

**TPC Benchmark™ C**  
**Full Disclosure Report**  
**IBM eServer iSeries 400**  
**Model 840-2420**



**October 24, 2000**



## Special Notices

The following terms used in this publication are trademarks of **International Business Machines** Corporation in the United States and/or other countries:

- IBM ~
- iSeries 400
- Application System/400
- AS/400
- AS/400e
- DB2 for AS/400
- INT LNG ENV COBOL
- INT LNG ENV C
- IBM
- OS/400
- Structured Query Language/400 (SQL/400)

The following terms used in this publication are trademarks of other companies as follows:

**TPC Benchmark**            Trademark of the Transaction Processing Performance Council  
**TUXEDO**                    Trademark of BEA Systems, Inc.

## Revised July 12, 2000

The information contained in this document is distributed on an AS IS basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is a customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

In this document, any references made to an IBM-licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such products, programming, or services in your country. All performance data contained in this publication was obtained in a controlled environment, and therefore the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data in their specific environment.

© **International Business Machines Corporation 2000. All right reserved.**


**Note:** U.S. Government Users--Documentation related to restricted rights--Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corporation.

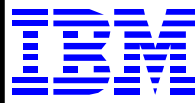
# Abstract

This report documents the full disclosure information required by the TPC Benchmark™ C Standard Specification dated October 25, 1999 for measurements on the IBM ~ iSeries 400 Model 840 with Feature Code 2420 running at 500 Mhz. The software used on the iSeries 400 Model 840-2420 systems includes OS/400 Version 4, Release 5, Modification 0, DB2 for AS/400 Version 4, Release 5, Modification 0, OS/400 Version 4, Release 5, Modification 0, INT LNG ENV COBOL OS/400 V4 R4, INT LNG ENV C OS/400 V4 R4, DB2 Query Manager and SQL Development Kit, BEA TUXEDO 6.4, and Application Development Tools.

---

**IBM®server iSeries Model 840-2420**

Company Name	System Name	Data Base Software	Operating System Software
	IBM ^ iSeries Model 840-2420	DB2 for AS/400 Version 4 Release 5	OS/400 Version 4 Release 5
Availability Date: September 15, 2000			
Total System Cost	TPC-C Throughput	Price/Performance	
- Hardware - Software -5 Years Maintenance	Sustained maximum throughput of system running TPC-C expressed in transactions per minute	Total system cost/tpmC (\$/tpmC)	
\$7,846,074	152,346.25 tpmC	\$51.50 per tpmC	



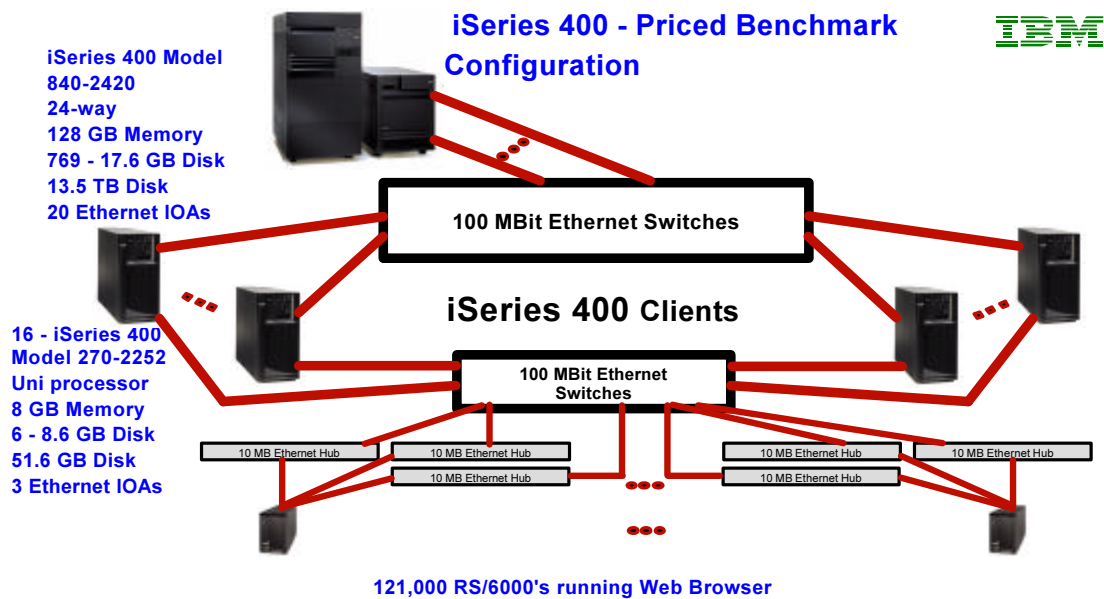
# IBM<sup>®</sup> server iSeries 400 Model 840-2420

TPC-C Rev. 3.5

Report Date:  
October 24, 2000

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date	
<b>\$7,846,074</b>	<b>152,346.25</b>	<b>\$51.50</b>	<b>September 15, 2000</b>	
Processors	Database Manager	Operating System	Other Software	Number of Users
1 iSeries 400 840-2420 (24-way) 16 iSeries 400 9406-270 2252	DB2 for AS/400 Version 4 Release 5	OS/400 Version 4 Release 5	BEA TUXEDO 6.4 INT LNG ENV COBOL OS/400 V4 R4 INT LNG ENV C OS/400 V4 R4	121,000

## iSeries 400 Model 840-2420 Priced Configuration



System components:	Qty:	Description:
Server	1	
Processors	24	iSeries 400 Model 840-2420
Memory		128 GB main memory
Disk controllers	52	RAID Disk Unit Controller (4748)
Disk drives	769	17.6 GB DASD
Total storage		Total: 13.5 TB
Clients (each):	16	
Processor	1	iSeries 400 Model 270-2252
Memory		8 GB main memory
Disk controller	2	RAID Disk Unit Controller (4748)
Disk drives	6	8.6 GB DASD
		Total: 51.6 GB



# IBM e server iSeries 400 Model 840-2420

TPC-C REV 3.5  
EXECUTIVE SUMMARY  
Report Date: October 24, 2000

## iSeries 400 Model 840-2420 Config

Item Description	Feature Number	Third Party Pricing	Unit Price	Quantity	Extended Price	Extended Maintenance	5-Year Price
<b>Server Hardware:</b>							
iSeries 400 Model 840 24-way Base System Unit	9406-840		640,000	1	640,000	84,816	724,816
Processor - ISTAR 7S 500 MHz 24-way, Cache 8x4 MB	2420		550,000	1	550,000	92,928	642,928
Interactive Card	1540		0	1	0		0
6m HSL Cable - Base I/O Tower	1461		550	20	11,000		11,000
8192 MB (RIVER - 256 MB technology)	3196		147,456	16	2,359,296		2,359,296
Programmable Regulator (req'd with 3196)	2730		750	2	1,500		1,500
PCI Ultra Magnetic Media Controller	2749		1,300	1	1,300		1,300
PCI IOP with 64 MB memory	2843		1,925	22	42,350		42,350
17.6 GB 10K RPM Disk Unit	4318		2,520	769	1,937,880		1,937,880
CD-ROM	4425		415	1	415		415
PCI Two Line WAS IOA (ECS) req'd with 5540	4745		425	1	425		425
PCI Twinaxial IOA - Req'd with 5540	4746		750	1	750		750
PCI RAID Disk Unit Controller with 26 MB write cache	4748		6,000	52	312,000		312,000
PCI 100/10 mbps Ethernet IOA	4838		900	20	18,000		18,000
PCI 100/16/4 mbps Token Ring IOA	2744		840	2	1,680		1,680
PCI Expansion Tower / Mantis	5074		17,900	17	304,300	282,336	586,636
PCI Expansion Tower	5101		9,000	18	162,000	357,696	519,696
25 GB 1/4-Inch Tape Drive	4486		6,000	1	6,000		6,000
V.24/EIA232 50ft PCI Cable - System Console	0348		125	1	125		125
<b>Server Subtotal</b>					<b>6,349,021</b>	<b>817,776</b>	<b>7,166,797</b>

<b>Client Hardware:</b>							
iSeries 400 Model 270 Base System Unit	9406-270		4,000	1	4,000	4,416	8,416
V.24/EIA 232 20ft PCI Cable for System Console	0348		125	1	125		125
6m HSL Cable - Base I/O Tower	1461		550	2	1,100		1,100
Processor Card	2252		16,000	1	16,000	7,824	23,824
Interactive Card	1516		0	1	0		0
PCI IOP with 32 MB memory	2842		1,800	3	5,400		5,400
Main Store - memory riser card - 16 slots	2884		2,200	1	2,200		2,200
512 MB DIMMS Main Storage	3025		2,048	16	32,768		32,768
8.6 GB 10K RPM Disk Unit	4317		1,400	6	8,400		8,400
CD-ROM	4525		415	1	415		415
4 GB 1/4-Inch Tape Drive	4582		1,300	1	1,300		1,300
PCI Expansion Tower	5075		6,000	1	6,000	5,856	11,856
PCI Two Line WAS IOA (ECS) req'd with 5540	4745		425	1	425		425
PCI Twinaxial IOA - Req'd with 5540	4746		750	1	750		750
PCI RAID Disk Unit Controller with 26 MB write cache	4748		6,000	2	12,000		12,000
PCI 100/10 mbps Ethernet IOA	4838		900	3	2,700		2,700
<b>Single Client Subtotal</b>					<b>93,583</b>	<b>18,096</b>	<b>111,679</b>
Number of Clients				16			
<b>Client Subtotal</b>					<b>1,497,328</b>	<b>289,538</b>	<b>1,786,864</b>

<b>Server Software:</b>							
IBM Operating System/400 V4R5M0	Bundled			1	0	41,164	41,164
DB2 Query Manager Dev. Toolkit	QU1 and ST1		28,800	1	28,800		28,800
<b>Client Software:</b>							
IBM Operating System/400 V4R5M0	Bundled		Bundled	16	0	310,263	310,263
BEA Tuxedo V6.4		1	3,000	16	48,000	38,400	86,400
ILE COBOL	5769CB1		2,400	1	2,400		2,400
Application Development Toolkit	5769PW1		3,020	1	3,020		3,020
ILE C	5769CX2		2,400	1	2,400		2,400
DB2 Query Mgr and SQL Development Kit	5769ST1		1,600	1	1,600		1,600
<b>Software Subtotal</b>					<b>86,220</b>	<b>389,827</b>	<b>476,047</b>

<b>User Connectivity:</b>							
24-port 10/100 Ethernet Switch (2 spares)		2	975	20	19,504		19,504
8-Port 10Mb Hub (10% spares)		2	29	22315	637,093		637,093
<b>User Connectivity Subtotal</b>					<b>656,598</b>	<b>0</b>	<b>656,598</b>
<b>5-year System Subtotal</b>					<b>8,589,167</b>	<b>1,497,139</b>	<b>10,086,306</b>

	% Allowance	Volume	Discount	
Revenue Allowance	22	7,884,569	(1,734,605)	
5-year Term Maintenance Contract Discount	8	1,107,312	(88,585)	
5-year Maintenance Prepayment Discount	15.05	1,018,727	(153,318)	
Software Support Discount For Secondary Systems	85	310,262	(263,723)	
		<b>Purchase</b>	<b>Maintenance</b>	<b>Total</b>
5-year System Total		6,854,562	991,512	7,846,074
tpmC				152,346.25
\$/tpmC				51.50

Note: All pricing is from IBM except items noted in Third Party Pricing column. 1. BEA Systems, Inc.; 2. Venture Computer Systems

**Notes:**  
 Server Hardware requires no charge RPK 847109  
 Revenue Allowance is applied to hardware and software for the priced configuration.  
 5-Year Term Maintenance Contract Discount is given when 5-year contract is signed.  
 5-Year Maintenance Prepay Discount is given when 5-year maintenance costs are prepaid.  
 Software Support Discount For Secondary Systems is given for maintenance costs on Multiple systems in a single I/S shop.  
 Results audited by Francois Raab of InfoSizing Inc.

