files

Description

The Excel 5.0/95 Workbook files do not contain the large CRSP Daily US Government Bond Master and Cross-Sectional Files. These files are too large to be supported in Excel. The Excel Files were imported from the ASCII files. The number of decimal places matches those in the original ASCII Files. Therefore, adding decimal places in the cell formatting will not improve accuracy in data output. The dates are stored as Excel dates and displayed in a MM/DD/YY format, unless otherwise indicated on the readme worksheet. The first worksheet in each file is a readme worksheet that outlines the contents of the rest of the sheets.

The following table contains the file name, the work sheet names within them, and the section of the documentation, which describes them.

CRSP Monthly US Government Bond Files in Excel.

Files	Work Sheet Names	Documentation Reference
bmheader.xls	BMHEADER.XLS	Section 3.2, Daily Us Government Bond File Specifications Master File Specifications
bxcalind.xls	BXCALIND.XLS	Section 3.2, Daily US Government Bond Cross-Sectional File Specifications
bxdlyind_10yr.xls	BXDLYIND_10YR.XLS	Section 3.2, Daily Fixed Term Indices File Specifications
bxdlyind_1yr.xls	BXDLYIND_1YR.XLS	Section 3.2, Daily Fixed Term Indices File Specifications
bxdlyind_20yr.xls	BXDLYIND_20YR.XLS	Section 3.2, Daily Fixed Term Indices File Specifications
bxdlyind_2yr.xls	BXDLYIND_2YR.XLS	Section 3.2, Daily Fixed Term Indices File Specifications
bxdlyind_30yr.xls	BXDLYIND_30YR.XLS	Section 3.2, Daily Fixed Term Indices File Specifications
bxdlyind_5yr.xls	BXDLYIND_5YR.XLS	Section 3.2, Daily Fixed Term Indices File Specifications
bxdlyind_7yr.xls	BXDLYIND_7YR.XLS	Section 3.2, Daily Fixed Term Indices File Specifications

Microsoft Excel Support Disclaimer

CRSP does not support Microsoft Excel. These files have been included in this format as a courtesy. If you are unable to load the files or to use the software, please contact Microsoft or your System Administrator for support. These files are in ASCII in the \DATA\ directory if you want to convert them yourself.

SAS Files

Description

The complete CRSP Daily US Government Bond Master File was imported into one large SAS Transport Format, `dlybonds.trp`. Sample SAS code is included in the `bdimport.sas` file may expand the transport data set. The table below has SAS's NT file extensions. File extensions may vary among platforms. `indexdly.sas` can be run to create indices to speed retrieval and to create the data and cross-sectional order. Files created with the `indexdly.sas` program are indicated below with an asterick.

CRSP Monthly US Government Bond Files in SAS (extracted)

Extracted File Names	Documentation Reference
BMDEBT.SD2 BMDEBT.SI2	Section 3.2 Debt File Specifications (Master File) CRSPID index for the BMDEBT File
BMHEADER.SD2 BMHEADER.SI2*	Section 3.2 Header File Specifications (Master File) CRSPID index for the BMHEADER File
BMPAYMTS.SD2 BMPAYMTS.SI2*	Section 3.2 Coupon Payments File Specifications (Master File) CRSPID index for the BMPAYMTS File
BMQUOTES.SD2 BMQUOTES.SI2*	Section 3.2 Quotes File Specifications (Master File) CRSPID index for the BMQUOTES File
BMYIELD.SD2 BMYIELD.SI2*	Section 3.2 Yield File Specifications (Master File) CRSPID index for the BMYIELD File
BXCALIND.SD2	Section 3.2 Calendar File Specifications (Cross Sectional File)
BXDLYIND.SD2*	Section 3.2 Fixed Term Indices (Cross Sectional File)
BXQUOTES.SD2* BXQUOTES.SI2*	Section 3.2 Quote File Specifications (Cross Sectional File) QDATE index for BXQUOTES
BXYIELD.SD2* BSYIELD.SI2*	Section 3.2 Yield File Specifications (Cross Sectional File) QDATE index for BXYIELD

SAS Support Disclaimer

CRSP does not support SAS. These files have been included in this format as a courtesy. If you are unable to load the files or to use the software, please contact SAS or your System Administrator for support. The files are in ASCII in the \DATA\ directory if you want to convert them yourself.

3.3 Suggestions for installation of CRSP Daily US Government Bond Data

The CRSP Daily US Government Bond File has been created in tabular format to make it easy to use with CRSP sample programs or other tools. The files can be directly loaded into relational databases or statistical packages as well as used with CRSP sample programs that can support sequential or random access. The data were split into independent files that can be managed in parts or groups. It is highly recommended that the Bond Files are loaded from the CD and accessed on disk.

There are three possible strategies for using the files:

1. Use with third party tools or applications, or user-created programs.

2. CRSP FORTRAN and/or C access of the character files.

3. Conversion to binary with CRSP C conversion programs and CRSP FORTRAN and /or C access of the binary files.

If using the first option, see the CD Layouts in Appendix C and File Specifications in section 3.2. If exclusively using CRSP sample programs, it is recommended to convert to binary to take advantage of random access and smaller files.

Installation and Use of CRSP Sample Programs

This section provides the information needed to install the programs and data from CDs and to convert the master and cross-sectional bond data from character format to binary format and use the sample programs that CRSP provides to access the data.

The files on CD do not have installation scripts. Therefore, we recommend following the strategy outlined below for utilizing the programs described in Section 3.1.

- Load the CD in your CD drive.

- The system will mount the CD for NT and Sun Solaris.

  A sample OpenVMS mount command would be (device name `dka600:` may be different depending on your machine.):

  ```
  mount /media=cd dka600: BMR1_199712 /undefined_fat=(stream_lf:500)
  - /shared /bind=BMR1_199712#1
  ```

  A sample Digital Unix Installation on Digital Alpha mount command would be: (device name `/dev/rz4c` may be different depending on your machine.)

  ```
  mount -t cdfs -o noversion /dev/rz4c /cd
  ```

Programs were developed on an OpenVMS system and tested on Sun OS Unix. Standard FORTRAN and C functions were used whenever possible, but users will have to make minor modifications to open statements and include files on some systems.

In the FORTRAN sources the open statements are performed in the sample programs only, the access and utility subroutines assume that all files are already opened.

The C programs and subroutines use generic input/output functions, so only these routines should need to be modified. The generic C input/output routines are contained in the file `file_fncts.c`. Some modifications are also necessary in the FORTRAN sample programs that call the C access routines, according to specific compiler requirements regarding passing parameters between C and FORTRAN. The provided programs and listings are specific to VMS C compiler.

We recommend following the strategy outlined below for utilizing the programs described earlier in this section.

Copy the sample programs, subroutines, and include files from the CD to disk. Choose only sources that are suitable for you according to the following table:

Language	Access	Data Files Type	Necessary Files
FORTRAN	Sequential	Character	for_samp.for for_sub.for for_incl.txt
FORTRAN + C access	Sequential + Random	Binary	for_samp.for for_sub.for for_incl.txt c_sub.c c_incl.h
C	Sequential + Random	Character + Binary	c_samp.c c_sub.c c_incl.h

Separate the files (which consist of program segments merged together) into their individual components according to the following table:

CD File	Component Name	Line Numbers
FOR_SAMP	BMSAMP.FOR	1-174
	BXSAMP.FOR	175-314
	BMBFOR.FOR	315-417
	BMBRAN.FOR	418-546
	BXBFOR.FOR	547-646
	BXBRAN.FOR	647-774
FOR_SUB	BMGETC.FOR	1-180
	BXGETC.FOR	181-312
	BXCGTC.FOR	313-374
	BMUTIL.FOR	375-676
	BXUTIL.FOR	677-781
	CALUTI.FOR	782-1444
FOR_INCL	BMINCL.TXT	1-142
	BXINCL.TXT	143-212
	CALINC.TXT	213-285
	BMBPRM.TXT	286-312
	BXBPRM.TXT	313-338
C_SAMP	BMC_READ_SEQ.C	1-106
	BMC_READ_RAND.C	107-229
	BXC_READ_SEQ.C	230-330
	BXC_READ_RAND.C	331-450
	BMB_READ_SEQ.C	451-557
	BMB_READ_RAND.C	558-680
	BXB_READ_SEQ.C	681-781
	BXB_READ_RAND.C	782-904
	BMC_BMB_CONV.C	905-1019
	BXC_BXB_CONV.C	1020-1141

C_SUB	BMC_ACCESS.C	1-824
	BXC_ACCESS.C	825-1419
	BMB_ACCESS.C	1420-2193
	BXB_ACCESS.C	2194-2785
	BXCALC_ACCESS.C	2786-2862
	BXCALB_ACCESS.C	2863-2956
	BM_UTIL.C	2957-3256
	BX_UTIL.C	3257-3318
	BXCAL_UTIL.C	3319-3824
	BMB_WRITE.C	3825-4281
	BXB_WRITE.C	4282-4556
	BXCALB_WRITE.C	4557-4642
	FILE_FNCTS.C	4643-4873
	BXBRDK.C	4874-5032
	BXBRDK.C	5033-5168
	BXBCAL.C	5169-5249
C_INCL	BND_STRUCT.H	1-156
	BND_CONST.H	157-281

Modify the include statements and open statements in the programs and subroutines according to your system and compiler. Compile the subroutines and include them in an object library. We suggest creating separate libraries for FORTRAN and C sources.

Compile the sample programs and link them with the libraries. The sample programs can be modified to meet your requirements.

Copy all the data files from the CD to disk.

Run `bmc_bmb_conv` to create the binary calendar file and the binary master file. Provide as parameter the path of the directory of the data files.

Run `bxc_bxb_conv` to create the binary cross-sectional file. Provide as parameter the path of the directory of the data files.

Modifications for Unix

The file input/output and the compatibility between FORTRAN and C must be changed on some systems. The following changes should be made to CRSP programs to run the code provided on a SunOS Unix system.

Replace header `files <unixio.h> and <file.h> with <fcntl.h>`

Add `"#define SEEK_SET 0"`

If `<fcntl.h>` does not exist in your system, modify the open and `lseek` system calls in `file_fncts.c` to remove VMS-specific options, for example

```
—... if ((fdes = open(buf, 0)) == -1) { ...
—... if ((ret_index = lseek(fdes,offset,0)) != offset) { ...
```

Files that are created by binary conversion will need permissions set with the `chmod` command.

For the C functions called by FORTRAN (the `bmbrdk.c` and `bxbrdk.c` files) you should modify the name of the functions by adding a "_" at the very end of the name. For example, `bmbclo_()` will become `bmbclo_()`. This is because the FORTRAN compiler adds a '_' at the end of the name of a function in the library.

For the FORTRAN source calling the C functions, take all %REF out from the passed parameters.

APPENDICES

A. SPECIAL ISSUES

A.1 Issues with Special Provisions

The following is a list of issues having special provisions and coded with `ITYPE` = 9. You may wish to consider these provisions before using the data from these issues.

19330315.902000 Redeemable at option of holder at par plus accrued interest with 60 days notice. Principal and interest payable in United States gold coin.

19340415.904250 Issue created by early call of 19381015.904250. Similar numbers selected to be called for redemption on 19340415 were promulgated by the Treasury effectively creating a new issue which was quoted separately up to the call date.

19341015.904250 Issue created by early call of 19381015.904250. Similar to 19340415.904250.

19350415.904250 Issue related by early call of 19381015.904250. Similar to 19340415.904250.

19381015.904250 Principal and interest payable in United States gold coin.

19451015.903250 Accrued interest at the rate of 4¼% up to 19341015 and at 3¼% thereafter.

19590801.904000 Issue created from 19610801.904000 (see below).

19600215.904000 Issue created from 19620815.904000 (see below).

19610801.904000 Redeemable at the option of the holder at par and accrued interest on August 1, 1959. Notice of intent to redeem must be made by May 1, 1959 and certificates to be redeemed to be stamped. Once stamped, certificates mature on August 1, 1959 (not August 1, 1961 as issued). These stamped certificates were traded and quoted under the new `CRSPID`, even though no such security was actually issued by the treasury.

19620815.904000 Similar to 19600801.90400. Redeemable at option of holder on February 15, 1960, written notice and surrender required on or before November 16, 1959. Issue thus created was 19600215.904000.

99990401.902000 Consol bond, paid interest quarterly in perpetuity. Principal returned only if called. Issue actually called in 1935.

A.2 Stripped Notes and Bonds

Stripped notes and bonds are issues, which have been broken into their component cash flows, each of which is then traded separately. This was originally done by various financial institutions who issued treasury backed securities (e.g., CATS, TIGERS etc.). A fully-constituted Treasury note of bond consists of a principal payment and semiannual interest payments. In 1985 the treasury began participating in this market by designating certain issues as eligible to be stripped. All 10 year notes and all bonds issued since November 15, 1984 have been made eligible for the STRIPS program either upon their original issue or after their first interest payment date. Issues so designated could be broken up and the individual cash flows registered separately. As of September 1997, All new Treasury marketable fixed-rate notes and bonds issued on and after September 30, 1997 are eligible for STRIPS. The Treasury itself did not sell the individual payments, this being done by dealers who first purchased eligible securities.

The following issues have been designated as eligible for stripping by the Treasury:

19941115.211620	20010815.207870	20150815.110620
19950215.211250	20011115.207500	20151115.109870
19950515.211250	20020515.207500	20160215.109250
19950815.210500	20020815.206370	20160515.107250
19951115.209500	20020930.205870	20161115.107500
19960215.208870	20021031.205750	20170515.108750
19960515.207370	20021130.205750	20170815.108870
19961115.207250	20021231.205620	20180515.109120
19970515.208500	20030215.206250	20181115.109000
19970815.208620	20030815.205750	20190215.108870
19971115.208870	20040215.205870	20190815.108120
19980215.208120	20040515.207250	20200215.108500
19980515.209000	20040815.207250	20200515.108750
19980815.209250	20041115.111620	20200815.108750
19981115.208870	20041115.207870	20210215.107870
19990215.208870	20050215.207500	20210515.108120
19990515.209120	20050515.112000	20210815.108120
19990815.208000	20050515.206500	20211115.108000
19990930.205750	20050815.110750	20220815.107250
19991031.205620	20050815.206500	20221115.107620
19991115.207870	20051115.205870	20230215.107120
19991130.205620	20060215.109370	20230815.106250
19991231.205620	20060515.206870	20241115.107500
20000215.208500	20060715.207000	20250215.107620
20000515.208870	20061015.206500	20250815.106870
20000815.208750	20060215.205620	20260215.106000
20001115.205750	20070215.206250	20260815.106750
20001115.208500	20070515.206620	20261115.106500
20011115.208500	20070815.206120	20270215.106620
20011115.207500	20141115.511750	20270815.106370
20010215.207750	20150215.111250	20271115.106120
20010515.208000		

These issues are also traded as normal notes and bonds and are quoted as such in the files.

A.3 Foreign Targeted Securities

Foreign targeted issues are not included in the US Government Bond Files. Certain recent notes have been issued in pairs with identical coupon rates, maturities and dated dates. One issue of the pair is intended for domestic holders and is normal in all respects. The other issue is intended for United States aliens. These "Foreign Targeted Securities" are exempt from certain federal taxes when held by eligible foreigners. They pay interest annually and may be converted into their domestic equivalent or sale to domestic holders. The converse is not true.

The following notes which are included are known to have Foreign Targeted equivalents:

19880930.211370	dated 841031
19900215.211000	dated 841203
19900815.209870	dated 850604
19960215.208870	dated 860215

A.4 Inflation-Indexed Notes

Following is a list of inflation-indexed notes issued for the first time in 1997 by the US Treasury Department. The interest rate, which is set a auction, remains fixed throughout the term of the security. The principal amount of the security will be adjusted for inflation by using the non-seasonally adjusted CPI-U rate, but the inflation-adjusted principal will not be paid until maturity. Semiannual interest payments will be based on the inflation-adjusted principal at the time the interest is paid. Related information can be found on the US Treasury web page at http://www.publicdebt.treas.gov/of/of298pr. CRSP calculates real yields on these issues which excludes any increased payments due to inflation adjustments.

CRSPID	Dated	Maturity Date
20020715.003620	July 15, 1997	July 15, 2002
20070115.003370	January 15, 1997	January 15, 2007

B. LISTING OF PROGRAMS

The following pages contain the listings for the bond sample programs and include files provided. The numbers before each line are part of this documentation and not part of the programs -- they are provided to facilitate reference to the code.

B.1 FORTRAN Sample Programs

BMSAMP — Read Character Master Files Sequentially

```
 1
 2          *
 3          *
 4          *     PROGRAM BMSAMP READS THE CHARACTER CALENDAR AND DAILY BOND MASTER
 5          *     FILES SEQUENTIALLY.
 6          *
 7          *     BMSAMP CALLS SUBROUTINES BXCGTC TO READ THE CHARACTER CALENDAR FILE
 8          *     AND BMGETC TO READ THE CHARACTER BM* STRUCTURES.
 9          *
10          *     THE FOLLOWING FILES, LOGICAL UNIT NUMBERS ARE USED HERE:
11          *
12          *     INPUT:
13          *         BMHEADER.DAT  IUNIT1 = 20
14          *         BMQUOTES.DAT  IUNIT2 = 21
15          *         BMYIELD.DAT   IUNIT3 = 22
16          *         BMDEBT.DAT    IUNIT4 = 23
17          *         BMPAYMTS.DAT  IUNIT5 = 24
18          *         BXCALIND.DAT  IUNIT6 = 25
19          *
20          *     THEY ARE DEFINED IN THE COMMON BLOCK UNITS AND SHOULD BE
21          *     ASSIGNED IN THE PROGRAM
22          *
23
24          C
25          C     DECLARE PARAMETERS AND COMMON BLOCKS
26          C
27
28                INCLUDE 'CRSP:BNDDLY_INCLUDE(BMINCL)'
29
30          C
31          C     NREC         NUMBER OF RECORDS READ IN
32          C
33                INTEGER NREC
34
35          C
36          C     ASSIGN THE UNITS NUMBERS
37          C
38
39                IUNIT1 = 20
40                IUNIT2 = 21
41                IUNIT3 = 22
42                IUNIT4 = 23
43                IUNIT5 = 24
44                IUNIT6 = 25
45
46          C
47          C     OPEN CALENDAR AND DATA FILES
48          C
49
50                OPEN (UNIT=IUNIT1,
51               .      FILE='CRSP:BMHEADER.DAT',STATUS='OLD',
52               .      ACCESS='SEQUENTIAL',
53               .      FORM='FORMATTED',READONLY,RECORDTYPE = 'STREAM_LF',
54               .       RECL=155,ERR=980)
55
56                OPEN (UNIT=IUNIT2,
57               .      FILE='CRSP:BMQUOTES.DAT',STATUS='OLD',
```

```
58                .      ACCESS='SEQUENTIAL',
59                .      FORM='FORMATTED',READONLY,RECORDTYPE = 'STREAM_LF',
60                .       ERR=981)
61
62                 OPEN (UNIT=IUNIT3,
63                .      FILE='CRSP:BMYIELD.DAT',STATUS='OLD',
64                .      ACCESS='SEQUENTIAL',
65                .      FORM='FORMATTED',READONLY,RECORDTYPE = 'STREAM_LF',ERR=982)
66
67                 OPEN (UNIT=IUNIT4,
68                .      FILE='CRSP:BMDEBT.DAT',STATUS='OLD',
69                .      ACCESS='SEQUENTIAL',
70                .      FORM='FORMATTED',READONLY,RECORDTYPE = 'STREAM_LF',ERR=983)
71
72                 OPEN (UNIT=IUNIT5,
73                .      FILE='CRSP:BMPAYMTS.DAT',STATUS='OLD',
74                .      ACCESS='SEQUENTIAL',
75                .      FORM='FORMATTED',READONLY,RECORDTYPE = 'STREAM_LF',ERR=984)
76
77                 OPEN (UNIT=IUNIT6,
78                .      FILE='CRSP:BXCALIND.DAT',STATUS='OLD',
79                .      ACCESS='SEQUENTIAL',
80                .      FORM='FORMATTED',READONLY,RECORDTYPE = 'STREAM_LF',ERR=985)
81
82           C
83           C     READ THE CHARACTER CALENDAR FILE
84           C
85
86                 CALL BXCGTC ()
87
88                 CLOSE(IUNIT6)
89
90           C
91           C     PROCESS BM* STRUCTURES SEQUENTIALLY.
92           C
93
94                 NREC = 0
95
96           C
97           C     READ THE BM* STRUCTURE
98           C       BMGETC()
99           C       998 - SUCCESFULLY REACHED END OF FILE
100          C       993 - AN ERROR OCCURED
101          C
102
103            100 CALL BMGETC(*998,*993)
104
105                NREC = NREC + 1
106
107          C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
108          C     THE BM* STRUCTURES HAVE BEEN READ. INSERT YOUR CODE HERE.        C
109          C     SAMPLE CODE WRITES CRSPID, NAME, DATDT, FIRST QUOTE DATE, LAST   C
110          C     QUOTE  DATE TO TERMINAL                                          C
111          C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
112
113          C
114          C     PRINT CRSPID, NAME, DATDT, QDATE(FSTQUO) - QDATE(LSTQUO)
115          C
116
117                WRITE(*,900) CRSPID, NAME, DATDT, QDATE(FSTQUO), QDATE(LSTQUO)
118            900 FORMAT(/1X,A15,1X,A8,1X,I8,1X,I8,' - ',I8)
119                GO TO 100
120          C
121          C     ERRORS
122          C
123
124            980 WRITE(*,1001)
125           1001 FORMAT(' ERROR. BMSAMP. COULD NOT OPEN BMHEADER.')
126                STOP
127
128            981 WRITE(*,1002)
```

```
129            1002 FORMAT(' ERROR. BMSAMP. COULD NOT OPEN BMQUOTES.')
130                 STOP
131
132             982 WRITE(*,1003)
133            1003 FORMAT(' ERROR. BMSAMP. COULD NOT OPEN BMYIELD.')
134                 STOP
135
136             983 WRITE(*,1004)
137            1004 FORMAT(' ERROR. BMSAMP. COULD NOT OPEN BMDEBT.')
138                 STOP
139
140             984 WRITE(*,1005)
141            1005 FORMAT(' ERROR. BMSAMP. COULD NOT OPEN BMPAYMTS.')
142                 STOP
143
144             985 WRITE(*,1006)
145            1006 FORMAT(' ERROR. BMSAMP. COULD NOT OPEN BXCALIND.')
146                 STOP
147
148
149             993 WRITE(*,1007) NREC
150            1007 FORMAT(' ERROR. BMSAMP. NUMBER OF BM STRUCTURES READ = ', I6)
151                 CLOSE(IUNIT1)
152                 CLOSE(IUNIT2)
153                 CLOSE(IUNIT3)
154                 CLOSE(IUNIT4)
155                 CLOSE(IUNIT5)
156                 STOP
157
158           C
159           C     END OF FILE CLOSE THE DATA FILES AND REPORT THE NUMBER OF
160           C     STRUCTURES READ
161           C
162
163              998 CLOSE(IUNIT1)
164                 CLOSE(IUNIT2)
165                 CLOSE(IUNIT3)
166                 CLOSE(IUNIT4)
167                 CLOSE(IUNIT5)
168                 WRITE(*,1009) NREC
169            1009 FORMAT(' END OF FILE. NUMBER OF BM* STRUCTURES READ = ', I6)
170                 STOP
171
172                 END
```

BXSAMP - Read Character Cross-Sectional Files Sequentially

PROGRAM *BXSAMP*

```
1
2           *
3           *
4           *     PROGRAM BXSAMP READS THE CHARACTER CALENDAR AND DAILY BOND CROSS
5           *     SECTIONAL FILES.
6           *
7           *     BXSAMP CALLS SUBROUTINES BXCGTC TO READ THE CHARACTER CALENDAR FILE
8           *     AND BXGETC TO READ THE CHARACTER BX* STRUCTURES.
9           *
10          *     THE FOLLOWING FILES, LOGICAL UNIT NUMBERS ARE USED HERE:
11          *
12          *     INPUT:
13          *         BXCALIND.DAT  IUNIT6 = 25
14          *         BXQUOTES.DAT  IUNIT7 = 26
15          *         BXYIELD.DAT   IUNIT8 = 27
16          *
17          *     THEY ARE DEFINED IN THE COMMON BLOCK BXUNITS AND SHOULD BE
18          *     ASSIGNED IN THE PROGRAM
19          *
20
21          C
22          C     DECLARE PARAMETERS AND COMMON BLOCKS
23          C
24
25                INCLUDE 'CRSP:BNDDLY_INCLUDE(BXINCL)'
26
27          C
28          C     NREC         NUMBER OF RECORDS READ IN
29          C
30                INTEGER NREC
31
32          C
33          C     ASSIGN THE UNITS NUMBERS
34          C     NREC         NUMBER OF RECORDS READ IN
35          C
36
37                IUNIT6 = 28
38                IUNIT7 = 26
39                IUNIT8 = 27
40
41          C
42          C     OPEN CALENDAR AND DATA FILES
43          C
44
45                OPEN (UNIT=IUNIT6,
46               .      FILE='CRSP:BXCALIND.DAT',STATUS='OLD',
47               .      ACCESS='SEQUENTIAL',
48               .      FORM='FORMATTED',READONLY,RECORDTYPE = 'STREAM_LF',ERR=985)
49
50                OPEN (UNIT=IUNIT7,
51               .      FILE='CRSP:BXQUOTES.DAT',STATUS='OLD',
52               .      ACCESS='SEQUENTIAL',
53               .      FORM='FORMATTED',READONLY,RECORDTYPE = 'STREAM_LF',
54               .       ERR=986)
55
56                OPEN (UNIT=IUNIT8,
57               .      FILE='CRSP:BXYIELD.DAT',STATUS='OLD',
58               .      ACCESS='SEQUENTIAL',
59               .      FORM='FORMATTED',READONLY,RECORDTYPE = 'STREAM_LF',ERR=987)
60
61
62          C
63          C     READ THE CHARACTER CALENDAR FILE
64          C
65
66                CALL BXCGTC ()
```

```
 67
 68                CLOSE(IUNIT6)
 69
 70          C
 71          C     PROCESS BX* STRUCTURES SEQUENTIALLY.
 72          C
 73
 74                NREC = 0
 75
 76          C
 77          C     READ THE BX* STRUCTURE
 78          C       BXGETC()
 79          C       998 - SUCCESFULLY REACHED END OF FILE
 80          C       993 - AN ERROR OCCURED
 81          C
 82
 83                DO 100, I=1, NQDAT
 84                CALL BXGETC(QDATE(I), NUMACT(I), *998,*993)
 85
 86                NREC = NREC + 1
 87
 88          C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 89          C     THE BX* STRUCTURES HAVE BEEN READ. INSERT YOUR CODE HERE.        C
 90          C     SAMPLE CODE WRITES QDATE, NUMIDS AND THE FIRST AND LAST CRSPID   C
 91          C     TO TERMINAL                                                      C
 92          C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 93
 94          C
 95          C     PRINT QDATE(I), NUMACT(I), CRSPID(1), CRSPID(NUMACT(I))
 96          C
 97
 98                WRITE(*,900) QDATE(I),NUMACT(I),CRSPID(1),CRSPID(NUMACT(I))
 99            900 FORMAT(/1X,I8,1X,I5,1X,A15,1X,A15)
100            100 CONTINUE
101                WRITE(*,1011) NREC
102           1011 FORMAT(' NUMBER OF BX* STRUCTURES READ = ', I6)
103                STOP
104          C
105          C     ERRORS
106          C
107
108             985 WRITE(*,1006)
109            1006 FORMAT(' ERROR. BXSAMP. COULD NOT OPEN BXCALIND.DAT.')
110                 STOP
111
112             986 WRITE(*,1007)
113            1007 FORMAT(' ERROR. BXSAMP. COULD NOT OPEN BXQUOTES.DAT.')
114                 STOP
115
116             987 WRITE(*,1008)
117            1008 FORMAT(' ERROR. BXSAMP. COULD NOT OPEN BXYIELD.DAT.')
118                 STOP
119
120
121             993 WRITE(*,1009) NREC
122            1009 FORMAT(' ERROR. BXSAMP. NUMBER OF BX STRUCTURES READ = ', I6)
123                 STOP
124
125          C
126          C     END OF FILE CLOSE THE DATA FILES AND REPORT THE NUMBER OF
127          C     STRUCTURES READ
128          C
129
130          998   CLOSE(IUNIT6)
131                CLOSE(IUNIT7)
132                CLOSE(IUNIT8)
133
134                WRITE(*,1010) NREC
135          1010  FORMAT(' END OF FILE. NUMBER OF BX* STRUCTURES READ = ', I6)
136
137                STOP
```

```
138                 END
```