**Five-Year Cost of Ownership: \$7,846,074**

**tpmC Rating: 152,346.25**

**\$/tpmC: \$51.50**

Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org.

**Numerical Quantities Summary for IBM ^**

**iSeries 400 Model 840-2420**

**MQTH**, computed Maximum Qualified Throughput 152,346.25  
 % throughput difference, reported & repeated 0.99%

<b>Response Times</b> (in seconds)	<u>90%</u>	<u>Avg.</u>	<u>Max.</u>
- New Order	0.5	0.32	9.97
- Payment	0.3	0.20	4.99
- Order-Status	0.5	0.30	2.76
- Delivery (interactive portion)	0.2	0.11	0.19
- Delivery (deferred portion)	2.1	1.70	6.18
- Stock-Level	1.8	0.82	6.00
- Menu	0.1	0.10	0.37

- Response time delayed for emulated components 0.1

**Transaction Mix** (in percent of total transactions)

- New Order	44.85%
- Payment	43.06%
- Order-Status	4.03%
- Delivery	4.03%
- Stock-Level	4.02%

**Keying/Think Times** (in seconds)

	<u>Min.</u>	<u>Avg.</u>	<u>Max.</u>
- New Order	18.00/0.00	18.00/12.01	18.23/120.23
- Payment	3.00/0.00	3.00/12.03	3.17/120.23
- Order-Status	2.00/0.00	2.00/10.01	2.10/100.20
- Delivery	2.00/0.00	2.00/5.05	2.07/50.22
- Stock-Level	2.00/0.00	2.00/5.03	2.21/50.21

**Test Duration**

- Ramp-up time	75 minutes
- Measurement interval	20 minutes
- Number of checkpoints	N/A <sup>1</sup>
- Checkpoint interval	N/A <sup>1</sup>
- Number of transactions (all types) completed in Measurement Interval	6,792,938

1. Transparent file synchronization occurred at least five times during the measurement interval.

For additional information on the IBM iSeries 400 see: <http://www.ibm.com/eserver/iseries>

## Preface

TPC Benchmark™ C Standard Specification was developed by the Transaction Processing Performance Council (TPC) and released August 13, 1992.

This is the full disclosure report for benchmark testing of the IBM ~ iSeries 400 Model 840-2420 system according to the TPC Benchmark™ C Standard Specification. Measurements were done on the IBM ~ iSeries 400 Model 840-2420 500 MHz. TPC Benchmark™ C exercises the system components necessary to perform tasks associated with that class of on-line transaction processing (OLTP) environments emphasizing a mixture of read-only and update intensive transactions. This is a complex OLTP application environment exercising a breadth of system components associated with such environments characterized by:

- The simultaneous execution of multiple transaction types that span a breadth of complexity.
- On-line and deferred transaction execution modes.
- Multiple on-line terminal sessions.
- Moderate system and application execution time.
- Significant disk input/output.
- Transaction integrity (ACID properties).
- Nonuniform distribution of data access through primary and secondary keys.
- Data bases consisting of many tables with a wide variety of sizes, attributes, and relationships.
- Contention on data access and update.

The benchmark defines four on-line transactions and one deferred transaction, intended to emulate functions that are common to many OLTP applications. However, this benchmark does not reflect the entire range of OLTP requirements. The extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other work loads or environments. Extrapolations to any other environment are not recommended.

Benchmark results are highly dependent upon work load, specific application requirements, systems design, and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-C should not be used as a substitute for a specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

The performance metric reported by TPC-C is a “business throughput” measuring the number of orders processed per minute. Multiple transactions are used to simulate the business activity of processing an order, and each transaction is subject to a response time constraint. The performance metric for this benchmark is expressed in transactions-per-minute-C (tpmC). To be compliant with the TPC-C standard, all references to tpmC results must include the tpmC rate, the associated price-per-tpmC, and the availability date of the priced configuration.

# Table of Contents

1.0 General Items .....	14
1.1 Application Code Disclosure .....	14
1.2 Benchmark Sponsor .....	14
1.3 Parameter Settings .....	14
1.4 Configuration Diagrams .....	14
1.5 IBM ~ iSeries 400 Model 840-2420 Benchmark Configuration .....	15

## **2.0 Clause 1: Logical Database Design - Related**

<b>Items</b> .....	16
2.1 Table Definitions .....	16
2.2 Database Organization .....	16
2.3 Insert and/or Delete Operations .....	16
2.4 Horizontal or Vertical Partitioning .....	17

## **3.0 Clause 2: Transaction and Terminal Profiles -**

<b>Related Items</b> .....	18
3.1 Verification for the Random Number Generator .....	18
3.2 Input/Output Screens .....	18
3.3 Terminal Features .....	19
3.4 Presentation Managers .....	19
3.5 Home and Remote Order Lines .....	19
3.6 New-Order Rollback transactions .....	19
3.7 Number of Items per Order .....	19
3.8 Home and Remote Payment Transactions .....	19
3.9 Nonprimary Key Transactions .....	20
3.10 Skipped Delivery Transactions .....	20
3.11 Mix of Transaction Types .....	20
3.12 Queuing Mechanism of Delivery .....	20
Table 1. Numerical Quantities for Transaction and Terminal Profiles.....	21

## **4.0 Clause 3: Transaction and System Properties -**

<b>Related Items</b> .....	22
4.1 Atomicity Requirements .....	22
4.1.1 Atomicity of Completed Transaction .....	22
4.1.2 Atomicity of Aborted Transactions.....	22
4.2 Consistency Requirements .....	23
4.2.1 Consistency Condition 1 .....	23
4.2.2 Consistency Condition 2 .....	23



4.2.3 Consistency Condition 3 .....	24
4.2.4 Consistency Condition 4 .....	24
4.2.5 Consistency Condition 5 .....	24
4.2.6 Consistency Condition 6 .....	25
4.2.7 Consistency Condition 7 .....	25
4.2.8 Consistency Condition 8 .....	25
4.2.9 Consistency Condition 9 .....	25
4.2.10 Consistency Condition 10 .....	25
4.2.11 Consistency Condition 11 .....	26
4.2.12 Consistency Condition 12 .....	26
4.2.13 Consistency Tests .....	26
<b>4.3 Isolation Requirements .....</b>	<b>26</b>
4.3.1 Isolation Test 1 .....	26
4.3.2 Isolation Test 2 .....	27
4.3.3 Isolation Test 3 .....	27
4.3.4 Isolation Test 4 .....	27
4.3.5 Isolation Test 5 .....	28
4.3.6 Isolation Test 6 .....	28
4.3.7 Isolation Test 7 .....	28
4.3.7.1 Isolation Test 8 .....	29
4.3.7.2 Isolation Test 9 .....	29
<b>4.4 Durability Requirements .....</b>	<b>30</b>
4.4.1 Permanent Unrecoverable Failure of any Single Durable Medium .....	30
4.4.1.1 Failure of Durable Medium of Journal Receiver and Instantaneous Interruption and Memory Failure .....	30
4.4.1.2 Failure of Durable Medium of Database .....	30
 <b>5.0 Clause 4: Scaling and Database Population -</b>	
<b>Related Items .....</b>	<b>32</b>
5.1 Cardinality of Tables .....	32
5.2 Distribution of Tables and Logs .....	32
5.3 Database Model Implemented .....	32
5.4 Partitions/Replications Mapping .....	33
 <b>6.0 Clause 5: Performance Metrics and Response</b>	
<b>Time - Related Items .....</b>	<b>34</b>
6.1 Response Times .....	34
6.2 Keying and Think Times .....	34
6.3 Response Time Frequency Distribution .....	35
6.4 Performance Curve for Response Time versus Throughput .....	38
6.5 Think Time Frequency Distribution .....	38

6.6 Throughput Versus Elapsed Time .....	39
6.7 Work Performed During Steady State .....	39
6.8 Reproducibility .....	40
6.9 Measurement Interval .....	40

## **7.0 Clause 6: SUT, Driver, and Communication**

<b>Definition-Related Items</b> .....	41
7.1 RTE Availability .....	41
7.2 Functionality and Performance of Emulated Components .....	41
7.3 Network Bandwidth .....	41
7.4 Operator Intervention .....	41

## **8.0 Clause 7: Pricing - Related Items** .....

8.1 Hardware and Programs Used .....	42
8.2 Five Year Cost of System Configuration .....	42
8.3 Statement of tpmC and Price/Performance .....	42
8.3.1 IBM ~ iSeries 400 Model 840-2420 Five-Year System Price Configuration .....	44

## **9.0 Clause 8: Audit - Related Items** .....

## **Appendix A. System Parameters and User Profile** .....

A.1 System Parameters .....	46
A.2 Transaction Subsystem Description .....	47

## **Appendix B. Database File Definitions** .....

B.1 AREFFIL: Reference File .....	48
B.2 CSTMRLFCRT: Customer Logical File .....	48
B.3 CSTMRLFNAM: Customer Names Logical File .....	48
B.4 CSTMR: Customer Logical File .....	48
B.5 CSTMRPF: Customer Physical File .....	48
B.6 DSTRCTLF: District Logical File .....	48
B.7 DSTRCT: District Physical File .....	49
B.8 HSTRY: History Logical File .....	49
B.9 HSTRYLF: History Logical File .....	49
B.10 ITEM: Item Physical File .....	49
B.11 NEWORD: New Order Logical File .....	49
B.12 NEWORDLF: New Order Logical File .....	49
B.13 NEWORDPF: New Order Physical File .....	49
B.14 ORDERS: Orders Logical File .....	49
B.15 ORDERSLF: Orders Logical File .....	49

B.16 ORDERSPF: Orders Physical File .....	49
B.17 ORDLIN: Order Lines Logical File .....	49
B.18 ORDLINLF: Order Lines Logical File .....	50
B.19 ORDLINPF: Order Lines Physical File .....	50
B.20 STOCK: Stock Logical File .....	50
B.21 STOCKPF: Stock Logical File .....	50
B.22 WRHSLF: Warehouse Logical File .....	50
B.23 WRHS: Warehouse Physical File .....	50

<b>Appendix C. Database Build Programs .....</b>	<b>51</b>
Program Flow For Build of Server, Client, and Database .....	51
C.1 CHKIXSPACE.C: .....	52
C.2 CHKWSPACE.C: .....	52
C.3 COMMON.C: .....	52
C.4 COMMON.H: .....	53
C.5 COMMONTOOL.H: .....	53
C.6 COMMONTGCC.C: .....	53
C.7 COMMONTGCC.H: .....	55
C.8 CRT1VIEW.CL: .....	56
C.9 CRT2VIEW.CL: .....	56
C.10 CRTBLDSPCE.C: .....	56
C.11 CRTDBDTA.CL: .....	57
C.12 CRTDBIX.CL: .....	58
C.13 CRTDIST.CL: .....	58
C.14 CRTHASH.C: .....	59
C.15 CRTHISFILE.CL: .....	59
C.16 CRTHSTRYLF.CL: .....	59
C.17 CRTINDEX.CL: .....	59
C.18 CRTITEM.CL: .....	60
C.19 CRTORDFILE.CL: .....	60
C.20 CRTTPCCDB.CL: .....	61
C.21 CRTTPCCJRN.CL: .....	62
C.22 CRTWHFILE.CL: .....	63
C.23 CSTMRVIEW.CL: .....	63
C.24 FILCUSFILE.CL: .....	63
C.25 FILLCUS.C: .....	64
C.26 FILLDATA.H: .....	65
C.27 FILLDIST.C: .....	66
C.28 FILLHIST.C: .....	66
C.29 FILLITEM.C: .....	68
C.30 FILLORD.C: .....	69
C.31 FILLPARTS.CL: .....	70

C.32 FILLSTOCK.C:	72
C.33 FILLWH.C:	73
C.34 FILSTKFILE.CL:	74
C.35 MASTCUSSTK.C:	74
C.36 MASTORD.C:	78
C.37 CRTSTOCKLF.CL:	79
C.38 STRTPCCJRN.CL:	79
C.39 TPCCSPACE.C:	80
C.40 TPCCSPACE.H:	81
C.41 TPCCSPACEI.H:	82

<b>Appendix D. Application Source Code</b>	83
Program Flow	83
CLIENT CODE:	84
D.1 COMMON.C:	84
D.2 COMMON.H:	84
D.3 DELVRYSRV.C:	85
D.4 FORMCHILDS.C:	86
D.5 FORMPARENT.C:	86
D.6 FORMSERVER.C:	87
D.7 GETDTTMSTR.C:	91
D.8 IMPORT.H:	92
D.9 NEWORDSRV.C:	93
D.10 ORDSTSSRV.C:	93
D.11 PAYMNTSRV.C:	93
D.12 POSTRESULT.C:	94
D.13 SCANINPUT.C:	96
D.14 STKLVLRSRV.C:	99
D.15 TUXCONFIG File:	99
SERVER CODE:	99
D.16 COMMON.H	99
D.17 DELIVERY.CBL:	100
D.18 DLVRSRVR.C:	101
D.19 GETTIME.C:	102
D.20 NEWORDER.CBL:	102
D.21 NEWSRVR.C:	105
D.22 ORDSRVR.C:	105
D.23 ORDSTS.CBL:	106
D.24 PAYMENT.CBL:	107
D.25 PAYSRVR.C:	110
D.26 STKLVL.CBL:	110
D.27 STKSRVR.C:	111

D.28 TOOMANY.CBL: .....	112
D.29 TPCCUSER.H: .....	112
<b>Appendix E. RTE Scripts</b> .....	113
E.1 RTE Parameters .....	113
E.2 TRANSACTIONS.H .....	115
E.3 TRANSACTIONS.C .....	115
<b>Appendix F. 180-Day DASD Requirements</b> .....	121
F.1 IBM ~ iSeries 400 Model 840-2420 -- 180-Day DASD Requirements .....	121
F.2 IBM ~ iSeries 400 Model 840-2420 -- Journal DASD Requirements .....	122
<b>Appendix G. Third Party Quotes</b> .....	123
BEA Tuxedo Quote .....	123
Venture Computer Systems - Ethernet Switches/Hubs Quote .....	125
<b>Appendix H. Auditor Letter</b> .....	126

# 1.0 General Items

## 1.1 Application Code Disclosure

*The application program (as defined in Clause 2.1.7) must be disclosed. This includes, but is not limited to, the code implementing the five transactions and the terminal input and output functions.*

Appendix D contains the iSeries 400 application code for the five TPC Benchmark™ C transactions and the terminal functions.

## 1.2 Benchmark Sponsor

*A statement identifying the benchmark sponsor(s) and other participating companies must be provided.*

This benchmark was sponsored by **International Business Machines** Corporation.

## 1.3 Parameter Settings

*Settings must be provided for all customer-tunable parameters and options which have been changed from the defaults found in actual products including, but not limited to:*

- *Database tuning options.*
- *Recovery/commit options.*
- *Consistency/locking options.*
- *Operating system and application configuration parameters.*

Appendix A contains the system, database, and application parameters changed from their default values used in these TPC Benchmark™ C tests.

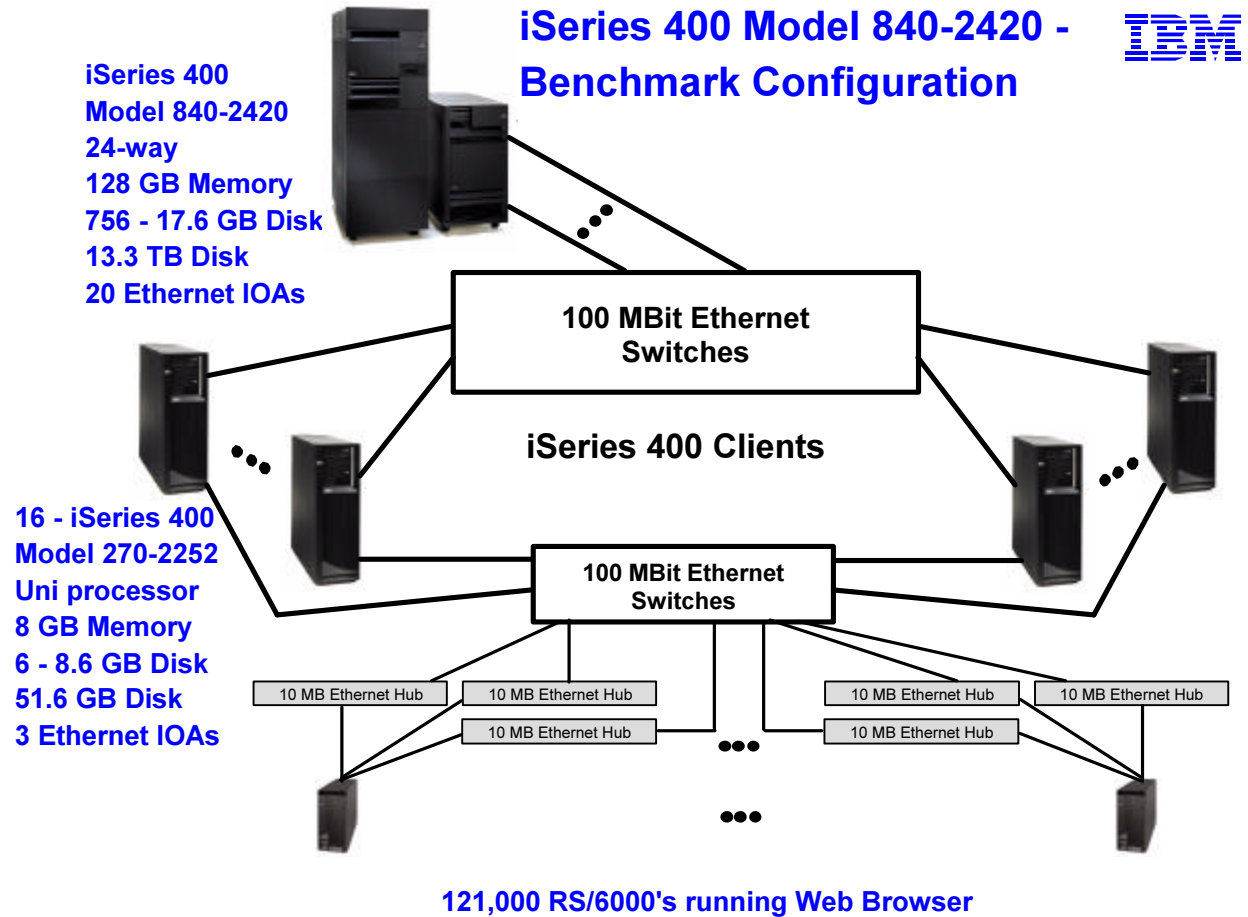
## 1.4 Configuration Diagrams

*Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:*

- *Number and type of processors.*
- *Size of allocated memory, and any specific mapping/partitioning of memory unique to the test.*
- *Number and type of disk units (and controllers if applicable).*
- *Number of channels or bus connections to disk units, including the protocol type.*
- *Number of LAN (e.g., Ethernet) connections, including routers, workstations, terminals, etc., that were physically used in the test or are incorporated into the pricing structure (see Clause 8.1.8).*
- *Type and run-time execution location of software components (e.g., DBMS, client processes, transaction monitors, software drivers, etc.).*

# 1.5 IBM ^ Configuration

## iSeries 400 Model 840-2420 Benchmark



# 2.0 Clause 1: Logical Database Design - Related Items

## 2.1 Table Definitions

*Listing must be provided for all table definition statements and all other statements used to set up the database.*

The listings for all file definitions to create the database files are available in Appendix B and the programs used for loading the database (minimal population) are provided in Appendix C.

## 2.2 Database Organization

*The physical organization of table and indices, within the database, must be disclosed.*

Physical space for each file (table) is allocated by OS/400 as the file is filled. Although the initial build and the application transactions add records (rows) to several files in the same transaction, each file's extent will reside in separate areas on physical disk. OS/400 will spread the extents for each file across all available disk units to ensure that multiple access requests to the same file may be handled simultaneously. Records are added contiguously within extents, crossing page boundaries where necessary.

Files are created in sequential order according to their primary key.

Indices are generated concurrently with data for Warehouse, District, and Item tables. All other indices are generated after the database is populated. The available space within the index is included in the space reported in this disclosure.

## 2.3 Insert and/or Delete Operations

*It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C transaction mix. Furthermore, any restriction in the SUT database implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the maximum key value for these new rows.*

During the course of the testing, records were inserted into the ITEM file while users were executing the defined TPC-C transactions.

All of the files used by TPC-C transactions were created with the following attributes:

Authority	*PUBLIC	(Any user can view and modify the files).
ALWUPD	*YES	(Allow update and insert of records).
ALWDLT	*YES	(Allow delete of records).
ALWWRT	*YES	(Allow write of records).



Static files were created with \*NOMAX specified on the number of records allowed. This limit is therefore set by the operating system at 2,147,483,648 records. Dynamic files were created with an initial size that is slightly larger than initial database requirement, and extent definitions that would allow expansion, as needed.