BMBFOR — Read Binary Master Files Sequentially

```
 1
 2          *
 3          *
 4          *     PROGRAM BMBFOR READS THE BINARY CALENDAR AND DAILY BOND MASTER
 5          *     BINARY FILES SEQUENTIALLY USING C ACCESS FUNCTIONS.
 6          *
 7          *     BMBFOR CALLS SUBROUTINES BXBCAL TO READ THE BINARY CALENDAR FILE
 8          *     AND BMBRDK TO READ THE BM* STRUCTURES. IT ALSO CALLS BMBOPE TO
 9          *     OPEN THE FILES AND LOAD THE INDEX AND BMBCLO TO CLOSE THE FILES.
10          *
11
12          C
13          C     DECLARE PARAMETERS AND COMMON BLOCKS
14          C
15
16                INCLUDE 'CRSP:BNDDLY_INCLUDE(BMINCL)'
17                INCLUDE 'CRSP:BNDDLY_INCLUDE(BMBPRM)'
18          C
19          C     NREC         NUMBER OF RECORDS READ IN
20          C     RET           THE RETURN CODE FROM THE C FUNCTIONS
21          C     WANTED        THE DESIRED INFORMATION; SHOULD BE
22          C                           QUOTES, YIELDS,  PAYMTS, DEBTS, ALLBM
23          C     BNDPAT        THE PATH WHERE BOND FILES ARE LOCATED
24          C     BNDLEN        THE DIMENSION OF BNDPATH
25          C     KEY           THE KEY FOR DIRECT ACCESS; SHOULD BE A CRSPID OR
26          C                    MFIRST, MPREV, MLAST, MSAME, MNEXT
27          C     MODE         THE MODE FOR OPENING THE FILES; SHOULD BE 'R' OR 'W'
28          C
29                INTEGER NREC, RET
30                INTEGER*4 WANTED, BNDLEN
31                CHARACTER BNDPAT*60, KEY*15, MODE*1
32
33                NREC   = 0
34                BNDPAT = 'CRSP:'
35                BNDLEN = 5
36                WANTED = ALLBM
37                KEY    = MNEXT
38                MODE   = 'R'
39                CALL VAXC$CRTL_INIT
40
41          C
42          C     OPEN CALENDAR AND DATA FILES
43          C
44
45                CALL BMBOPE(%REF(BNDPAT), BNDLEN, WANTED, %REF(MODE), RET)
46                IF (RET.EQ.-1) GOTO 993
47
48          C
49          C     READ THE BINARY CALENDAR FILE
50          C
51
52                CALL BXBCAL (QDATE, NQDAT, %REF(BNDPAT), BNDLEN,RET)
53                IF (RET.EQ.-1) GOTO 993
54
55
56          C
57          C     PROCESS BM* STRUCTURES SEQUENTIALLY.
58          C
59          C     READ THE BM* STRUCTURE
60          C       BMBRDK()
61          C       -2  - SUCCESFULLY REACHED END OF FILE
62          C       -1  - AN ERROR OCCURED
63          C
64
65              100 CALL BMBRDK(%REF(CRSPID), BID, ACCINT,
66                 .      DQDATE, PQDATE, %REF(MNEXT), WANTED, RET)
67                IF (RET .EQ. -1) THEN
```

```
 68               GOTO 993
 69                 ELSE IF (RET .EQ. -2) THEN
 70               GOTO 998
 71                 ELSE
 72                       NREC = NREC + 1
 73
 74           C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 75           C     THE BM* STRUCTURES HAVE BEEN READ. INSERT YOUR CODE HERE.        C
 76           C     SAMPLE CODE WRITES CRSPID, NAME, DATDT, FIRST QUOTE DATE, LAST   C
 77           C     QUOTE  DATE TO TERMINAL                                          C
 78           C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 79
 80
 81                WRITE(*,900) CRSPID, NAME, DATDT, QDATE(FSTQUO), QDATE(LSTQUO)
 82              900 FORMAT(/1X,A15,1X,A8,1X,I8,1X,I8,' - ',I8)
 83                GO TO 100
 84                END IF
 85
 86           C
 87           C     ERRORS
 88           C
 89
 90                993 WRITE(*,1007) NREC
 91               1007 FORMAT(' ERROR. BMBFOR. NUMBER OF BM STRUCTURES READ = ', I6)
 92                CALL BMBCLO(BNDPAT, RET)
 93                STOP
 94
 95                998 WRITE(*,1008) NREC
 96               1008 FORMAT(' BMBFOR. NUMBER OF BM STRUCTURES READ = ', I6)
 97                CALL BMBCLO(BNDPAT, RET)
 98                STOP
 99
100                END
```

BMBRAN — Read Binary Master Files Randomly

```
1
2          *
3          *
4          *     PROGRAM BMBRAN READS THE BINARY CALENDAR AND DAILY BOND MASTER
5          *     BINARY FILES RANDOMLY USING C ACCESS FUNCTIONS.
6          *     THE DESIRED CRSPIDS ARE READ FROM AN INPUT FILE
7          *     BMBFOR CALLS SUBROUTINES BXBCAL TO READ THE BINARY CALENDAR FILE
8          *     AND BMBRDK TO READ THE BM* STRUCTURES. IT ALSO CALLS BMBOPE TO
9          *     OPEN THE FILES AND LOAD THE INDEX AND BMBCLO TO CLOSE THE FILES.
10         *
11
12         C
13         C     DECLARE PARAMETERS AND COMMON BLOCKS
14         C
15
16               INCLUDE 'CRSP:BNDDLY_INCLUDE(BMINCL)'
17               INCLUDE 'CRSP:BNDDLY_INCLUDE(BMBPRM)'
18         C
19         C     NREC         NUMBER OF RECORDS READ IN
20         C     RET           THE RETURN CODE FROM THE C FUNCTIONS
21         C     WANTED        THE DESIRED INFORMATION; SHOULD BE
22         C                           QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM
23         C     BNDPAT        THE PATH WHERE BOND FILES ARE LOCATED
24         C     BNDLEN        THE DIMENSION OF BNDPATH
25         C     KEY           THE KEY FOR DIRECT ACCESS; SHOULD BE A CRSPID OR
26         C                    MFIRST, MPREV, MLAST, MSAME, MNEXT
27         C     MODE         THE MODE FOR OPENING THE FILES; SHOULD BE 'R' OR 'W'
28         C     INPFIL        THE NAME OF THE INPUT FILE
29         C     INPUNI        THE UNIT NUMBER FOR THE INPUT FILE
30         C
31               INTEGER NREC, RET
32               INTEGER*4 WANTED, BNDLEN
33               CHARACTER BNDPAT*60, KEY*15, MODE*1, INPFIL*32
34
35               NREC = 0
36               BNDPAT = 'CRSP:'
37               BNDLEN = 5
38               INPFIL = 'INPCRSPID.DAT'
39               INPUNI = 20
40               WANTED = ALLBM
41               MODE   = 'R'
42               CALL VAXC$CRTL_INIT
43
44         C
45         C     OPEN THE INPUT FILE
46         C
47               OPEN (UNIT=INPUNI,
48              .      FILE=INPFIL, STATUS='OLD',
49              .      ACCESS='SEQUENTIAL',
50              .      FORM='FORMATTED',READONLY,
51              .       ERR=980)
52
53
54         C
55         C     OPEN CALENDAR AND DATA FILES
56         C
57               CALL BMBOPE(%REF(BNDPAT), BNDLEN, WANTED, %REF(MODE), RET)
58               IF (RET.EQ.-1) GOTO 993
59
60         C
61         C     READ THE BINARY CALENDAR FILE
62         C
63
64               CALL BXBCAL (QDATE, NQDAT, %REF(BNDPAT), BNDLEN, RET)
65               IF (RET.EQ.-1) GOTO 993
66
```

```
 67
 68          C
 69          C     READ THE INPUT FILE SEQUENTIALLY.
 70          C
 71            100 READ (INPUNI,120, END=199)KEY
 72            120 FORMAT(A15)
 73
 74          C
 75          C     READ THE BM* STRUCTURE
 76          C       BMBRDK()
 77          C       -2  - SUCCESFULLY REACHED END OF FILE
 78          C       -1  - AN ERROR OCCURED
 79          C
 80
 81                CALL BMBRDK(%REF(CRSPID), BID, ACCINT,
 82               .      DQDATE, PQDATE, %REF(KEY), WANTED, RET)
 83                IF (RET .EQ. -1) THEN
 84              GOTO 993
 85                ELSE IF (RET .EQ. -2) THEN
 86              GOTO 998
 87                ELSE
 88                        NREC = NREC + 1
 89
 90          C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 91          C     THE BM* STRUCTURES HAVE BEEN READ. INSERT YOUR CODE HERE.        C
 92          C     SAMPLE CODE WRITES CRSPID, NAME, DATDT, FIRST QUOTE DATE, LAST   C
 93          C     QUOTE  DATE TO TERMINAL                                          C
 94          C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 95
 96                WRITE(*,900) CRSPID, NAME, DATDT, QDATE(FSTQUO), QDATE(LSTQUO)
 97              900 FORMAT(/1X,A15,1X,A8,1X,I8,1X,I8,' - ',I8)
 98                GO TO 100
 99                END IF
100
101              993 WRITE(*,1007) NREC
102             1007 FORMAT(' ERROR. BMBRAN. NUMBER OF BM STRUCTURES READ = ', I6)
103                CALL BMBCLO(BNDPAT,RET)
104                CLOSE (INPUNI)
105                STOP
106
107              998 WRITE(*,1008) NREC
108             1008 FORMAT(' BMBRAN. NUMBER OF BM STRUCTURES READ = ', I6)
109                CALL BMBCLO(BNDPAT,RET)
110                CLOSE (INPUNI)
111                STOP
112
113          C
114          C     ERRORS
115          C
116              980 WRITE(*,1001)
117             1001 FORMAT(' ERROR. BMSAMP. COULD NOT OPEN INPUT FILE.')
118                STOP
119
120              199 WRITE(*,1009) NREC
121             1009 FORMAT(' BMBRAN. NUMBER OF BM STRUCTURES READ = ', I6)
122                CALL BMBCLO(BNDPAT,RET)
123                CLOSE(INPUNI)
124                STOP
125
126                END
```