## 2.4 Horizontal or Vertical Partitioning

*While there are few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark, any such partitioning must be disclosed.*

Horizontal partitioning was implemented on all the files except ITEM. The partitioning was done based on the Warehouse ID key field. The following eight tables: Warehouse, District, Customer, History, Neworder, Orders, Orderline, and Stock were split such that records for Warehouse ID's 1 through 6,100 were in the first partition and 6,101 through 12,200 were in the second partition.

## 3.0 Clause 2: Transaction and Terminal Profiles - Related Items

### 3.1 Verification for the Random Number Generator

*The method of verification for the random number generation must be disclosed.*

The `srandom()`, `getpid()`, and `gettimeofday()` functions are used to produce unique random seeds for each driver. The drivers use these seeds to seed the `srand()`, `srandom()`, and `srand48()` functions. Random numbers are produced using wrappers around the standard system random number generators.

The negative exponential distribution uses the following function to generate the distribution. This function has the property of producing a negative exponential curve with a specified average and a maximum value 4 times the average.

```
const double RANDOM_4_Z=0.89837799236185
const double RANDOM_4_K=0.97249842407114

double neg_exp_4(double average {
    return - average * (1/RANDOM_4_Z * log (1 - RANDOM_4_K * drand48()));
})
```

The random functions used by the driver system and the database generation program were verified. The `C_LAST` column was queried to verify the random values produced by the database generation program. After a measurement, the `HISTORY`, `ORDER`, and `ORDER_LINE` tables were queried to verify the randomness of values generated by the driver. The rows were counted and grouped by customer and item numbers.

Here is an example of one SQL query used to verify the random number generation functions:

- `create table TEMP (W_ID int, D_ID, C_LAST char(16), CNTR int);`
- `insert into TEMP select C_W_ID, C_D_ID, C_LAST, COUNT(*) from CUSTOMER group by C_W_ID, C_D_ID, C_LAST;`
- `select CNTR, COUNT(*) from TEMP group by CNTR order by 1;`

### 3.2 Input/Output Screens

*The actual layouts of the terminal input/out screens must be disclosed. (8.1.3.2)*

The screen layouts are based on those in Clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3 and 2.8.3 of the TPC Benchmark C Standard Specification. .

### **3.3 Terminal Features**

*The method used to verify that the emulated terminals provide all the features described in Clause 2.2.2.4 must be explained. Although not specifically priced, the type and model of the terminals used must for the demonstration in 8.1.3.3 must be disclosed and commercially available (including supporting software and maintenance). (8.1.3.3)*

The auditor verified terminal features by direct experimentation. The benchmarked configuration uses a browser and HTML scripts as the terminal interface

The following numbered items correspond directly to the seven items listed under Clause 2.2.2.4 with a description of how the requirement was met.

### **3.4 Presentation Managers**

*Any usage of presentation managers or intelligent terminals must be explained. (8.1.3.4)*

The terminals emulated in the priced configuration are IBM RS/6000 desktop computer systems. All processing of the input/output screens was handled by the IBM iSeries 400 clients. The screen input/output was managed via HTML strings that comply with the HTML Version 2.0 specification. A listing of the code used to implement the intelligent terminals is provided in Appendix A. All data manipulation was handled by the IBM iSeries 400 clients.

### **3.5 Home and Remote Order Lines**

*The percentage of home and remote order lines in the New-Order transactions must be disclosed.*

Table 1 on page 21 shows the percentage of home and remote transactions that occurred during the measurement period for the New-Order transactions.

### **3.6 New-Order Rollback transactions**

*The percentage of New-Order transactions that were rolled back as a result of an illegal item number must be disclosed.*

Table 1 on page 21 shows the percentage of New-Order transactions that were rolled back due to an illegal item being entered.

### **3.7 Number of Items per Order**

*The number of items per orders entered by New-Order transactions must be disclosed.*

Table 1 on page 21 shows the average number of items ordered per New-Order transaction.

### **3.8 Home and Remote Payment Transactions**

*The percentage of Home and Remote Payment transactions must be disclosed.*

Table 1 on page 21 shows the percentage of Home and Remote transactions that occurred during the measurement period for the Payment transactions.

### **3.9 Nonprimary Key Transactions**

*The percentage of Payment and Order-Status transactions that used nonprimary key (C\_LAST) access to the database must be disclosed.*

Table 1 on page 21 shows the percentage of nonprimary key access to the database by the Payment and Order-Status transactions.

### **3.10 Skipped Delivery Transactions**

*The percentage of Delivery transactions that were skipped as a result of an insufficient number of rows in the NEW-ORDER table must be disclosed.*

Table 1 on page 21 shows the percentage of Delivery transactions missed due to a shortage of supply in the NEW-ORDER table.

### **3.11 Mix of Transaction Types**

*The mix (ie, percentages) of transaction types seen by the SUT must be disclosed.*

Table 1 on page 21 shows the mix percentage for each of the transaction types executed by the SUT.

### **3.12 Queuing Mechanism of Delivery**

*The queuing mechanism used to defer execution of the Delivery transaction must be disclosed.*

Deferred queuing of the Delivery transaction is handled within standard support from the Tuxedo transaction monitor.

**Table 1. Numerical Quantities for Transaction and Terminal Profiles**

<b>New Order</b>	<b>IBM ^</b>	<b>iSeries 400 Model 840-2420</b>
Percentage of Home order lines		99.00%
Percentage of Remote order lines		1.00%
Rolled Back Transactions		0.99%
Number of Items per order		10
<b>Payment</b>		
Percentage of Home transactions		85.02%
Percentage of Remote transactions		14.98%
<b>Nonprimary Key Access</b>		
Percentage of Payment using C_LAST		59.95%
Percentage of Order-Status using C_LAST		59.92%
<b>Delivery</b>		
Delivery transactions skipped		0
<b>Transaction Mix</b>		
New-Order		44.85%
Payment		43.06%
Order-Status		4.03%
Stock-Level		4.02%
Delivery		4.03%

## 4.0 Clause 3: Transaction and System Properties - Related Items

*The results of the ACID test must be disclosed along with a description of how the ACID requirements were met.*

### 4.1 Atomicity Requirements

*The system under test must guarantee that database transactions are atomic; the system will either perform all individual operations on the data, or will assure that no partially-completed operations leave any effects on the data.*

#### 4.1.1 Atomicity of Completed Transaction

*Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number) and verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have been changed appropriately.*

The following steps were performed to verify the atomicity of completed transactions:

1. Randomly select a Customer, District, and Warehouse, and query the Customer, District, and Warehouse tables.
2. Execute a Payment transaction for the Customer, District, and Warehouse used in Step1 above. Commit the transaction.
3. Repeat the query performed in Step1 to demonstrate that the appropriate changes have been made.

#### 4.1.2 Atomicity of Aborted Transactions

*Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number) and substitute a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have NOT been changed.*

The following steps were performed to verify the atomicity of the aborted Payment transaction:

1. Randomly select a Customer, District, and Warehouse, and query the Customer, District, and Warehouse tables.
2. Execute a Payment transaction for the Customer, District, and Warehouse used in Step1 above. Roll-back the transaction.
3. Repeat the query performed in Step1 to verify that no changes have been made to the database.

## 4.2 Consistency Requirements

*Consistency is the property of the application that requires an execution of a database transaction to take the database from one consistent state to another, assuming that the database is initially in a consistent state.*

### 4.2.1 Consistency Condition 1

*Entries in the WAREHOUSE and DISTRICT tables must satisfy the relationship:*

$$W\_YTD = \text{sum}(D\_YTD)$$

*for each warehouse defined by ( $W\_ID = D\_W\_ID$ )*

The following SQL queries were executed before and after transactions were run to show that the database was always in a consistent state.

```
Select WID, WYTD from WRHS
Select DWID, sum(DYTD) from DSTRCT group by DWID
```

The results of these two queries were then compared to verify consistency.

### 4.2.2 Consistency Condition 2

*Entries in the DISTRICT, ORDER, and NEW-ORDER tables must satisfy the relationship:*

$$D\_NEXT\_O\_ID - 1 = \text{max}(O\_ID) = \text{max}(NO\_O\_ID)$$

*for each district defined by ( $D\_W\_ID = O\_W\_ID = NO\_W\_ID$ ) and ( $D\_ID = O\_D\_ID = NO\_D\_ID$ ). This condition does not apply to the NEW-ORDER table for any districts which have no outstanding new orders.*

The following SQL queries were executed before and after transactions were run to show that the database was always in a consistent state.

```
Create View QRYTEMP1(OWID, ODID, MAXOID)
  As Select OWID, ODID, MAX(OID) from ORDERS group by OWID, ODID
Create View QRYTEMP2(NOWID, NODID, MAXNOOID)
  As Select NOWID, NODID, MAX(NOOID) from NEWORDS group by NOWID, NODID
Select DWID, DID, (DNXTOR-1), MAXOID, MAXNOOID
  from DSTRCT, QRYTEMP1, QRYTEMP2 where DWID = OWID and DWID=NOWID
  and DID =ODID and DID = NODID and (((DNXTOR-1) <> MAXOID)
  or ((DNXTOR-1) <> MAXNOOID) or (MAXOID <> MAXNOOID))
```

If any records are produced by these queries, the database would be inconsistent. No records were produced by these queries.

### 4.2.3 Consistency Condition 3

Entries in NEW-ORDER table must satisfy the relationship:

$$\max(NO\_O\_ID) - \min(NO\_O\_ID) + 1 = \{\text{number of rows in the NEW-ORDER table for this district}\}$$

for each district defined by NO\_W\_ID and NO\_D\_ID. This condition does not apply to any districts which have no outstanding new orders.

The following SQL queries were executed before and after transactions were run to show that the database was always in a consistent state.

```
Create View QRYTEMP3(NOWID, NODID, MAXNOOID, MINNOOID, MAXMIN, COUNTO ID)
As Select NOWID, NODID, MAX(NOOID), MIN(NOOID), (MAX(NOOID) -
MIN(NOOID) +1), COUNT(*) from NEWORD group by NOWID, NODID
Select * from QRYTEMP3 where MAXMIN <> COUNTOID
```

If any records are produced by these queries, the database would be inconsistent. No records were produced by these queries.

### 4.2.4 Consistency Condition 4

Entries in the ORDER and ORDER-LINE tables must satisfy the relationship:

$$\text{sum}(O\_OL\_CNT) = \{\text{number of rows in the ORDER-LINE table for this district}\}$$

for each district defined by (O\_W\_ID = OL\_W\_ID) and (O\_D\_ID = OL\_D\_ID).

The following SQL queries were executed before and after transactions were run to show that the database was always in a consistent state.

```
Create View QRYTEMP4(OLWID, OLDID, COUNTORD)
As Select OLWID, OLDID, COUNT(*)
From ORDERLINE Group by OLWID, OLDID
Create View QRYTEMP5(OWID, ODID, SUMOLINES)
As Select OWID, ODID, SUM(OLINES)
From ORDERS Group by OWID, ODID
Select OWID, ODID, SUMOLINES, COUNTORD
From QRYTEMP4, QRYTEMP5 Where OWID = OLWID and ODID = OLDID
and SUMOLINES <> COUNTORD Order by OWID, ODID
```

If any records are produced by these queries, the database would be inconsistent. No records were produced by these queries.

### 4.2.5 Consistency Condition 5

For any row in the ORDER table, O\_CARRIER\_ID is set to a null value if and only if there is a corresponding row in the NEW-ORDER table defined by (O\_W\_ID, O\_D\_ID, O\_ID) = (NO\_W\_ID, NO\_D\_ID, NO\_O\_ID).

This consistency test completed successfully.



## 4.2.6 Consistency Condition 6

For any row in the ORDER table,  $O\_OL\_CNT$  must equal the number of rows in the ORDER-LINE table for the corresponding order defined by  $(O\_W\_ID, O\_D\_ID, O\_ID) = (OL\_W\_ID, OL\_D\_ID, OL\_O\_ID)$ .

This consistency test completed successfully.

## 4.2.7 Consistency Condition 7

For any row in the ORDER-LINE table,  $OL\_DELIVERY\_D$  is set to a null date/time if and only if the corresponding row in the ORDER table defined by  $(O\_W\_ID, O\_D\_ID, O\_ID) = (OL\_W\_ID, OL\_D\_ID, OL\_O\_ID)$  has  $(O\_CARRIER\_ID)$  set to a null value.

This consistency test completed successfully.

## 4.2.8 Consistency Condition 8

Entries in the WAREHOUSE and HISTORY tables must satisfy the relationship:

$$W\_YTD = \text{sum}(H\_AMOUNT)$$

for each warehouse defined by  $(W\_ID = H\_W\_ID)$ .

This consistency test completed successfully.

## 4.2.9 Consistency Condition 9

Entries in the DISTRICT and HISTORY tables must satisfy the relationship:

$$D\_YTD = \text{sum}(H\_AMOUNT)$$

for each district defined by  $(D\_W\_ID, D\_ID = H\_W\_ID, H\_D\_ID)$ .

This consistency test completed successfully.

## 4.2.10 Consistency Condition 10

Entries in the CUSTOMER, HISTORY, ORDER, and ORDER-LINE tables must satisfy the relationship:

$$C\_BALANCE = \text{sum}(OL\_AMOUNT) - \text{sum}(H\_AMOUNT)$$

where:

$H\_AMOUNT$  is selected by  $(C\_W\_ID, C\_D\_ID, C\_ID) = (H\_C\_W\_ID, H\_C\_D\_ID, H\_C\_ID)$

and:

$OL\_AMOUNT$  is selected by:

$(OL\_W\_ID, OL\_D\_ID, OL\_O\_ID) = (O\_W\_ID, O\_D\_ID, O\_ID)$  and

$(O\_W\_ID, O\_D\_ID, O\_C\_ID) = (C\_W\_ID, C\_D\_ID, C\_ID)$  and

$(OL\_DELIVERY\_D)$  is not a null value

This consistency test completed successfully.

#### **4.2.11 Consistency Condition 11**

*Entries in the CUSTOMER, ORDER, and NEW-ORDER tables must satisfy the relationship:*

$$(count(*) \text{ from } ORDER) - (count(*) \text{ from } NEW-ORDER) = sum(C\_DELIVERY\_CNT)$$

*for each district defined by (O\_W\_ID, O\_D\_ID) = (NO\_W\_ID, NO\_D\_ID) = (C\_W\_ID, C\_D\_ID).*

This consistency test completed successfully.

#### **4.2.12 Consistency Condition 12**

*Entries in the CUSTOMER, and ORDER-LINE table must satisfy the relationship:*

$$C\_BALANCE + C\_YTD\_PAYMENT = sum(OL\_AMOUNT)$$

*for any randomly selected customers and where OL\_DELIVERY\_ID is not set to a null date/time.*

This consistency test completed successfully.

All 12 consistency tests were completed successfully.

#### **4.2.13 Consistency Tests**

*Verify that the database is initially consistent by verifying that it meets the consistency conditions defined in Clauses 3.3.2.1 to 3.3.2.4. Describe the steps used to do this in sufficient detail so that the steps are independently repeatable.*

The queries defined in 4.2.1 through 4.2.4 were run after initial database build and prior to executing any transactions. All queries showed that the database was in a consistent state.

After executing transactions at full load for approximately 10 minutes, the queries defined in 4.2.1 through 4.2.4 were run again. All queries showed that the database was still in a consistent state.

### **4.3 Isolation Requirements**

Operations of concurrent database transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed to completion in some order.

#### **4.3.1 Isolation Test 1**

*This test demonstrates isolation for read-write conflicts of Order-Status and New-Order transactions.*

The following steps were performed to satisfy the test of isolation for Order-Status and New-Order transactions:

1. First terminal: Start a New-Order transaction with the necessary inputs. The transaction is delayed to pause the program execution.
2. Second terminal: Start an Order-Status transaction for the same customer as used in the New-Order transaction.
3. Second terminal: The Order-Status transaction attempts to read the CUSTOMER file but is locked out by the New-Order transaction waiting to complete.
4. First terminal: The New-Order transaction is released and the Commit is executed releasing the record. With the CUSTOMER record now released, the Order-Status transaction can now complete.
5. Second terminal: Verify that the Order-Status transaction completes after the New-Order transaction. and that the results displayed for the Order-Status transaction match the input for the New-Order transaction.

### **4.3.2 Isolation Test 2**

*This test demonstrates isolation for read-write conflicts of Order-Status and New-Order transactions when the New-Order transaction is rolled back.*

The following steps were performed to satisfy the test of isolation for Order-Status and a rolled back New-Order transaction.

1. First terminal: Perform a New-Order transaction for the same customer in the Order-Status transaction in Isolation Test 1; include an invalid item number in the order. The transaction is delayed just prior to the rollback.
2. Second terminal: Start an Order-Status transaction for the same customer used in the New-Order transaction. The Order-Status transaction attempts to read the CUSTOMER file but is locked by the New-Order transaction.
3. First terminal: Roll back the New-Order transaction. With the CUSTOMER record now released, the Order-Status transaction completes.
4. Verify the results from the Order-Status transaction matches those in Isolation Test 1.

### **4.3.3 Isolation Test 3**

*This test demonstrates isolation for write-write conflicts of two New-Order transactions.*

1. The following steps were performed to verify isolation of two New-Order transactions:
2. First terminal: Start a New-Order transaction using the necessary inputs. The transaction is delayed just prior to the Commit.
3. Second terminal: Start a second New-Order transaction for the same customer used by the first terminal. This transaction is forced to wait while the first terminal holds a lock on the DISTRICT record requested by the second terminal.
4. First terminal: The New-Order transaction is allowed to complete and Commit the transaction. With the DISTRICT record released, the second terminal New-Order transaction will complete.
5. Verify the order number from the second terminal New-Order transaction is one greater than the order number from the first terminal.

### **4.3.4 Isolation Test 4**

*This test demonstrates isolation for write-write conflicts of two New-Order transactions when one transaction is rolled back.*

The following steps were performed to verify isolation of two New-Order transactions after one is rolled back:

1. First terminal: Start a New-Order transaction using the necessary inputs to cause a roll back (invalid item number). The transaction is delayed just prior to the rollback.
2. Second terminal: Start a second New-Order transaction for the same customer used by the first terminal. This transaction is forced to wait while the first terminal holds a lock on the DISTRICT record requested by the second terminal.
3. First terminal: The New-Order transaction is allowed to complete, and the transaction is rolled back due to the invalid item number.
4. Second terminal: With the DISTRICT record released, the second terminal New-Order transaction will complete normally.
5. Verify the order number from the second terminal New-Order transaction is equal to the next order number before either New-Order transaction was started.

### **4.3.5 Isolation Test 5**

*This test demonstrates isolation for write-write conflicts of Payment and Delivery transactions.*

The following steps were performed to successfully conduct this test:

1. First terminal: A Delivery transaction is started. The transaction is delayed just prior to the Commit.
2. Second terminal: Start a Payment transaction for a customer that will have an order delivered in this transaction started in Step 1. The Payment transaction is forced to wait while the Delivery transaction holds a lock on the CUSTOMER record.
3. The Delivery transaction completes.
4. Second terminal: With the CUSTOMER record released, the Payment transaction is now able to complete.

### **4.3.6 Isolation Test 6**

*This test demonstrates isolation for write-write conflicts of Payment and Delivery transactions when the Delivery transaction is rolled back.*

The following steps were performed to successfully conduct this test:

1. First terminal: Start a Delivery transaction. The transaction is delayed just prior to the rollback.
2. Second terminal: Start a Payment transaction for a customer that will have an order delivered in this transaction started in Step 1. The Payment transaction is forced to wait while the Delivery transaction holds a lock on the CUSTOMER record.
3. The Delivery transaction rolls back.
4. Second terminal: With the CUSTOMER record released, the Payment transaction is now able to complete.

### **4.3.7 Isolation Test 7**

*This test demonstrates repeatable reads for the New-Order transaction while an interactive transaction updates the price of an item.*

The following steps were performed to successfully conduct this test:

1. First terminal: Execute a New Order transaction including items x and y.

2. First terminal: A New-Order transaction is started that contains item x twice and item y once. This transaction is stopped after reading the price of item x from the item file the first time.
3. Second terminal: Using interactive SQL, an update transaction is started for items x and y, increasing their price by 10%. Case A, transaction 3 stalls, occurs.
4. First terminal: The New-Order transaction is allowed to complete. It is verified that the prices for items x and y are the same throughout the entire transaction and that they match the results of Step 1.
5. Second terminal: After the New-Order transaction completes, the update transaction completes and is committed.
6. First terminal: Step1 is repeated, noting that the prices of items x and y now match those set in Step 3.

#### **4.3.7.1 Isolation Test 8**

*This test demonstrates isolation for phantom protection between a Delivery and a New-Order transaction.*

The following steps were performed to successfully conduct this test:

1. First terminal: All rows for a randomly selected district and warehouse were removed from the NEW-ORDER table.
2. First terminal: A Delivery transaction for the selected warehouse was started.
3. First terminal: The Delivery transaction was stopped immediately after reading the NEW-ORDER table for the selected district. No qualifying row was found.
4. Second terminal: A New-Order transaction was started for the same warehouse and district. Case A, Transaction 2 stalled.
5. First terminal: Repeated read of the NEW-ORDER table for the selected district.
6. Again no qualifying row was found.
7. First terminal: The Delivery transaction was allowed to complete and was COMMITTED.
8. Second terminal: The NEW-ORDER transaction completed successfully.

#### **4.3.7.2 Isolation Test 9**

*This test demonstrates isolation for phantom protection between an Order-Status and a New-Order transaction.*

The following steps were performed to successfully conduct this test:

1. First terminal: An Order-Status transaction for a selected customer was started.
2. First terminal: The Order-Status transaction was stopped immediately after reading the ORDER table for the selected customer. The most recent order for that customer was found.
3. Second terminal: A NEW-ORDER transaction was started for the same customer. Case A, Transaction 2 stalled.
4. First terminal: Repeated read of the ORDER table for the selected customer.
5. Verified the order found was the same as in step 3.
6. First terminal: The Order-Status transaction was allowed to complete and was COMMITTED.
7. Second terminal: The NEW-ORDER transaction completed successfully.