BXBFOR — Read Binary Cross-Sectional Files Sequentially

```
 1
 2          *
 3          *
 4          *     PROGRAM BXBFOR READS THE BINARY CALENDAR AND DAILY BOND CROSS-
 5          *     SECTIONAL BINARY FILES SEQUENTIALLY USING C ACCESS FUNCTIONS.
 6          *
 7          *     BXBFOR CALLS SUBROUTINE BXBRDK TO READ THE BX* STRUCTURES.
 8          *     IT ALSO CALLS BXBOPE TO
 9          *     OPEN THE FILES AND LOAD THE INDEX AND BXBCLO TO CLOSE THE FILES.
10          *
11
12          C
13          C     DECLARE PARAMETERS AND COMMON BLOCKS
14          C
15
16                INCLUDE 'CRSP:BNDDLY_INCLUDE(BXINCL)'
17                INCLUDE 'CRSP:BNDDLY_INCLUDE(BXBPRM)'
18          C
19          C     NREC         NUMBER OF RECORDS READ IN
20          C     RET           THE RETURN CODE FROM THE C FUNCTIONS
21          C     WANTED        THE DESIRED INFORMATION; SHOULD BE
22          C                           QUOTES, YIELDS, ALLBX
23          C     BNDPAT        THE PATH WHERE BOND FILES ARE LOCATED
24          C     BNDLEN        THE DIMENSION OF BNDPATH
25          C     KEY           THE KEY FOR DIRECT ACCESS; SHOULD BE A QDATE  OR
26          C                           XFIRST, XPREV, XLAST, XSAME, XNEXT
27          C     MODE         THE MODE FOR OPENING THE FILES; SHOULD BE 'R' OR 'W'
28          C
29                INTEGER NREC, RET
30                INTEGER*4 WANTED, KEY, BNDLEN
31                CHARACTER BNDPAT*60, MODE*1
32
33                NREC   = 0
34                BNDPAT = 'CRSP:'
35                BNDLEN = 5
36                WANTED = ALLBX
37                KEY    = XNEXT
38                MODE   = 'R'
39                CALL VAXC$CRTL_INIT
40
41          C
42          C     OPEN THE DATA FILES
43          C
44                CALL BXBOPE(%REF(BNDPAT), BNDLEN, WANTED, %REF(MODE), RET)
45                IF (RET.EQ.-1) GOTO 993
46
47          C
48          C     READ THE BINARY CALENDAR FILE
49          C
50
51                CALL BXBCAL (QDATE, NQDAT, %REF(BNDPAT), BNDLEN, RET)
52                IF (RET.EQ.-1) GOTO 993
53
54
55          C
56          C     PROCESS BX* STRUCTURES SEQUENTIALLY.
57          C
58          C     READ THE BX* STRUCTURE
59          C       BXBRDK()
60          C       -2  - SUCCESFULLY REACHED END OF FILE
61          C       -1  - AN ERROR OCCURED
62          C
63
64              100 CALL BXBRDK(XQDATE, BID, ACCINT, KEY, WANTED, RET)
65                  IF (RET .EQ. -1) THEN
66                GOTO 993
67                  ELSE IF (RET .EQ. -2) THEN
68                GOTO 998
```

```
69                ELSE
70                        NREC = NREC + 1
71
72              C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
73              C     THE BX* STRUCTURES HAVE BEEN READ. INSERT YOUR CODE HERE.        C
74              C     SAMPLE CODE WRITES THE QUOTE DATE, NUMBER OF ACTIVATIONS, FIRST  C
75              C     CRSPID AND LAST CRSPID TO TERMINAL                               C
76              C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
77
78                    WRITE(*,900) XQDATE, XNUM, CRSPID(1), CRSPID(XNUM)
79                900 FORMAT(/1X,I8,1X,I5,1X,A15,1X,A15)
80                    GOTO 100
81                    END IF
82                    STOP
83              C
84              C     ERRORS
85              C
86
87                998 WRITE(*,1008) NREC
88               1008 FORMAT(' BXBFOR. NUMBER OF BX STRUCTURES READ = ', I6)
89                    CALL BXBCLO(BNDPAT, RET)
90                    STOP
91
92                993 WRITE(*,1009) NREC
93               1009 FORMAT(' ERROR. BXBFOR. NUMBER OF BX STRUCTURES READ = ', I6)
94                    CALL BXBCLO(BNDPAT, RET)
95                    STOP
96
97                    END
```

BXBRAN - Read Binary Cross-Sectional Files Randomly

```
 1
 2          *
 3          *
 4          *     PROGRAM BXBRAN READS THE BINARY CALENDAR AND DAILY BOND CROSS-
 5          *     SECTIONAL BINARY FILES RANDOMLY USING C ACCESS FUNCTIONS.
 6          *
 7          *     BXBRAN CALLS SUBROUTINE BXBRDK TO READ THE BX* STRUCTURES.
 8          *     IT ALSO CALLS BXBOPE TO
 9          *     OPEN THE FILES AND LOAD THE INDEX AND BXBCLO TO CLOSE THE FILES.
10          *
11
12          C
13          C     DECLARE PARAMETERS AND COMMON BLOCKS
14          C
15
16                INCLUDE 'CRSP:BNDDLY_INCLUDE(BXINCL)'
17                INCLUDE 'CRSP:BNDDLY_INCLUDE(BXBPRM)'
18          C
19          C     NREC         NUMBER OF RECORDS READ IN
20          C     RET           THE RETURN CODE FROM THE C FUNCTIONS
21          C     WANTED        THE DESIRED INFORMATION; SHOULD BE
22          C                           QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM
23          C     BNDPAT        THE PATH WHERE BOND FILES ARE LOCATED
24          C     BNDLEN        THE DIMENSION OF BNDPATH
25          C     KEY           THE KEY FOR DIRECT ACCESS; SHOULD BE A QDATE OR
26          C                           XFIRST, XPREV, XLAST, XSAME, XNEXT
27          C     MODE         THE MODE FOR OPENING THE FILES; SHOULD BE 'R' OR 'W'
28          C     INPFIL        THE NAME OF THE INPUT FILE
29          C     INPUNI        THE UNIT NUMBER FOR THE INPUT FILE
30          C
31                INTEGER NREC, RET
32                INTEGER*4 WANTED, KEY, BNDLEN
33                CHARACTER BNDPAT*60, MODE*1, INPFIL*32
34
35                NREC = 0
36                BNDPAT = 'CRSP:'
37                BNDLEN = 5
38                INPFIL = 'INPDATE.DAT'
39                WANTED = ALLBX
40                MODE = 'R'
41                CALL VAXC$CRTL_INIT
42                INPUNI=20
43
44          C
45          C     OPEN THE INPUT FILE
46          C
47                OPEN (UNIT=INPUNI,
48               .      FILE=INPFIL,STATUS='OLD',
49               .      ACCESS='SEQUENTIAL',
50               .      FORM='FORMATTED',READONLY,
51               .       ERR=980)
52
53          C
54          C     OPEN THE DATA FILES
55          C
56
57                CALL BXBOPE(%REF(BNDPAT), BNDLEN, WANTED, %REF(MODE), RET)
58                IF (RET.EQ.-1) GOTO 993
59
60          C
61          C     READ THE BINARY CALENDAR FILE
62          C
63
64                CALL BXBCAL (QDATE, NQDAT, %REF(BNDPAT), BNDLEN, RET)
65                IF (RET.EQ.-1) GOTO 993
66
67          C
68          C     READ THE INPUT FILE SEQUENTIALLY.
```

```
 69           C
 70             100 READ (INPUNI,120, END=199)KEY
 71             120 FORMAT(I8)
 72
 73           C
 74           C     READ THE BX* STRUCTURE
 75           C       BXBRDK()
 76           C       -2  - SUCCESFULLY REACHED END OF FILE
 77           C       -1  - AN ERROR OCCURED
 78           C
 79
 80                 CALL BXBRDK(XQDATE,BID,ACCINT, KEY , WANTED,RET)
 81                 IF (RET .EQ. -1) THEN
 82               GOTO 993
 83                 ELSE IF (RET .EQ. -2) THEN
 84               GOTO 998
 85                 ELSE
 86                         NREC = NREC + 1
 87
 88           C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 89           C     THE BX* STRUCTURES HAVE BEEN READ. INSERT YOUR CODE HERE.        C
 90           C     SAMPLE CODE WRITES THE QUOTE DATE, NUMBER OF ACTIVATIONS, FIRST  C
 91           C     CRSPID AND LAST CRSPID TO TERMINAL                               C
 92           C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 93
 94                 WRITE(*,900) XQDATE, XNUM, CRSPID(1), CRSPID(XNUM)
 95               900 FORMAT(/1X,I8,1X,I5,1X,A15,1X,A15)
 96                 GOTO 100
 97                 END IF
 98
 99           C
100           C     ERRORS
101           C
102
103               998 WRITE(*,1008) NREC
104              1008 FORMAT(' BXBRAN. NUMBER OF BX STRUCTURES READ = ', I6)
105                 CALL BXBCLO(BNDPAT, RET)
106                 CLOSE(INPUNI)
107                 STOP
108
109               993 WRITE(*,1009) NREC
110              1009 FORMAT(' ERROR. BXBRAN. NUMBER OF BX STRUCTURES READ = ', I6)
111                 CALL BXBCLO(BNDPAT, RET)
112                 CLOSE(INPUNI)
113                 STOP
114
115               980 WRITE(*,1001)
116              1001 FORMAT(' ERROR. BXBRAN. COULD NOT OPEN INPUT FILE.')
117                 STOP
118
119               199 WRITE(*,1010) NREC
120              1010 FORMAT(' BXBRAN. NUMBER OF BX STRUCTURES READ = ', I6)
121                 CALL BXBCLO(BNDPAT, RET)
122                 CLOSE(INPUNI)
123                 STOP
124
125                 END
```

B.2 FORTRAN Include Files

BMINCL — FORTRAN Declarations For Master Files

```
      C
      C       CONTAINS ALL CONSTANTS AND VARIABLES DECLARATIONS FOR THE
      C       BONDS MASTER FILES
      C
            INCLUDE 'CRSP:BNDDLY_INCLUDE(CALINC)'

      C
      C     THE CONSTANT VALUES USED TO SET THE DIMENSIONS OF THE VECTORS IN
      C     THE COMMON BLOCKS.
      C

      C     MAXQUO    MAXIMUM NUMBER OF QUOTES
      C     MAXPAY    MAXIMUM NUMBER OF PAYMTS
      C     MAXDBT     MAXIMUM NUMBER OF DEBTS

            INTEGER MAXQUO,MAXPAY,MAXDBT

            PARAMETER (MAXQUO=10000,MAXPAY=100,MAXDBT=1000)

      *
      *     THE VARIABLE DECLARATIONS FOR COMMON BLOCK BMHEAD.
      *

      C     CRSPID       CRSP ISSUE IDENTIFICATION NUMBER
      C     CUSIP        CUSIP NUMBER
      C     NAME         NAME OF GOVERNMENT SECURITY
      C     MATDT        MATURITY DATE AT THE TIME OF ISSUE
      C     TYPE         TYPE OF ISSUE
      C     COUPRT       COUPON RATE (PER CENT PER ANNUM)
      C     UNIQ         UNIQUENESS NUMBER ASSIGNED TO CRSPID
      C     WHY          REASON FOR END OF DATA ON FILE
      C     DATDT        DATE DATED BY TREASURY
      C     BANKDT       BANK ELIGIBILITY DATE AT TIME OF ISSUE
      C     FCALDT       FIRST CALL DATE AT TIME OF ISSUE
      C     YMCNOT       YEAR AND MONTH OF FIRST CALL NOTICE
      C     NOTICE       NOTICE REQUIRED ON CALLABLE ISSUE
      C     TAX          TAXABILITY OF INTEREST
      C     FLOWER       PAYMENT OF ESTATE TAXES CODE
      C     NIPPY        NUMBER OF INTEREST PAYMENTS PER YEAR
      C     FCPDT        DATE OF FIRST COUPON PAYMENT
      C     FCPDTF        A FLAG FOR FCPDT 0 = POSITIVE, 1 = NEGA TIVE
      C     VALFC        AMOUNT OF FIRST COUPON PAYMENT
      C     NUMDBT       NUMBER OF DEBTS
      C     NUMPAY       NUMBER OF PAYMENTS
      C     FSTQUO       INDEX OF THE DATE OF THE FIRST QUOTE
      C     LSTQUO       INDEX OF THE DATE OF THE LAST QUOTE
      C     FSTYLD       INDEX OF THE DATE OF THE FIRST YIELD
      C     LSTYLD       INDEX OF THE DATE OF THE LAST YIELD


            COMMON /BMHEAD/ CRSPID,TYPE,MATDT,COUPRT,
           .                UNIQ,WHY,DATDT,BANKDT,FCALDT,YMCNOT,
           .                NOTICE,TAX,FLOWER,NIPPY,FCPDT,FCPDTF,VALFC,
           .                CUSIP,NAME,FSTQUO,LSTQUO,FSTYLD,
           .                 LSTYLD,NUMPAY,NUMDBT

            INTEGER MATDT,TYPE,UNIQ,WHY,
           .        DATDT,BANKDT,FCALDT,YMCNOT,NOTICE,TAX,FLOWER,NIPPY,
           .        FCPDT,FCPDTF,NUMDBT,NUMPAY,FSTQUO,LSTQUO,FSTYLD,LSTYLD

            REAL*8 COUPRT,VALFC

            CHARACTER CRSPID*16, CUSIP*8, NAME*8

      *
```

```
      *     THE VARIABLE DECLARATIONS FOR COMMON BLOCK  BMQUO.
      *
      C     BID          BID PRICE WHERE AVAILABLE
      C     ASK          ASK PRICE WHERE AVAILABLE
      C     SOURCE       PRIMARY DATA SOURCE

            COMMON /BMQUO/ BID(MAXQUO),ASK(MAXQUO),SOURCE(MAXQUO)

            REAL*8 BID, ASK

            CHARACTER SOURCE*1

      *
      *     THE VARIABLE DECLARATIONS FOR COMMON BLOCK BMYIELD.
      *
      C     ACCINT       TOTAL ACCRUED INTEREST AT END OF DAY
      C     YLD          PROMISED DAILY YIELD
      C     RETNUA       UNADJUSTED RETURN
      C     DURATN       DURATION
            COMMON /BMYLD/  ACCINT(MAXQUO), YLD(MAXQUO),RETNUA(MAXQUO),
           .                DURATN(MAXQUO)

            REAL*8 ACCINT,YLD,RETNUA,DURATN

      *
      *     THE VARIABLE DECLARATIONS FOR COMMON BLOCK BMDEBT.
      *

      C     DQDATE       DEBT QDATE
      C     TOTOUT       PAR VALUE OUTSTANDING
      C     PUBOUT       PAR VALUE PUBLICLY HELD

            COMMON /BMDEBT/ DQDATE(MAXDBT), TOTOUT(MAXDBT), PUBOUT(MAXDBT)

            INTEGER DQDATE, TOTOUT, PUBOUT

      *
      *     THE VARIABLE DECLARATIONS FOR COMMON BLOCK BMPAY.
      *

      C     PQDATE       PAYMENT QDATE
      C     PDINT        INTEREST PAYABLE DURING MONTH

            COMMON /BMPAY/ PQDATE(MAXPAY), PDINT(MAXPAY)

            INTEGER PQDATE

            REAL*8  PDINT

      C
      C     THE VARIABLE DECLARATIONS FOR COMMON BLOCK UNITS CONTAINING
      C     THE UNITS FOR THE OPENED FILES
      C
      C     IUNIT1         UNIT FOR THE BMHEADER.DAT FILE
      C     IUNIT2         UNIT FOR THE BMQUOTES.DAT FILE
      C     IUNIT3         UNIT FOR THE BMYIELD.DAT FILE
      C     IUNIT4         UNIT FOR THE BMDEBT.DAT FILE
      C     IUNIT5         UNIT FOR THE BMPAYMTS.DAT FILE

            COMMON /BMUNITS/IUNIT1, IUNIT2, IUNIT3, IUNIT4, IUNIT5

            INTEGER IUNIT1, IUNIT2, IUNIT3, IUNIT4, IUNIT5

      C
      C     DECLARATION OF THE UTILITY FUNCTIONS FOR THE MASTER FILES
      C
      C     IDBT (IDXCAL)     RETURNS AN INDEX IN THE DEBT STRUCTURE
      C     IPAY (IDXCAL)     RETURNS AN INDEX INTO THE PAYMENTS STRUCTURE
      C     FPDINT (IDXCAL)   CALCULATES THE PAYED INTEREST
      C     NTOUT (IDXCAL)    CALCULATES THE FACE VALUE OUTSTANDING
      C     NPOUT (IDXCAL)    CALCULATES THE PUBLICLY HELD FACE VALUE OUTSTANDING
```

```
138
139
140                INTEGER  IDBT,IPAY,NTOUT,NPOUT
141                REAL*8   FPDINT
```