## 4.4 Durability Requirements

*The tested system must guarantee durability: the ability to preserve the effects of committed transactions and ensure database consistency after recovery from any one of the failures listed in Clause 3.5.3.*

### 4.4.1 Permanent Unrecoverable Failure of any Single Durable Medium

*Permanent unrecoverable failure of any single durable medium containing TPC-C database tables or recovery log data.*

The iSeries 400 Model 840-2420 implementation of the TPC Benchmark™ C divides the configured disk space into two available auxiliary storage pools (ASP): System ASP and User ASP. The system ASP contains the operating system, integrated relational database, the application libraries, and the TPC-C tables. The System ASP is protected by device parity protection (RAID-5) and the User ASP, which contains the journal receiver (recovery log), is mirrored.

#### 4.4.1.1 Failure of Durable Medium of Journal Receiver and Instantaneous Interruption and Memory Failure

The following steps were performed to successfully complete the test of the Durability of the journal receiver:

1. The current count of the total number of orders was determined by the sum of D\_NEXT\_O\_ID of all rows in the DISTRICT table giving SUM\_1.
2. A full scale test was started on the SUT. The test was allowed to run for 10 minutes before creating the failure.
3. The signal cable from a single disk unit in the User ASP was disconnected. Since the User ASP is protected by mirroring, the system continued to process transactions. Detection of the device failure caused a diagnostic message to be issued. Processing of transactions continued without performance degradation.
4. Processing was allowed to continue for an additional 10 minutes.
5. An instantaneous power failure was simulated by issuing an immediate power-off at the service panel.
6. The system was then powered on and IPLed.
7. The number of New-Order transactions executed by the SUT is verified against the number of successful transactions logged by the RTE.
8. Step1 above was performed again retrieving the new total of orders processed, SUM\_2. The difference between SUM\_2 and SUM\_1 was compared to the number of transactions reported by RTE.

#### 4.4.1.2 Failure of Durable Medium of Database

The following steps were performed to successfully perform the Durability test of failure of a disk unit with database tables:

Note: This test was combined with the tests in 4.4.1.1, because the ASP with the disk units containing the database tables is protected by device parity protection (RAID5).

1. The database tables reside on the system ASP.
2. After a disk unit in user ASP was disconnected as mentioned in step 3 in 4.4.1.1, and we let the processing continue for about 10 minutes as mentioned in step 4 in 4.4.1.1, a disk unit in the system ASP was disconnected.

Since the system ASP is protected by device parity protection, ( RAID 5) the system continued to process transactions. Detection of the device failure caused a diagnostic message to be issued. This also resulted in parity protection being suspended for the parity set that that disk unit was in. Processing of transactions continued without performance degradation.

3. After the system was powered off, the disk unit was plugged in.
4. After the system was powered back on, the disk was reinstalled and the parity protection was resumed again for that parity set.

# 5.0 Clause 4: Scaling and Database Population - Related Items

## 5.1 Cardinality of Tables

The cardinality (ie, the number of rows) of each table, as it existed at the start of the benchmark run, must be disclosed.

Table 2 portrays the TPC Benchmark™ C defined tables and the number of rows for each table as they were built initially.

**Table 2. Initial Database Build (# of rows per table)**

TPC Benchmark C Tables	IBM ~ iSeries 400 Model 840-2420
WAREHOUSE	12,200
CUSTOMER	366,000,000
NEW-ORDER	109,800,000
DISTRICT	122,000
STOCK	1,220,000,000
ORDERS	366,000,000
ORDER-LINE	3,660,140,702
HISTORY	366,000,000
ITEM	100,000

## 5.2 Distribution of Tables and Logs

The distribution of tables and logs across all media must be explicitly depicted for the tested and priced systems.

The iSeries 400 system utilizes a Single-Level Storage concept where OS/400 views all drives in an Auxiliary Storage Pool (ASP) as a single virtual drive. This technique spreads information across all available drives in an ASP, attempting to maintain equivalent percentages of free storage. For this Benchmark, a 630-Disk RAID-5 ASP was used for system code, application code, and the database. A separate, fully mirrored, 126-Disk ASP was used for log data.

## 5.3 Database Model Implemented

A statement must be provided that describes the database model implemented by the DBMS used.

The type of database implemented in all iSeries 400 systems is an integrated relational database. The database is integrated into the OS/400 operating system.



## 5.4 Partitions/Replications Mapping

The mapping of database partitions/replications must be explicitly described.

Horizontal partitioning was implemented on all the files except ITEM. The partitioning was done based on the Warehouse ID key field. The following eight tables: Warehouse, District, Customer, History, Neworder, Orders, Orderline, and Stock were split such that records for Warehouse ID's 1 through 6,100 were in the first partition and 6,101 through 12,200 were in the second partition.

# 6.0 Clause 5: Performance Metrics and Response Time - Related Items

## 6.1 Response Times

Ninetieth percentile, maximum and average response times must be reported for all transaction types as well as for the Menu response time.

Table 3 lists the response times and the ninetieth percentiles for each of the transaction types for the IBM ~ iSeries 400 Model 840-2420 system.

## 6.2 Keying and Think Times

The minimum, the average, and the maximum keying and think times must be reported for each transaction type.

Table 3 lists the keying and think times from the measured TPC-C tests for the IBM iSeries 400e Model 840-2420.

**Table 3. IBM ~ iSeries 400 Model 840-2420 - Response, Keying, and Think Times**

Response Times	NewOrder	Payment	Order Status	Delivery (int/def)	Stock Level	Menus
90%	0.5	0.3	0.5	0.2/2.1	1.8	0.1
Average	0.32	0.20	0.30	0.10/1.70	0.82	0.10
Maximum	9.97	4.99	2.76	0.19/6.18	6.00	0.37
			<b>Think Times</b>			
Minimum	0.00	0.00	0.00	0.00	0.00	N/A
Average	12.01	12.03	10.01	5.05	5.03	N/A
Maximum	120.23	120.23	100.21	50.22	50.21	N/A
			<b>Keying Times</b>			
Minimum	18.00	3.00	2.00	2.00	2.00	N/A
Average	18.00	3.00	2.00	2.00	2.00	N/A
Maximum	18.23	3.17	2.10	2.07	2.21	N/A

## 6.3 Response Time Frequency Distribution

Response time frequency distribution curves must be reported for each transaction type.

### iSeries 400 Model 840-2420 New-Order Response Time Distribution

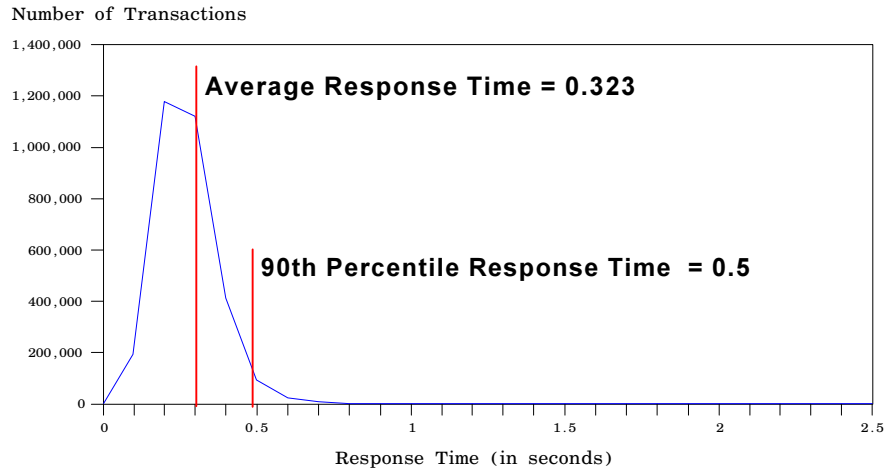


Figure 1. iSeries 400 Model 840-2420 New-Order Response Time Distribution

### iSeries 400 Model 840-2420 Payment Response Time Distribution

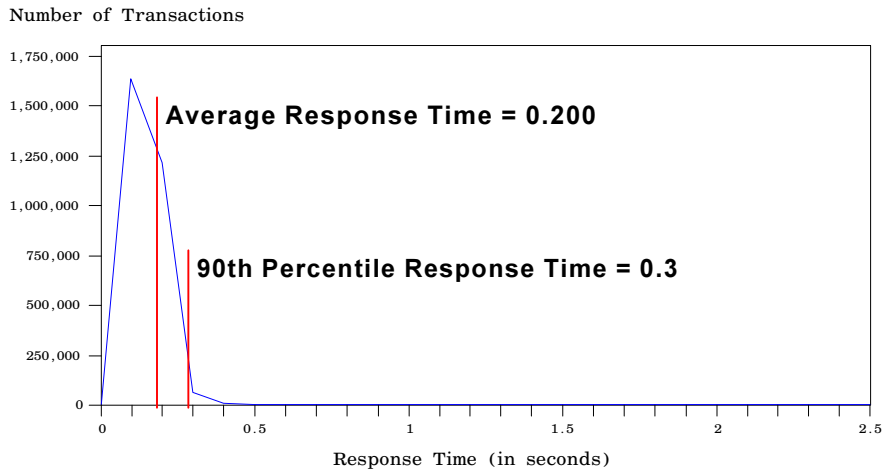


Figure 2. iSeries 400 Model 840-2420 Payment Response Time Distribution

### iSeries 400 Model 840-2420 Order Status Time Distribution

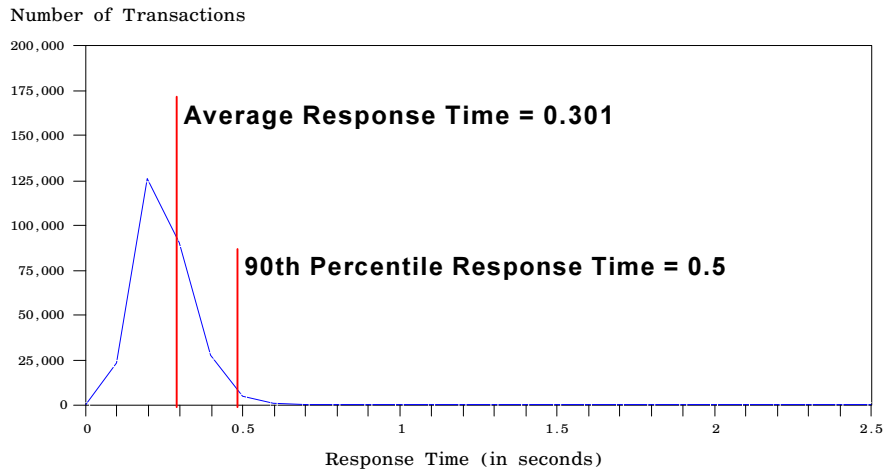


Figure 3. iSeries 400 Model 840-2420 Order Status Response Time Distribution

### iSeries 400 Model 840-2420 Delivery Response Time Distribution (Interactive)

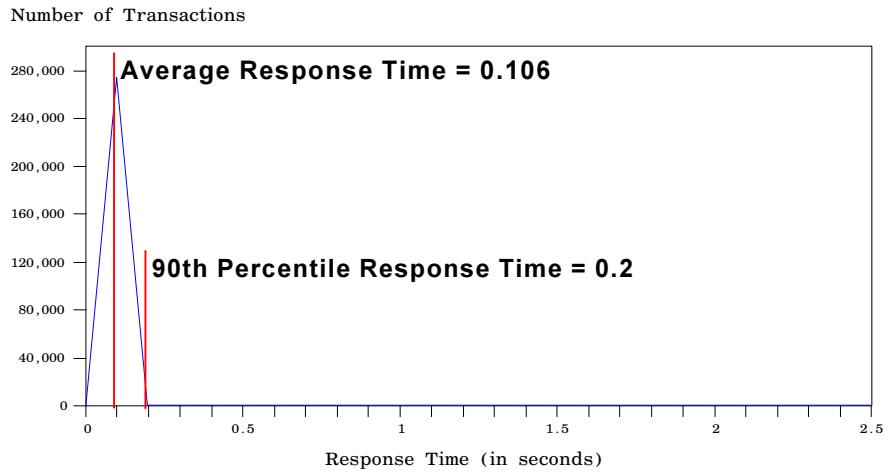


Figure 4. iSeries 400 Model 840-2420 Delivery (Interactive) Response Time Distribution