BXINCL — FORTRAN Declarations For Cross-Sectional Files

```
 1
 2          C
 3          C       CONTAINS ALL CONSTANTS AND VARIABLES DECLARATIONS FOR THE
 4          C       BONDS CROSS-SECTIONAL FILES
 5          C
 6                INCLUDE 'CRSP:BNDDLY_INCLUDE(CALINC)'
 7
 8          C
 9          C     THE CONSTANT VALUES USED TO SET THE DIMENSIONS OF THE VECTORS
10          C     IN THE COMMON BLOCKS.
11          C
12
13          C     MAXIDS    MAXIMUM NUMBER OF ISSUES (CRSPID)
14
15                INTEGER MAXIDS
16
17                PARAMETER (MAXIDS=5500)
18
19          *
20          *     THE VARIABLE DECLARATIONS FOR COMMON BLOCK BXHEAD.
21          *
22          C     XQDATE    QUOTE DATE
23          C     XNUM      NUMBER OF ACTIVATIONS FOR XQDATE
24          C     CRSPID    CRSP ISSUE IDENTIFICATION NUMBER
25
26
27                COMMON /BXHEAD/XQDATE, XNUM,  CRSPID(MAXIDS)
28
29                INTEGER XQDATE, XNUM
30                CHARACTER CRSPID*16
31
32          *
33          *     THE VARIABLE DECLARATIONS FOR COMMON BLOCK BXQUO.
34          *
35          C     BID          BID PRICE WHERE AVAILABLE
36          C     ASK          ASK PRICE WHERE AVAILABLE
37          C     SOURCE       PRIMARY DATA SOURCE
38
39                COMMON /BXQUO/ BID(MAXIDS),ASK(MAXIDS),SOURCE(MAXIDS)
40
41                REAL*8 BID, ASK
42
43                CHARACTER SOURCE*1
44          *
45          *     THE VARIABLE DECLARATIONS FOR COMMON BLOCK BXYLD.
46          *
47          C     ACCINT       TOTAL ACCRUED INTEREST AT END OF DAY
48          C     YLD          PROMISED DAILY YIELD
49          C     RETNUA       UNADJUSTED RETURN
50          C     DURATN       DURATION
51
52                COMMON /BXYLD/  ACCINT(MAXIDS), YLD(MAXIDS),RETNUA(MAXIDS),
53               .                DURATN(MAXIDS)
54
55                REAL*8  ACCINT,YLD,RETNUA,DURATN
56          *
57          *     THE VARIABLE DECLARATIONS FOR COMMON BLOCK UNITS CONTAINING THE
58          *     UNITS FOR THE OPENED FILES
59          *
60          C     IUNIT7         UNIT FOR THE BXQUOTES.DAT FILE
61          C     IUNIT8         UNIT FOR THE BXYIELD.DAT FILE
62
63                COMMON /BXUNITS/IUNIT7, IUNIT8
64
65                INTEGER IUNIT7, IUNIT8
66          C
67          C     DECLARATION OF THE UTILITY FUNCTIONS FOR THE CROSS-SECTIONAL FILES
```

```
68            C
69            C     INDCID(CRSPID,CODE,ARRAY,MAXARR) RETURNS THE INDEX OF ARRAY
70
71                  INTEGER INDCID
```

CALINC — FORTRAN Declarations For Calendar File

```
1
2           C
3           C     THE INCLUDE FILE CALINC CONTAINS ALL CONSTANTS AND VARIABLES
4           C     DECLARATIONS FOR THE CALENDAR RELATED FUNCTIONS.
5           C
6
7
8           C
9           C     THE CONSTANT VALUES USED TO SET THE DIMENSIONS OF THE CALENDAR
10          C     VECTOR IN THE COMMON BLOCKS.
11          C
12          C     MAXCAL     MAXIMUM NUMBER OF DATES IN THE DAILY CALENDAR
13
14                INTEGER MAXCAL
15
16                PARAMETER (MAXCAL=10000)
17
18          *
19          *     THE VARIABLE DECLARATIONS FOR COMMON BLOCK UNITS CONTAINING
20          *     THE UNITS FOR THE CALENDAR FILES
21          *
22          C     IUNIT6          UNIT FOR THE BXCALIND.DAT FILE
23
24                COMMON /CALUNI/IUNIT6
25
26                INTEGER IUNIT6
27
28          *
29          *     THE VARIABLE DECLARATIONS FOR COMMON BLOCK BXCAL.
30          *
31
32          C     QDATE        CALENDAR DATE
33          C     DELDAT       DELIVERY DATE
34          C     CD1M         CERTIFICATE OF DEPOSIT RATE 1 MONTH
35          C     CD3M         CERTIFICATE OF DEPOSIT RATE 3 MONTHS
36          C     CD6M         CERTIFICATE OF DEPOSIT RATE 6 MONTHS
37          C     CP30D        COMMERCIAL PAPER RATE 30 DAYS
38          C     CP60D        COMMERCIAL PAPER RATE 60 DAYS
39          C     CP90D        COMMERCIAL PAPER RATE 90 DAYS
40          C     FFEFRT       FEDERAL FUNDS RATE (EFFECTIVE RATE)
41          C     FFMINR       FEDERAL FUNDS MINIMUM TRADING RANGE
42          C     FFMAXR       FEDERAL FUNDS MAXIMUM TRADING RANGE
43          C     NUMACT        NUMBER OF ACTIVITIONS
44          C     NQDAT         TOTAL NUMBER OF DATES IN THE CALENDAR
45
46                COMMON /BXCAL/ QDATE(MAXCAL),DELDAT(MAXCAL),CD1M(MAXCAL),
47               .               CD3M(MAXCAL), CD6M(MAXCAL),CP30D(MAXCAL),
48               .                CP60D(MAXCAL),
49               .               CP90D(MAXCAL), FFEFRT(MAXCAL),FFMINR(MAXCAL),
50               .                FFMAXR(MAXCAL),NUMACT(MAXCAL), NQDAT
51
52                INTEGER QDATE, DELDAT, NUMACT, NQDAT
53
54                REAL CD1M, CD3M, CD6M, CP30D, CP60D, CP90D, FFEFRT, FFMINR,
55               .     FFMAXR
56
57          C
58          C     DECLARATION OF THE UTILITY FUNCTIONS FOR THE CALENDAR
59          C
60          C    NDIFDT (IDAT1,IDAT2) RETURNS THE DIFFERENCE BETWEEN TWO DATES
61          C    NQDATE (IDXCAL)       CALCULATES DAY NUMBER OF QUOTATION DATE
62          C    NDZERO (IDXCAL)      CALCULATES DAY NUMBER OF ZERO'TH DAY OF THE MONTH
63          C    NDDATE (IDXCAL)      CALCULATES DAY NUMBER OF DELIVERY DATE
64          C    JAHRMO (IDXCAL)      CALCULATES YEAR AND MONTH(YYYYMM) OF QUOTE DATE
65          C    IQDAY (IDXCAL)       CALCULATES DAY AND MONTH OF QUOTATION DATE
66          C    NDHFYR (IDXCAL)      CALCULATES THE LINEAR NUMBER OF DATES IN A HALF
67          C                          YEAR
68          C    NQTOQD (IDXCAL)      CALCULATES THE NUMBER OF DAYS FROM THE LAST
```

```
69          C                          QUOTATION DATE TO THIS QUOTATION DATE
70          C    NFQDAT (IDXCAL)      CALCULATES THE QUOTATION DATE (YYMMDD)
71          C    INDCAL(DATE,CODE,ARRAY,MAXARR) RETURNS THE INDEX OF ARRAY OF DATES
72
73                  INTEGER NDIFDT,NQDATE,NDZERO,NDDATE,JAHRMO,IQDAY,NDHFYR,
74               .  NQTOQD,NFQDAT,INDCAL
```

BMBPRM — C Declarations For Master Files

```
 1
 2          C
 3          C  THIS FILE CONTAINS THE DEFINITIONS OF CONSTANTS USED IN DAILY BONDS
 4          C  ACCESS C FUNCTIONS CALLED BY FORTRAN FOR THE MASTER FILES
 5          C
 6
 7          C  SUBSCRIPT NAMES FOR RANDOM ACCESSS IN MASTER FILES
 8
 9          CHARACTER MFIRST*16, MPREV*16, MLAST*16, MSAME*16, MNEXT*16
10
11          PARAMETER (MFIRST = 'FIRST           ')
12          PARAMETER (MPREV  = 'PREV            ')
13          PARAMETER (MLAST  = 'LAST            ')
14          PARAMETER (MSAME  = 'SAME            ')
15          PARAMETER (MNEXT  = 'NEXT            ')
16
17          C  CONSTANTS FOR DESCRIBING THE WANTED INFORMATION
18
19          INTEGER QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM
20
21          PARAMETER (QUOTES = 1)
22          PARAMETER (YIELDS = 2)
23          PARAMETER (PAYMTS = 4)
24          PARAMETER (DEBTS  = 8)
25          PARAMETER (ALLBM  = 15)
```

BXBPRM — C Declarations For Cross-Sectional Files

```
 1
 2          C
 3          C  THIS FILE CONTAINS THE DEFINITIONS OF CONSTANTS USED IN DAILY BONDS
 4          C  ACCESS C FUNCTIONS CALLED BY FORTRAN FOR CROSS-SECTIONAL FILES
 5          C
 6
 7          C  SUBSCRIPT NAMES FOR RANDOM ACCESSS IN CROSS_SECTIONAL FILES
 8
 9          INTEGER XFIRST, XPREV, XLAST, XSAME, XNEXT
10
11          PARAMETER (XFIRST = -91)
12          PARAMETER (XPREV = -92)
13          PARAMETER (XLAST = -93)
14          PARAMETER (XSAME = -94)
15          PARAMETER (XNEXT = -95)
16
17
18          C  CONSTANTS FOR DESCRIBING THE WANTED INFORMATION
19
20          INTEGER QUOTES, YIELDS, ALLBX
21
22          PARAMETER (QUOTES = 1)
23          PARAMETER (YIELDS = 2)
24          PARAMETER (ALLBX  = 3)
```