### iSeries 400 Model 840-2420 Delivery Response Time Distribution (Batch)

Number of Transactions

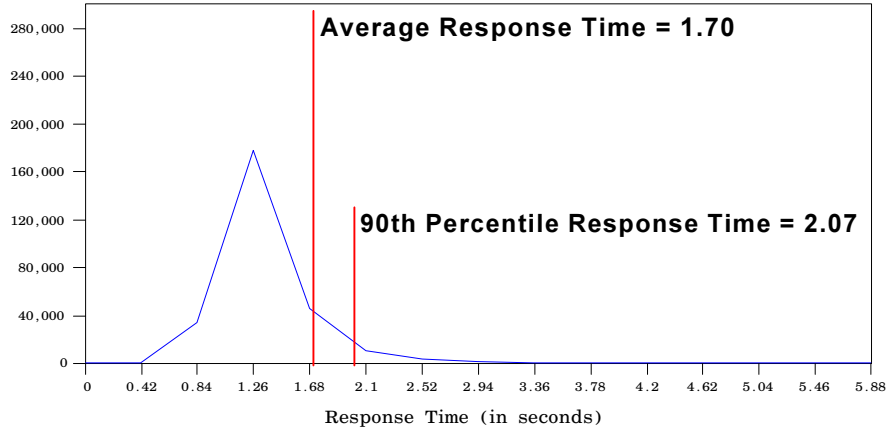


Figure 5. iSeries 400 Model 840-2420 Delivery (Batch) Response Time Distribution

### iSeries 400 Model 840-2420 Stock-Level Time Distribution

Number of Transactions

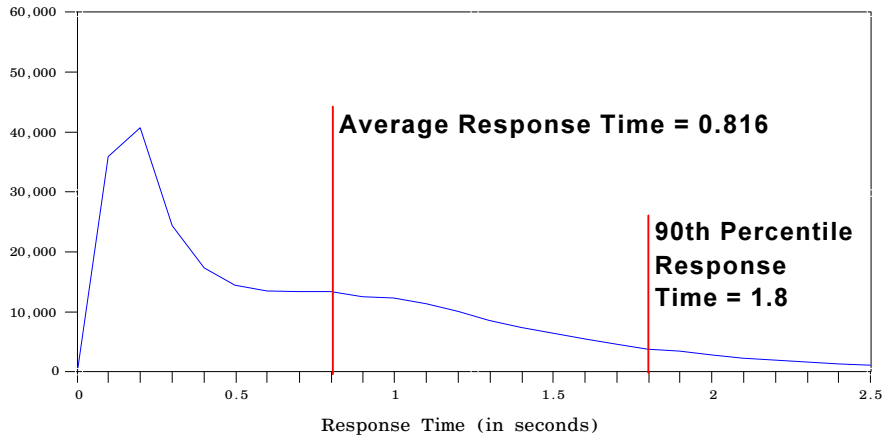


Figure 6. iSeries 400 Model 840-2420 Stock-Level Response Time Distribution

## 6.4 Performance Curve for Response Time versus Throughput

The performance curve for response times versus throughput must be reported for the New-Order transaction.

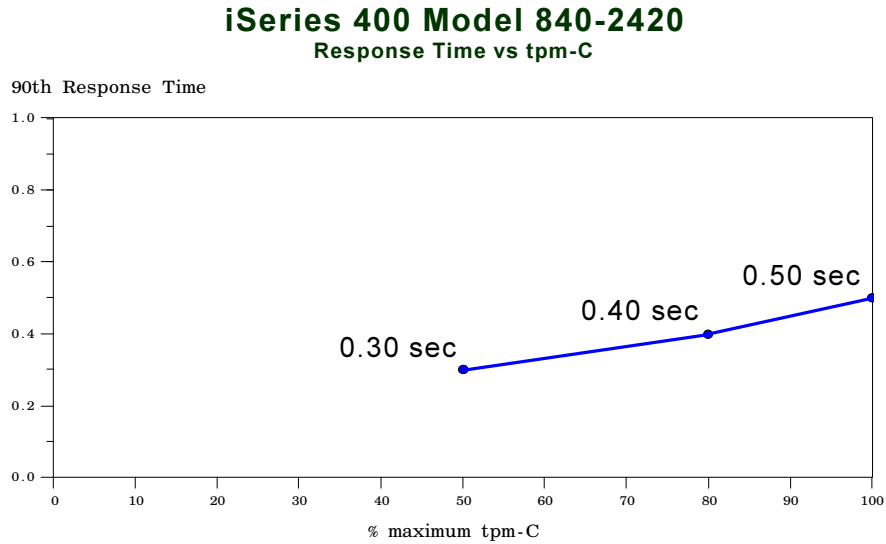


Figure 7. iSeries 400 Model 840-2420 New-Order Response Time Versus Throughput

## 6.5 Think Time Frequency Distribution

Think time frequency distribution curves must be reported for each transaction type.

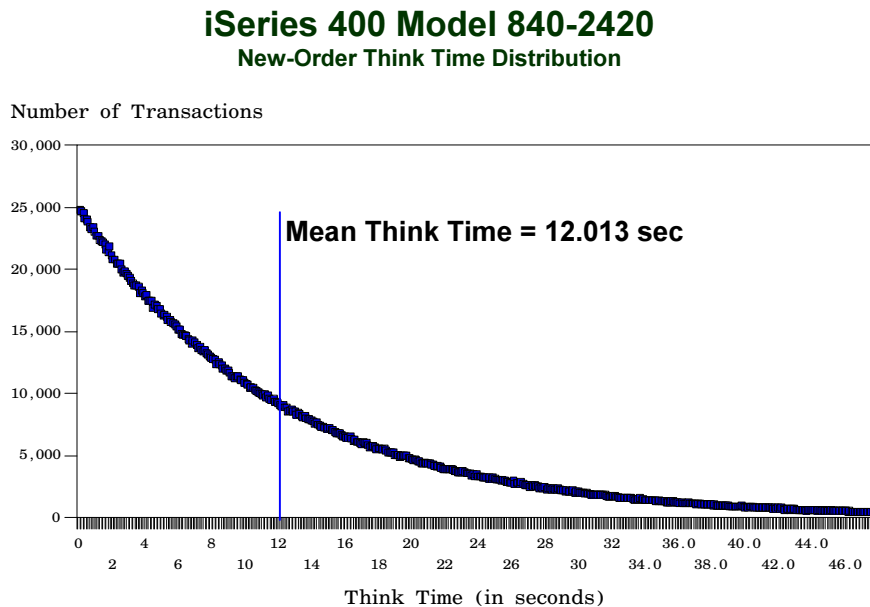


Figure 8. iSeries 400 Model 840-2420 New-Order Think Time Distribution

## 6.6 Throughput Versus Elapsed Time

*A graph of throughput versus elapsed time must be reported for each the New-Order transaction.*

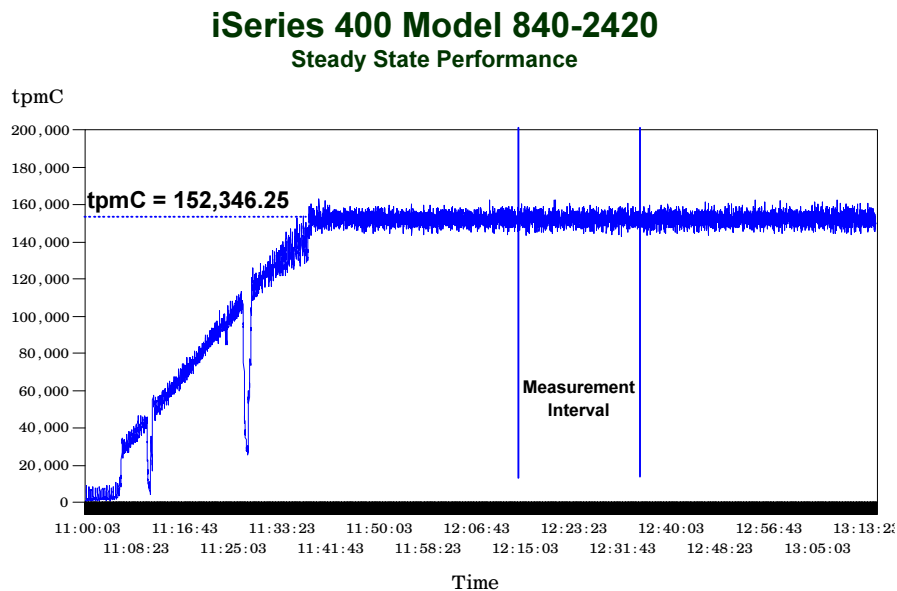


Figure 9. iSeries 400 Model 840-2420 New-Order Throughput Versus Elapsed Time

## 6.7 Work Performed During Steady State

*A description of how the work normally performed during a sustained test (for example, checkpointing, writing redo/undo log records, etc.) actually occurred during the measurement interval must be reported.*

For each of the TPC Benchmark™ C transaction types the following steps are executed:

- At the database transaction start, a “start of commit cycle” entry is recorded in the journal.
- For each of the files updated by a transaction:
  - The database record being updated is locked by the transaction preventing further updates or reads until the journal records are written.
  - A before image of the record is written to the journal receiver.
  - An after image of the record is written to the journal receiver.
- At the end of the database transaction, the records are committed in the journal and all locks on the database records by the transaction are released.

Recording a block of journal entries does not correspond directly to a disk write, since the iSeries 400 journal management function has the ability to block journal writes from one or more jobs into a single physical I/O.

The iSeries 400 system’s integrated relational database does not require an overt checkpointing system to ensure that data is written to disk. The iSeries 400 system’s standard journal management function ensures that data in disk files is synchronized with that in memory in a transparent, non-disruptive fashion.