B.3 C Sample Programs

bmc_read_seq — Read Character Master Files Sequentially

```
1
2           /*
3              Sample program to read sequentially the master bond character files
4              Must be run with a parameter bndpath = the path of the directory where
5              the daily bonds files are
6                                 */
7
8           #include <stdio.h>
9           #include "bnd_struct.h"
10
11          extern struct BXCAL     bx_cal; /* the calendar array */
12          extern int       nbx_cal; /*the number of records in the calendar file */
13
14          main (argc, argv)
15
16          int argc;
17          char *argv[];
18
19          {
20          struct BM_STRUCT bms; /* the bonds structure */
21          int ret; /* return code */
22          int wanted; /*the desired information; should be
23                          QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination */
24
25          /*
26             Check the arguments
27                               */
28
29          if (argc != 2)
30              {
31               fprintf(stderr,"Usage: %s bndpath\n", argv[0]);
32               exit (4);
33              }
34
35          /*
36              Set the wanted variable = the desired information; should be
37                          QUOTES, YIELDS, PAYMTS, DEBTS,  ALLBM or any combination
38                                  */
39          wanted = ALLBM;
40
41
42          /*
43             Open all data files and the  index(addresses) file
44                               */
45
46          if (bmc_open (argv[1],wanted,"R") == -1)
47              {
48               fprintf(stderr,"Error opening files\n");
49               exit (4);
50              }
51
52
53          /*
54             Load the calendar into the bx_cal array and initialize nbx_cal = the
55             number of dates
56                               */
57
58          if ((nbx_cal = bxc_cal_load(argv[1])) == -1)
59              {
60               fprintf(stderr,"Error loading the calendar\n");
61               exit (4);
62              }
63
64          /*
65             Main loop. Read sequentialy the bonds structures  till the end of files.
66             The function bmc_rdkey loads all wanted data in the bms structure  for
```

```
67              the next crspid.
68                                */
69
70           while ((ret = bmc_rdkey(&bms, MNEXT, wanted))!= EOFL) {
71
72           /*
73              Check if it is any error
74                                */
75
76               if (ret == -1)
77           {
78            fprintf(stderr,"Error loading a bond structure\n");
79            exit(4);
80                   }
81
82           /*
83              Now the structure is loaded. Insert your code here. The sample program
84              will print the crspid, name, datdt, first quote date, last quote date
85              to terminal
86                                */
87
88               printf("%15.15s %8.8s %8d %8d %8d\n", bms.bmhead.crspid,
89           bms.bmhead.name,bms.bmhead.datdt,
90           bx_cal.qdate[bms.bmhead.fstquo], bx_cal.qdate[bms.bmhead.lstquo]);
91
92             } /*end while*/
93
94
95           /*
96              Close the data files
97                                */
98           if (bmc_close (wanted) == -1)
99               {
100               fprintf(stderr,"Error closing files\n");
101               exit (4);
102              }
103          }
```

bmc_read_rand — Read Character Master Files Randomly

```
1
2          /* ---------------------------------------------------------------------
3             Sample program to read randomly the master character bond files. The
4             wanted crspids are read from a text file. The program  should be run
5             with two parameters:
6             bndpath     - the path of the directory where the daily bonds files are
7             inpfilename - the input file name(including the path)
8             --------------------------------------------------------------------- */
9
10         #include <stdio.h>
11         #include "bnd_struct.h"
12
13
14         extern struct BXCAL     bx_cal; /* the calendar array */
15         extern int      nbx_cal; /* The number of records in the calendar file */
16
17
18         main (argc, argv)
19
20         int argc;
21         char *argv[];
22
23         {
24         FILE               *finp; /* Pointer for the input file */
25         struct BM_STRUCT    bms;  /* The bonds master structure */
26         char                curcrspid[16]; /* The current wanted crspid */
27         int                 wanted; /*   the desired information; should be
28                                         QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM
29          or any combination */
30
31         /*
32            Check the arguments
33                               */
34
35         if (argc != 3)
36             {
37              fprintf(stderr,"Usage: %s bndpath inpfilename wanted\n",  argv[0]);
38              exit (4);
39             }
40
41
42         /*
43            Open the input file
44                               */
45
46         if ((finp = fopen (argv[2], "r"))== NULL)
47             {
48              fprintf(stderr,"Cannot open %s\n", argv[2]);
49              exit(4);
50             }
51
52         /*
53             Set the wanted variable = the desired information; should be
54                         QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination
55                               */
56         wanted = ALLBM;
57
58         /*
59            Open all data files and the index(addresses) file
60                               */
61
62         if (bmc_open (argv[1], wanted, "R") == -1)
63             {
64              fprintf(stderr,"Error opening files\n");
65              exit (4);
66             }
67
68         /*
```

```
69              Load the calendar into the bx_cal array and initialize nbx_cal= the number
70              of dates
71                                  */
72
73          if ((nbx_cal = bxc_cal_load(argv[1])) == -1)
74              {
75               fprintf(stderr,"Error loading the calendar\n");
76               exit (4);
77              }
78
79          /*
80              Read sequentialy the input file and call the bmc_rdkey for each read
81              crspid. The function bmc_rdkey loads all wanted data in the bms structure
82              for the desired crspid.
83                                  */
84           while (fscanf (finp, "%s", curcrspid) != EOF) {
85
86              if (bmc_rdkey(&bms, curcrspid , wanted)== -1)
87          {
88           fprintf(stderr,"Error loading the bond structure for %s\n",curcrspid);
89           exit(4);
90                 }
91
92          /*
93              Now the structure is loaded. Insert your code here. The sample program
94              will print the crspid, name, datdt, first quote date, last quote date
95              to terminal
96                                  */
97
98              printf("%15.15s %8.8s %8d %8d %8d\n", bms.bmhead.crspid,
99          bms.bmhead.name,bms.bmhead.datdt,
100         bx_cal.qdate[bms.bmhead.fstquo], bx_cal.qdate[bms.bmhead.lstquo]);
101
102           } /*end while*/
103
104         /*
105            Close the data files
106                                 */
107
108         if (bmc_close (wanted) == -1)
109             {
110              fprintf(stderr,"Error closing files\n");
111              exit (4);
112             }
113
114         /*
115            Close the input file
116                                 */
117
118         if (fclose (finp) == EOF)
119             {
120              fprintf(stderr,"Error closing the input file\n");
121              exit (4);
122             }
123         }
```

bxc_read_seq - Read Character Cross-Sectional Files Sequentially

```
1
2           /*
3              Sample program to read sequentially the character cross-sectional files
4              Must be run with a parameter bndpath = the path of the directory where
5              the daily bonds files are
6                                    */
7
8           #include <stdio.h>
9           #include "bnd_struct.h"
10
11          extern struct BXCAL     bx_cal; /* the calendar array */
12          extern int      nbx_cal; /*the number of records in the calendar file */
13
14
15          main (argc, argv)
16
17          int argc;
18          char *argv[];
19
20          {
21          struct BX_STRUCT bxs; /* the bonds structure */
22          char bndpath[80]; /* the path of the directory where the data files are */
23          int ret; /* return code */
24          int wanted; /*the desired information; should be
25                           QUOTES, YIELDS, ALLBX or any combination */
26
27          /*
28             Check the arguments
29                                   */
30
31          if (argc != 2)
32              {
33               fprintf(stderr,"Usage: %s bndpath\n", argv[0]);
34               exit (4);
35              }
36
37          strcpy (bndpath, argv[1]);
38
39          /*
40              Set the wanted variable = the desired information; should be
41                           QUOTES, YIELDS,  ALLBX or any combination
42                                   */
43          wanted = ALLBX;
44
45
46          /*
47             Open all data files and the index(addresses) file
48                                   */
49
50          if (bxc_open (bndpath,wanted,"R") == -1)
51              {
52               fprintf(stderr,"Error opening files\n");
53               exit (4);
54              }
55
56          /* load the calendar */
57          if ((nbx_cal = bxc_cal_load(bndpath)) == -1)
58              {
59               fprintf(stderr,"Error loading the calendar\n");
60               exit (4);
61              }
62
63          /*
64             Main loop. Read sequentialy the bonds structures till the end of files.
65             The function bxc_rdkey loads all wanted data in the bxs structure for
66             the next qdate.
67                                   */
68
```

```
        while ((ret = bxc_rdkey(&bxs, XNEXT, wanted))!= EOFL) {

        /*
           Check if it is any error
                                */

            if (ret == -1)
        {
         fprintf(stderr,"Error loading a bond structure\n");
         exit(4);
                }

        /*
           Now the structure is loaded. Insert your code here. The sample program
           will print the qdate, first crspid and last crspid
           to terminal
                                */

            printf("%8d %15.15s %15.15s\n", bxs.bxhead.qdate,
        bxs.bxquo.crspid[0], bxs.bxquo.crspid[bxs.bxhead.numact-1]);
          } /*end while*/


        /*
           Close the data files
                                */
        if (bxc_close (wanted) == -1)
            {
             fprintf(stderr,"Error closing files\n");
             exit (4);
            }
        }
```

bxc_read_rand — Read Character Cross-Sectional Files Randomly

```
1
2          /*
3             Sample program to read randomly the character  cross_sectional files.
4             The wanted qdates
5             are read from a text file. The program should be run with two parameters:
6             bndpath     - the path of the directory where the daily bonds files are
7             inpfilename - the input file name(including the path)
8
9                                 */
10
11         #include <stdio.h>
12         #include "bnd_struct.h"
13
14         extern struct BXCAL     bx_cal; /* the calendar array */
15         extern int      nbx_cal; /*the number of records in the calendar file */
16
17         main (argc, argv)
18
19         int argc;
20         char *argv[];
21
22         {
23         FILE               *finp;     /* Pointer for the input file */
24         struct BX_STRUCT    bxs;      /* The bonds structure */
25         char bndpath[80]; /* the path of the directory where the data files are */
26         int                 curqdate; /* The current wanted date */
27         int                 wanted;   /*   the desired information; should be
28                                          QUOTES, YIELDS, ALLBX
29          or any combination */
30
31         /*
32            Check the arguments
33                                */
34
35         if (argc != 3)
36             {
37              fprintf(stderr,"Usage: %s bndpath inpfilename wanted\n", argv[0]);
38              exit (4);
39             }
40
41         strcpy (bndpath, argv[1]);
42
43
44         /*
45            Open the input file
46                                */
47
48         if ((finp = fopen (argv[2], "r"))== NULL)
49             {
50              fprintf(stderr,"Cannot open %s\n", argv[2]);
51              exit(4);
52             }
53
54         /*
55             Set the wanted variable = the desired information; should be
56                         QUOTES, YIELDS, ALLBX or any combination
57                                */
58         wanted = ALLBX;
59
60         /*
61            Open all data files and the index(addresses) file
62                                */
63
64         if (bxc_open (bndpath, wanted,"R") == -1)
65             {
66              fprintf(stderr,"Error opening files\n");
67              exit (4);
68             }
```

```
69
70          /* load the calendar */
71          if ((nbx_cal = bxc_cal_load(bndpath)) == -1)
72              {
73               fprintf(stderr,"Error loading the calendar\n");
74               exit (4);
75              }
76
77          /*
78             Read sequentialy the input file and call the bxc_rdkey for each read
79             qdate. The function bxc_rdkey loads all wanted data in the bxs structure
80             for the desired qdate.
81                                 */
82           while (fscanf (finp, "%d", &curqdate) != EOF) {
83
84              if (bxc_rdkey(&bxs, curqdate , wanted)== -1)
85          {
86           fprintf(stderr,"Error loading the bond structure for %8d\n",curqdate);
87           exit(4);
88                  }
89
90          /*
91             Now the structure is loaded. Insert your code here. The sample program
92             will print the qdate, first crspid and last crspid
93             to terminal
94                                 */
95
96              printf("%8d %15.15s %15.15s\n", bxs.bxhead.qdate,
97          bxs.bxquo.crspid[0], bxs.bxquo.crspid[bxs.bxhead.numact-1]);
98
99            } /*end while*/
100
101
102         /*
103            Close the data files
104                                 */
105
106         if (bxc_close (wanted) == -1)
107             {
108              fprintf(stderr,"Error closing files\n");
109              exit (4);
110             }
111
112         /*
113            Close the input file
114                                 */
115
116         if (fclose (finp) == EOF)
117             {
118              fprintf(stderr,"Error closing the input file\n");
119              exit (4);
120             }
121         }
```

bmb_read_seq — Read Binary Master Files Sequentially

```
1
2          /*
3             Sample program to read sequentially the master bond binary files
4             Must be run with a parameter bndpath = the path of the directory where
5             the daily bonds files are
6                                 */
7
8          #include <stdio.h>
9          #include "bnd_struct.h"
10
11
12         extern struct BXCAL     bx_cal; /* the calendar array */
13         extern int      nbx_cal; /*the number of records in the calendar file */
14
15
16
17         main (argc, argv)
18
19         int argc;
20         char *argv[];
21
22         {
23         struct BM_STRUCT bms; /* the bonds structure */
24         int ret; /* return code */
25         int wanted; /*the desired information; should be
26                         QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination */
27
28         /*
29            Check the arguments
30                                */
31
32         if (argc != 2)
33             {
34              fprintf(stderr,"Usage: %s bndpath\n", argv[0]);
35              exit (4);
36             }
37
38         /*
39             Set the wanted variable = the desired information; should be
40                         QUOTES, YIELDS, PAYMTS, DEBTS,  ALLBM or any combination
41                                */
42         wanted = ALLBM;
43
44
45         /*
46            Open all data files and the index(addresses) file
47                                */
48
49         if (bmb_open (argv[1],wanted,"R") == -1)
50             {
51              fprintf(stderr,"Error opening files\n");
52              exit (4);
53             }
54
55         /*
56            Load the calendar into the bx_cal array and initialize nbx_cal= the
57            number of dates
58                                */
59
60         if ((nbx_cal = bxb_cal_load(argv[1])) == -1)
61             {
62              fprintf(stderr,"Error loading the calendar\n");
63              exit (4);
64             }
65
66
67         /*
68            Main loop. Read sequentialy the bonds structures till the end of files.
```

```
69              The function bmb_rdkey loads all wanted data in the bms structure for
70              the next crspid.
71                                  */
72
73          while ((ret = bmb_rdkey(&bms, MNEXT, wanted))!= EOFL) {
74
75          /*
76              Check if it is any error
77                                  */
78
79               if (ret == -1)
80          {
81           fprintf(stderr,"Error loading a bond structure\n");
82           exit(4);
83                  }
84
85          /*
86              Now the structure is loaded. Insert your code here. The sample program
87              will print the crspid, name, datdt, first quote date, last quote date
88              to terminal
89                                  */
90
91               printf("%15.15s %8.8s %8d %8d %8d\n", bms.bmhead.crspid,
92          bms.bmhead.name,bms.bmhead.datdt,
93          bx_cal.qdate[bms.bmhead.fstquo], bx_cal.qdate[bms.bmhead.lstquo]);
94
95            } /*end while*/
96
97
98          /*
99              Close the data files
100                                 */
101         if (bmb_close (wanted) == -1)
102              {
103               fprintf(stderr,"Error closing files\n");
104               exit (4);
105              }
106         }
```

bmb_read_rand — Read Binary Master Files Randomly

```
1
2          /*
3             Sample program to read randomly the master bond binary files. The wanted
4             crspids are read from a text file. The program should be run with two
5             parameters:
6             bndpath     - the path of the directory where the daily bonds files are
7             inpfilename - the input file name(including the path)
8                                 */
9
10         #include <stdio.h>
11         #include "bnd_struct.h"
12
13
14         extern struct BXCAL     bx_cal; /* the calendar array */
15         extern int      nbx_cal; /* The number of records in the calendar file */
16
17
18         main (argc, argv)
19
20         int argc;
21         char *argv[];
22
23         {
24         FILE               *finp; /* Pointer for the input file */
25         struct BM_STRUCT    bms;  /* The bonds master structure */
26         char                curcrspid[16]; /* The current wanted crspid */
27         int                 wanted; /*   the desired information; should be
28                                         QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM
29          or any combination */
30
31         /*
32            Check the arguments
33                                */
34
35         if (argc != 3)
36             {
37              fprintf(stderr,"Usage: %s bndpath inpfilename wanted\n", argv[0]);
38              exit (4);
39             }
40
41
42         /*
43            Open the input file
44                                */
45
46         if ((finp = fopen (argv[2], "r"))== NULL)
47             {
48              fprintf(stderr,"Cannot open %s\n", argv[2]);
49              exit(4);
50             }
51
52         /*
53             Set the wanted variable = the desired information; should be
54                         QUOTES, YIELDS, PAYMTS, DEBTS, ALLBM or any combination
55                                */
56         wanted = ALLBM;
57
58         /*
59            Open all data files and the index(addresses) file
60                                */
61
62         if (bmb_open (argv[1], wanted, "R") == -1)
63             {
64              fprintf(stderr,"Error opening files\n");
65              exit (4);
66             }
67
68         /*
```

```
69              Load the calendar into the bx_cal array and initialize nbx_cal= the number
70              of dates
71                                  */
72
73          if ((nbx_cal = bxb_cal_load(argv[1])) == -1)
74              {
75               fprintf(stderr,"Error loading the calendar\n");
76               exit (4);
77              }
78
79          /*
80              Read sequentialy the input file and call the bmb_rdkey for each read
81              crspid. The function bmb_rdkey loads all wanted data in the bms structure
82              for the desired crspid.
83                                  */
84           while (fscanf (finp, "%s", curcrspid) != EOF) {
85
86              if (bmb_rdkey(&bms, curcrspid , wanted)== -1)
87          {
88           fprintf(stderr,"Error loading the bond structure for %s\n",curcrspid);
89           exit(4);
90                 }
91
92          /*
93              Now the structure is loaded. Insert your code here. The sample program
94              will print the crspid, name, datdt, first quote date, last quote date
95              to terminal
96                                  */
97
98              printf("%15.15s %8.8s %8d %8d %8d\n", bms.bmhead.crspid,
99          bms.bmhead.name,bms.bmhead.datdt,
100         bx_cal.qdate[bms.bmhead.fstquo], bx_cal.qdate[bms.bmhead.lstquo]);
101
102           } /*end while*/
103
104         /*
105            Close the data files
106                                 */
107
108         if (bmb_close (wanted) == -1)
109             {
110              fprintf(stderr,"Error closing files\n");
111              exit (4);
112             }
113
114         /*
115            Close the input file
116                                 */
117
118         if (fclose (finp) == EOF)
119             {
120              fprintf(stderr,"Error closing the input file\n");
121              exit (4);
122             }
123         }
```

bxb_read_seq — Read Binary Cross-Sectional Files Sequentially

```
1
2           /*
3              Sample program to read sequentially the cross-sectional binary files
4              Must be run with a parameter bndpath = the path of the directory where
5              the daily bonds files are
6                                 */
7
8           #include <stdio.h>
9           #include "bnd_struct.h"
10
11          extern struct BXCAL     bx_cal; /* the calendar array */
12          extern int       nbx_cal; /*the number of records in the calendar file */
13
14
15          main (argc, argv)
16
17          int argc;
18          char *argv[];
19
20          {
21          struct BX_STRUCT bxs; /* the bonds structure */
22          char bndpath[80]; /* the path of the directory where the data files are */
23          int ret; /* return code */
24          int wanted; /*the desired information; should be
25                          QUOTES, YIELDS, ALLBX or any combination */
26
27          /*
28             Check the arguments
29                                */
30
31          if (argc != 2)
32              {
33               fprintf(stderr,"Usage: %s bndpath\n", argv[0]);
34               exit (4);
35              }
36
37          strcpy (bndpath, argv[1]);
38
39          /*
40              Set the wanted variable = the desired information; should be
41                          QUOTES, YIELDS,  ALLBX or any combination
42                                */
43          wanted = ALLBX;
44
45
46          /*
47             Open all data files and the index(addresses) file
48                                */
49
50          if (bxb_open (bndpath,wanted,"R") == -1)
51              {
52               fprintf(stderr,"Error opening files\n");
53               exit (4);
54              }
55
56          /* load the calendar */
57          if ((nbx_cal = bxb_cal_load(bndpath)) == -1)
58              {
59               fprintf(stderr,"Error loading the calendar\n");
60               exit (4);
61              }
62
63
64          /*
65             Main loop. Read sequentialy the bonds structures till the end of files.
66             The function bxb_rdkey loads all wanted data in the bxs structure for
67             the next qdate.
68                                */
```

```
 69
 70          while ((ret = bxb_rdkey(&bxs, XNEXT, wanted))!= EOFL) {
 71
 72          /*
 73             Check if it is any error
 74                                 */
 75
 76              if (ret == -1)
 77          {
 78           fprintf(stderr,"Error loading a bond structure\n");
 79           exit(4);
 80                  }
 81
 82          /*
 83             Now the structure is loaded. Insert your code here. The sample program
 84             will print the qdate, first crspid and last crspid
 85             to terminal
 86                                */
 87
 88              printf("%8d %15.15s %15.15s\n", bxs.bxhead.qdate,
 89          bxs.bxquo.crspid[0], bxs.bxquo.crspid[bxs.bxhead.numact-1]);
 90            } /*end while*/
 91
 92
 93          /*
 94             Close the data files
 95                                 */
 96          if (bxb_close (wanted) == -1)
 97              {
 98               fprintf(stderr,"Error closing files\n");
 99               exit (4);
100              }
101          }
```

bxb_read_rand — Read Binary Cross-Sectional Files Randomly

```
1
2          /*
3             Sample program to read randomly the binary  cross_sectional files.
4             The wanted qdates are read from a text file. The program should be run
5             with two parameters:
6             bndpath     - the path of the directory where the daily bonds files are
7             inpfilename - the input file name(including the path)
8
9                                 */
10
11         #include <stdio.h>
12         #include "bnd_struct.h"
13
14         extern struct BXCAL     bx_cal; /* the calendar array */
15         extern int      nbx_cal; /*the number of records in the calendar file */
16
17
18         main (argc, argv)
19
20         int argc;
21         char *argv[];
22
23         {
24         FILE               *finp;     /* Pointer for the input file */
25         struct BX_STRUCT    bxs;      /* The bonds structure */
26         char bndpath[80]; /* the path of the directory where the data files are */
27         int                 curqdate; /* The current wanted date*/
28         int                 wanted;   /*   the desired information; should be
29                                          QUOTES, YIELDS, ALLBX
30          or any combination */
31
32         /*
33            Check the arguments
34                                */
35
36         if (argc != 3)
37             {
38              fprintf(stderr,"Usage: %s bndpath inpfilename wanted\n",  argv[0]);
39              exit (4);
40             }
41
42
43         strcpy (bndpath, argv[1]);
44
45         /*
46            Open the input file
47                                */
48
49         if ((finp = fopen (argv[2], "r"))== NULL)
50             {
51              fprintf(stderr,"Cannot open %s\n", argv[2]);
52              exit(4);
53             }
54
55         /*
56             Set the wanted variable = the desired information; should be
57                           QUOTES, YIELDS, ALLBX or any combination
58                                */
59         wanted = ALLBX;
60
61         /*
62            Open all data files and the index(addresses) file
63                                */
64
65         if (bxb_open (bndpath, wanted,"R") == -1)
66             {
67              fprintf(stderr,"Error opening files\n");
68              exit (4);
```

```
69              }
70
71          /* load the calendar */
72          if ((nbx_cal = bxb_cal_load(bndpath)) == -1)
73              {
74               fprintf(stderr,"Error loading the calendar\n");
75               exit (4);
76              }
77
78
79          /*
80             Read sequentialy the input file and call the bxb_rdkey for each read
81             crspid. The function bxb_rdkey loads all wanted data in the bxs structure
82             for the desired qdate.
83                                 */
84           while (fscanf (finp, "%d", &curqdate) != EOF) {
85
86              if (bxb_rdkey(&bxs, curqdate , wanted)== -1)
87          {
88           fprintf(stderr,"Error loading the bond structure for %8d\n",curqdate);
89           exit(4);
90                  }
91
92          /*
93             Now the structure is loaded. Insert your code here. The sample program
94             will print the qdate, first crspid and last crspid
95             to terminal
96                                 */
97
98              printf("%8d %15.15s %15.15s\n", bxs.bxhead.qdate,
99          bxs.bxquo.crspid[0], bxs.bxquo.crspid[bxs.bxhead.numact-1]);
100
101           } /*end while*/
102
103
104         /*
105            Close the data files
106                                */
107
108         if (bxb_close (wanted) == -1)
109             {
110              fprintf(stderr,"Error closing files\n");
111              exit (4);
112             }
113
114         /*
115            Close the input file
116                                */
117
118         if (fclose (finp) == EOF)
119             {
120              fprintf(stderr,"Error closing the input file\n");
121              exit (4);
122             }
123         }
```

bmc_bmb_conv — Converts Master Files From Character To Binary

```
1
2          /*
3             Sample program to convert the master character bond files into binary
4                                */
5
6          #include <stdio.h>
7          #include "bnd_struct.h"
8
9
10         extern struct BXCAL     bx_cal; /* the calendar array */
11         extern int      nbx_cal; /*the number of records in the calendar file */
12
13         int bmb_quo_adr, bmb_yld_adr, bmb_debt_adr, bmb_pay_adr;
14
15
16         main (argc, argv)
17
18         int argc;
19         char *argv[];
20
21         {
22         struct BM_STRUCT bms; /* the bonds structure */
23         char bndpath[80]; /* the path of the directory where the data files are */
24         int i, j, ret;
25         int wanted;
26
27         if (argc != 2)
28             {
29              fprintf(stderr,"Usage: %s bndpath\n",  argv[0]);
30              exit (4);
31             }
32
33         strcpy (bndpath, argv[1]);
34
35         /*
36             Set the wanted variable = the desired information; should be
37                          QUOTES, YIELDS,PAYMTS, DEBTS,  ALLBM or any combination
38                                */
39         wanted = ALLBM;
40
41
42         /*
43             Initialize the addreses
44                               */
45
46         bmb_quo_adr = 0;
47         bmb_yld_adr = 0;
48         bmb_pay_adr = 0;
49         bmb_debt_adr = 0;
50
51
52         /*
53             Open the data files and the index(addresses) file - to numkeys will be
54             assigned the number of records in the address file = number of headers
55                               */
56
57         if (bmc_open (bndpath,wanted,"R") == -1)
58             {
59              fprintf(stderr,"Error opening character files\n");
60              exit (4);
61             }
62
63         if (bmb_openw (bndpath,wanted,"W") == -1)
64             {
65              fprintf(stderr,"Error opening binary files\n");
66              exit (4);
67             }
68
```

```
69          /* load the calendar */
70          if ((nbx_cal = bxc_cal_load(bndpath)) == -1)
71              {
72               fprintf(stderr,"Error loading the calendar\n");
73               exit (4);
74              }
75
76          /* write the calendar */
77          if (bxb_cal_write(bndpath) == -1)
78              {
79               fprintf(stderr,"Error writing the calendar\n");
80               exit (4);
81              }
82
83          while ((ret = bmc_rdkey(&bms, MNEXT, wanted))!= EOFL)
84            {
85          /*
86             Check if it is any error
87                                    */
88
89              if (ret == -1)
90          {
91           fprintf(stderr,"Error loading a bond structure\n");
92           exit(4);
93                  }
94
95
96          /*
97             Now the structure is loaded. Write it to the binary file
98                                     */
99              fprintf(stderr,"%s\n",bms.bmhead.crspid);
100             if (bmb_wrkey (&bms) == -1)
101             exit(1);
102           }
103
104         /* close the data files */
105         if (bmc_close (wanted) == -1)
106             {
107              fprintf(stderr,"Error character closing files\n");
108              exit (4);
109             }
110         if (bmb_closew (wanted) == -1)
111             {
112              fprintf(stderr,"Error closing binary files\n");
113              exit (4);
114             }
115         }
```

bxc_bxb_conv — Converts Cross-Sectional Files From Character To Binary

```
1
2           /*
3              Sample program to convert the cross-sectional  files from character to
4              binary.
5              Must be run with a parameter bndpath = the path of the directory where
6              the daily bonds files are
7                                       */
8
9           #include <stdio.h>
10          #include "bnd_struct.h"
11
12          extern struct BXCAL     bx_cal; /* the calendar array */
13          extern int      nbx_cal; /*the number of records in the calendar file */
14
15
16          int bxb_quo_adr, bxb_yld_adr;
17
18          main (argc, argv)
19
20          int argc;
21          char *argv[];
22
23          {
24          struct BX_STRUCT bxs; /* the bonds structure */
25          char bndpath[80]; /* the path of the directory where the data files are */
26          int ret; /* return code */
27          int wanted; /*the desired information; should be
28                           QUOTES, YIELDS, ALLBX or any combination */
29
30          /*
31             Check the arguments
32                                      */
33
34          if (argc != 2)
35              {
36               fprintf(stderr,"Usage: %s bndpath\n", argv[0]);
37               exit (4);
38              }
39
40          strcpy (bndpath, argv[1]);
41
42          /*
43              Set the wanted variable = the desired information; should be
44                           QUOTES, YIELDS,  ALLBX or any combination
45                                      */
46          wanted = ALLBX;
47
48
49          /*
50              Initialize the addreses
51                                      */
52
53          bxb_quo_adr = 0;
54          bxb_yld_adr = 0;
55
56
57          /*
58             Open all data files and the index(addresses) file
59                                      */
60
61          if (bxc_open (bndpath,wanted,"R") == -1)
62              {
63               fprintf(stderr,"Error opening files\n");
64               exit (4);
65              }
66
67          if (bxb_openw (bndpath,wanted,"W") == -1)
68              {
```

```
 69              fprintf(stderr,"Error opening binary files\n");
 70              exit (4);
 71             }
 72
 73          /* load the calendar */
 74          if ((nbx_cal = bxc_cal_load(bndpath)) == -1)
 75              {
 76               fprintf(stderr,"Error loading the calendar\n");
 77               exit (4);
 78              }
 79
 80
 81          /*
 82             Main loop. Read sequentialy the bonds structures till the end of files.
 83             The function bxc_rdkey loads all wanted data in the bxs structure for
 84             the next qdate. Function bxb_wrkey write it into the binary file.
 85                                 */
 86
 87          while ((ret = bxc_rdkey(&bxs, XNEXT, wanted))!= EOFL) {
 88
 89          /*
 90             Check if it is any error
 91                                 */
 92
 93              if (ret == -1)
 94          {
 95           fprintf(stderr,"Error loading a bond structure\n");
 96           exit(4);
 97                 }
 98
 99          /*
100             Now the structure is loaded. Write into binary files
101                                 */
102
103              printf("%8d\n", bxs.bxhead.qdate);
104              if (bxb_wrkey (&bxs) == -1)
105              exit(1);
106
107            } /*end while*/
108
109
110          /*
111             Close the data files
112                                 */
113          if (bxc_close (wanted) == -1)
114              {
115               fprintf(stderr,"Error closing files\n");
116               exit (4);
117              }
118          if (bxb_closew (wanted) == -1)
119              {
120               fprintf(stderr,"Error closing binary files\n");
121               exit (4);
122              }
123          }
```

B.4 C Include Files

bnd_struct.h — Structures Definitions

```
 1
 2          /*
 3           * bnd_struct.h
 4           *
 5           * Include file for bnd_access programs
 6           *
 7           */
 8
 9          #include "bnd_const.h"
10
11          /*
12           * The global structure used to read daily/monthly bonds master data
13           * files in indexed sequential format.
14                                    */
15
16
17          struct BM_STRUCT {
18              struct BMHEAD {
19          char    crspid[16];
20          int     type;
21          int     matdt;
22          double  couprt;
23          int     uniq;
24          int     why;
25          int     datdt;
26          int     bankdt;
27          int     fcaldt;
28          int     ymcnot;
29          int     notice;
30          int     tax;
31          int     flower;
32          int     nippy;
33          int     fcpdt;
34          int     fcpdtf;
35          double  valfc;
36          char    cusip[9];
37          char    name[9];
38          int     fstquo;
39          int     lstquo;
40          int     fstyld;
41          int     lstyld;
42          int     numpay;
43          int     numdbt;
44              }bmhead;
45
46              struct BMQUO {
47                  double  bid[MAXQUO];
48                  double  ask[MAXQUO];
49                  char    source[MAXQUO];
50              }bmquo;
51
52              struct BMYLD {
53          double  accint[MAXYLD];
54                  double  yld[MAXYLD];
55                  double  retnua[MAXYLD];
56                  double  duratn[MAXYLD];
57              }bmyld;
58
59              struct BMDEBT {
60          int     qdate[MAXDEBT];
61          int     totout[MAXDEBT];
62          int     pubout[MAXDEBT];
63              }bmdebt;
64
65              struct BMPAY {
66          int     qdate[MAXPAY];
```

```
 67          double   pdint[MAXPAY];
 68              }bmpay;
 69
 70           };
 71
 72
 73          /*
 74           * The global structure used to read the daily/monthly calendar
 75           *
 76           */
 77
 78
 79          struct BXCAL {
 80              int    qdate[MAXCAL];
 81              int    deldat[MAXCAL];
 82              float   cd1m[MAXCAL];
 83              float   cd3m[MAXCAL];
 84              float   cd6m[MAXCAL];
 85              float   cp30d[MAXCAL];
 86              float   cp60d[MAXCAL];
 87              float   cp90d[MAXCAL];
 88              float   ffefrt[MAXCAL];
 89              float   ffminr[MAXCAL];
 90              float   ffmaxr[MAXCAL];
 91              int    numact[MAXCAL];
 92           };
 93
 94
 95          /*
 96           * The global structure used to read daily/monthly bonds
 97           * cross-sectional data files in indexed sequential format.
 98           *
 99           */
100
101
102          struct BX_STRUCT {
103              struct BXHEAD {
104          int    qdate;
105          int     numact;
106              }bxhead;
107
108              struct BXQUO {
109          char    crspid[MAXHEAD][16];
110                  double  bid[MAXHEAD];
111                  double  ask[MAXHEAD];
112                  char    source[MAXHEAD];
113              }bxquo;
114
115              struct BXYLD {
116          char    crspid[MAXHEAD][16];
117          double  accint[MAXHEAD];
118                  double  yld[MAXHEAD];
119                  double  retnua[MAXHEAD];
120                  double  duratn[MAXHEAD];
121              }bxyld;
122
123           };
124
125
126
127          /*
128              The global structure for the master files index
129                                  */
130
131
132          static struct BM_ADDRS {
133          char  crspid[16];
134          int  quoloc;
135          int  quosiz;
136          int  yldloc;
137          int  yldsiz;
```

```
138          int  dbtloc;
139          int  dbtsiz;
140          int  payloc;
141          int  paysiz;
142              };
143
144          /*
145              The global structure for the cross-sectional files index
146                                   */
147
148          static struct BX_ADDRS {
149          int  qdate;
150          int  quoloc;
151          int  quosiz;
152          int  yldloc;
153          int  yldsiz;
154              };
```

bnd_const.h — Constants Definitions

```
 1
 2          /*
 3
 4           * bnd_const.h
 5           * This file contains the definitions of constants used in daily bonds
 6           * access functions
 7                                    */
 8
 9          /* Bond files types */
10
11          #define BMC      1  /* bond master character files */
12          #define BXC      2  /* bond cross-sectional character files */
13          #define BMB      3  /* bond master binary files */
14          #define BXB      4  /* bond cross-sectional files */
15
16
17          /* Maximum number of records of the character data files */
18
19          #define MAXHEAD     5500
20          #define MAXQUO       10000
21          #define MAXYLD       10000
22          #define MAXPAY       100
23          #define MAXDEBT      1000
24          #define MAXCAL      10000
25
26          /* the sizes of the records of the character files */
27
28          #define HEAD_REC       156
29          #define QUO_REC         52
30          #define YLD_REC         74
31          #define DEBT_REC        38
32          #define PAY_REC         36
33          #define BM_ADDRS_REC     96
34          #define BX_ADDRS_REC     50
35          #define CAL_REC          88
36
37
38          /* data file sequence number */
39
40          #define BMC_HEAD      0
41          #define BMC_QUO       1
42          #define BMC_YLD       2
43          #define BMC_PAY       3
44          #define BMC_DEBT      4
45          #define BMC_ADDRS     5
46          #define BXC_CAL       6
47          #define BXC_HEAD      7
48          #define BXC_QUO       8
49          #define BXC_YLD       9
50          #define BXC_ADDRS    10
51          #define BMB_HEAD     11
52          #define BMB_QUO      12
53          #define BMB_YLD      13
54          #define BMB_PAY      14
55          #define BMB_DEBT     15
56          #define BMB_ADDRS    16
57          #define BXB_CAL      17
58          #define BXB_HEAD     18
59          #define BXB_QUO      19
60          #define BXB_YLD      20
61          #define BXB_ADDRS    21
62
63          #define NUMFILES 22 /* The number of files */
64          #define MAXFILENAM       32 /* The length of the filename excluding the path */
65          #define CRSPIDLEN        15 /* The length of the crspid*/
66          #define PATHLEN          60 /* The length of the path where the files are */
67
68          /* subscript names for random accesss in master files */
```

```
 69
 70          #define MFIRST  "FIRST          "
 71          #define MPREV   "PREV           "
 72          #define MLAST   "LAST           "
 73          #define MSAME   "SAME           "
 74          #define MNEXT   "NEXT           "
 75
 76
 77          /* subscript names for random accesss in cross_sectional files */
 78
 79          #define XFIRST  -91
 80          #define XPREV   -92
 81          #define XLAST   -93
 82          #define XSAME   -94
 83          #define XNEXT   -95
 84
 85
 86          #define DQTNUM          0
 87          #define DRYNUM          1
 88          #define DPAYNUM         2
 89          #define DDEBTNUM 3
 90
 91          #define NONE    0L
 92
 93          #define QUOTES   1
 94          #define YIELDS  (1 << (DQTNUM + 1))
 95          #define PAYMTS  (1 << (DRYNUM + 1))
 96          #define DEBTS   (1 << (DPAYNUM + 1))
 97          #define ALLBM   (QUOTES | YIELDS| PAYMTS | DEBTS)
 98          #define ALLBX   (QUOTES | YIELDS)
 99
100
101          #define EOFL    -2
102          #define MAXCHAR            200
103
104
105          /* The names of the files */
106
107          #define BMC_HEAD_FILE       "bmheader.dat"
108          #define BMC_QUO_FILE        "bmquotes.dat"
109          #define BMC_YLD_FILE        "bmyield.dat"
110          #define BMC_PAY_FILE        "bmpaymts.dat"
111          #define BMC_DEBT_FILE      "bmdebt.dat"
112          #define BMC_ADDRS_FILE     "bmaddrs.dat"
113          #define BXC_CAL_FILE       "bxcalind.dat"
114          #define BXC_QUO_FILE        "bxquotes.dat"
115          #define BXC_YLD_FILE        "bxyield.dat"
116          #define BXC_ADDRS_FILE     "bxaddrs.dat"
117          #define BMB_HEAD_FILE       "bmheader.bin"
118          #define BMB_QUO_FILE        "bmquotes.bin"
119          #define BMB_YLD_FILE        "bmyield.bin"
120          #define BMB_PAY_FILE        "bmpaymts.bin"
121          #define BMB_DEBT_FILE      "bmdebt.bin"
122          #define BMB_ADDRS_FILE     "bmaddrs.bin"
123          #define BXB_CAL_FILE       "bxcalind.bin"
124          #define BXB_QUO_FILE        "bxquotes.bin"
125          #define BXB_YLD_FILE        "bxyield.bin"
126          #define BXB_ADDRS_FILE     "bxaddrs.bin"
```

C. FILE VERSION SPECIFICS

C.1 CD Label

The CRSP Daily US Government Bond Files are available on CD and have the internal volume label `BDR1_199712`. The external label has -00## appended to the internal volume label and can be ignored.

C.2 File Version Specifics

This section contains version specific information for CRSP Daily US Government Bond Files with data ending December 31, 1997. The number of issues in the master file is the total number of historical and current issues. File sizes are megabyte approximations. The binary file sizes are the sizes of files created with CRSP sample programs.

	Data Range	Trading Index Range	Total Issues	Maximum Active Issues
Daily Bond	610614-971231	1-9116	3162	254

File	# of Records	Size Character	Size Binary
BMHEADER	3,162	0.48	0.37
BMQUOTES	1,464,113	75.00	23.00
BMYIELD	1,464,113	105.00	43.00
BMDEBT	70,075	2.60	0.80
BMPAYMTS	12,794	0.45	0.22
BMADDRS	3,162	0.30	0.14
BXCALIND	9,116	0.78	0.42
BXQUOTES	1,464,113	75.00	44.00
BXYIELD	1,464,113	105.00	64.00
BXADDRS	9,116	0.45	0.17
BXDLYIND	63,812	6.40	N/A

INDEX

Q

R

S

T

U

V

W

Y