As database I/Os are committed, the journal entries associated with the change are written prior to the completion of the commitment function. The database I/Os are issued as asynchronous I/Os which may be delayed by other requests to use the same data. To ensure that all data updates are completed in a reasonable period of time, iSeries 400 system's journal management ensures that unwritten pages from all tables being journaled are forced to disk at least once for every 17,000,000 entries to the journal. That is, for the 17 tables in TPC-C, one table is forced to disk every 1,000,000 journal entries, so that all 17 tables are synchronized within the period of time it takes to log 17,000,000 journal entries.

The iSeries 400 Model 840-2420 system operating at 152,346.25 tpm-C completes 17,000,000 entries to the journal in approximately 3.5 minutes. A minimum measurement interval of 20 minutes includes approximately 6 complete cycles.

## 6.8 Reproducibility

*A description of the method used to determine the reproducibility of the measurement results must be reported.*

A repeatability measurement was taken on the IBM ~ iSeries 400 Model 840-2420 for the same length of time as the measured run. The repeatability interval was taken from the same measurement as the reported interval and the intervals were separated by 8 minutes and 54 seconds. The repeatability measurement was 152,344.15 tpmC.

## 6.9 Measurement Interval

*A statement of the duration of the measurement interval for the reported Maximum Qualified Throughput (tpmC) must be included.*

The measurement interval for the tpmC reported was for 20 minutes in duration. The reproducibility measurement was also for a 20-minute interval.



# **7.0 Clause 6: SUT, Driver, and Communication Definition-Related Items**

## **7.1 RTE Availability**

If the RTE is commercially available, then its inputs must be specified. Otherwise, a description must be supplied of what inputs to the RTE had been used.

Appendix E contains the scripts used in the Remote Terminal Emulator testing.

## **7.2 Functionality and Performance of Emulated Components**

It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system.

In the benchmark configuration the Remote Terminal Emulator (RTE) communicates with the client system over Ethernet. The communications mechanism used in the benchmarked and priced configurations are the same.

## **7.3 Network Bandwidth**

The bandwidth of the network(s) used in the tested/priced configuration must be disclosed.

RTE network used 10 MB/second Ethernet. The tested configuration of front-end iSeries 400 systems and RTEs was physically connected on a 10/100 Ethernet switched network. The tested configuration of front-end iSeries 400 systems and backend iSeries 400 system was physically connected with a 100 mbps Ethernet switched network.

The priced configuration observed the physical limitation of 1024 physical devices connected to a single Ethernet segment.

## **7.4 Operator Intervention**

If the configuration requires operator intervention, the mechanism and the frequency of this intervention must be disclosed.

The iSeries 400 configurations reported do not require any operator intervention to sustain the reported throughput during the 8-hour period.

## 8.0 Clause 7: Pricing - Related Items

### 8.1 Hardware and Programs Used

*A detailed list of the hardware and software used in the priced system must be reported. Each item must have vendor, part number, description, and release/revision level, and either general availability status or committed delivery date. If package-pricing is used, contents of the package must be disclosed. Pricing source(s) and effective date(s) must also be reported.*

The detailed list of all hardware and programs for the priced configuration is listed in the pricing sheets (please refer to Section 8.3.1 for details) for each system reported. The prices for all products and features are provided by IBM and available the same day as product or feature availability. All products are currently orderable for delivery on or before the published availability date.

### 8.2 Five Year Cost of System Configuration

*The total 5-year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.*

The price sheet for the iSeries 400 Model 840-2420 and associated client systems is contained on the following pages. The discounts used are disclosed below.

#### Revenue Allowance

This allowance of 22% was based on total hardware and software purchase price of the configuration.

#### 5-year Term Maintenance Contract Discount

This discount is available for customers who sign a 5-year maintenance agreement on the hardware. A discount of 8% is available for customers when they sign a 5-year maintenance agreement.

#### 5-year Maintenance Prepay Discount

This is a discount for prepayment of maintenance costs. A discount of 15.05% is available for this configuration based on payment for five years maintenance at time of purchase. This discount is applied to the balance after the 5-year term maintenance contract discount is applied.

#### Software Support Discount For Secondary Systems

This is a discount for multiple systems supported in a single I/S shop. The maintenance for secondary systems is heavily discounted and is available at 15% of the price of maintenance of a single system.

### 8.3 Statement of tpmC and Price/Performance

A statement of the measure tpmC, as well as the respective calculations for 5-year pricing, price/performance (price/tpmC), and the availability date must be disclosed.

The IBM ~ iSeries 400 Model 840-2420 was measured at 152,346.25 tpmC with a 5-year system price of \$7,846,074. The respective price-performance for the iSeries 400 Model 840-2420 is \$51.50 per tpmC. The iSeries 400 Model 840-2420 priced configuration is currently orderable and deliverable.

# 8.3.1 IBM ^ iSeries 400 Model 840-2420 Five-Year System Price Configuration

IBM		IBM @server iSeries 400 Model 840-2420		TPC-C REV 3.5 EXECUTIVE SUMMARY			
				Report Date: October 24, 2000			
<b>iSeries 400 Model 840-2420 Config</b>							
Item Description	Feature Number	Third Party Pricing	Unit Price	Quantity	Extended Price	Extended Maintenance	5-Year Price
<b>Server Hardware:</b>							
iSeries 400 Model 840 24-way Base System Unit	9406-840		640,000	1	640,000	84,816	724,816
Processor - ISTAR 7S 500 MHz 24-way, Cache 8x4 MB	2420		550,000	1	550,000	92,928	642,928
Interactive Card	1540		0	1	0	0	0
Battery Backup	Bundled			1	0	0	0
Twinax Work Station Controller (5540)	Bundled			1	0	0	0
200V 14ft Locking Power Cord	Bundled			1	0	0	0
Line Cord 200V 14 ft Locking for CEC	Bundled			1	0	0	0
8m HSL Cable - Base I/O Tower	1461	550		20	11,000		11,000
SPCN Cables - 6m SPCN Cable (1484)	Bundled			24	0	0	0
8192 MB (RIVER - 256 MB technology)	3196	147,456		16	2,359,296		2,359,296
Programmable Regulator (req'd with 3196)	2730	750		2	1,500		1,500
PCI Ultra Magnetic Media Controller	2749	1,300		1	1,300		1,300
PCI IOP with 64 MB memory	2843	1,925		22	42,350		42,350
17.6 GB 10K RPM Disk Unit	4318	2,520		769	1,937,880		1,937,880
CD-ROM	4425	415		1	415		415
PCI Two Line WAS IOA (ECS) req'd with 5540	4745	425		1	425		425
PCI Twinaxial IOA - Req'd with 5540	4746	750		1	750		750
PCI RAID Disk Unit Controller with 26 MB write cache	4748	6,000		52	312,000		312,000
PCI 100/10 mbps Ethernet IOA	4838	900		20	18,000		18,000
PCI 100/16/4 mbps Token Ring IOA	2744	840		2	1,680		1,680
PCI Expansion Tower / Mantis	5074	17,900		17	304,300	282,336	586,636
PCI Expansion Tower	5101	9,000		18	162,000	357,696	519,696
25 GB 1/4-Inch Tape Drive	4486	6,000		1	6,000		6,000
V.24/EIA232 50ft PCI Cable - System Console	0348	125		1	125		125
<b>Server Subtotal</b>					<b>6,349,021</b>	<b>817,776</b>	<b>7,166,797</b>
<b>Client Hardware:</b>							
iSeries 400 Model 270 Base System Unit	9406-270		4,000	1	4,000	4,416	8,416
Twinax Work Station Controller	Bundled			1	0	0	0
V.24/EIA 232 20ft PCI Cable for System Console	0348	125		1	125		125
125V 6 ft Line Cord	Bundled			1	0	0	0
System Expansion Tower - 200V 6ft Locking Cord	Bundled			1	0	0	0
8m HSL Cable - Base I/O Tower	1461	550		2	1,100		1,100
Processor Card	2252	16,000		1	16,000	7,824	23,824
Interactive Card	1516	0		1	0	0	0
PCI IOP with 32 MB memory	2842	1,800		3	5,400		5,400
Main Store - memory riser card - 16 slots	2884	2,200		1	2,200		2,200
512 MB DIMMS Main Storage	3025	2,048		16	32,768		32,768
8.6 GB 10K RPM Disk Unit	4317	1,400		6	8,400		8,400
CD-ROM	4525	415		1	415		415
4 GB 1/4-Inch Tape Drive	4582	1,300		1	1,300		1,300
PCI Expansion Tower	5075	6,000		1	6,000	5,856	11,856
PCI Two Line WAS IOA (ECS) req'd with 5540	4745	425		1	425		425
PCI Twinaxial IOA - Req'd with 5540	4746	750		1	750		750
PCI RAID Disk Unit Controller with 26 MB write cache	4748	6,000		2	12,000		12,000
PCI 100/10 mbps Ethernet IOA	4838	900		3	2,700		2,700
<b>Single Client Subtotal</b>					<b>93,563</b>	<b>18,096</b>	<b>111,679</b>
Number of Clients				16			
<b>Client Subtotal</b>					<b>1,497,328</b>	<b>289,538</b>	<b>1,786,864</b>
<b>Server Software:</b>							
IBM Operating System/400 V4R5M0	Bundled			1	0	41,164	41,164
DB2 Query Manager Dev. Toolkit	QU1 and ST1	28,800		1	28,800		28,800
<b>Client Software:</b>							
IBM Operating System/400 V4R5M0	Bundled		Bundled	16	0	310,263	310,263
BEA Tuxedo V6.4	1	3,000		16	48,000	38,400	86,400
ILE COBOL	5769CB1	2,400		1	2,400		2,400
Application Development Toolkit	5769PW1	3,020		1	3,020		3,020
ILE C	5769CX2	2,400		1	2,400		2,400
DB2 Query Mgr and SQL Development Kit	5769ST1	1,600		1	1,600		1,600
<b>Software Subtotal</b>					<b>86,220</b>	<b>389,827</b>	<b>476,047</b>
<b>User Connectivity:</b>							
24-port 10/100 Ethernet Switch (2 spares)	2	975		20	19,505		19,505
8-Port 10Mb Hub (10% spares)	2	29		22315	637,093	0	637,093
<b>User Connectivity Subtotal</b>					<b>656,598</b>	<b>0</b>	<b>656,598</b>
<b>5-year System Subtotal</b>					<b>8,589,167</b>	<b>1,497,139</b>	<b>10,086,306</b>
<b>Discounts:</b>							
			% Allowance		Volume	Discount	
Revenue Allowance			22		7,884,568	(1,734,605)	
5-year Term Maintenance Contract Discount			8		1,107,312	(88,585)	
5-year Maintenance Prepayment Discount			15.05		1,018,727	(153,318)	
Software Support Discount For Secondary Systems			85		310,262	(263,723)	
5-year System Total					6,854,562	991,512	7,846,074
tpmC							152,346.25
\$/tpmC							51.50
Note: All pricing is from IBM except items noted in Third Party Pricing column. 1. BEA Systems, Inc.; 2. Venture Computer Systems							
<b>Notes:</b>				<b>Five-Year Cost of Ownership: \$7,846,074</b>			
Server Hardware requires no charge RPQ 847109				<b>tpmC Rating: 152,346.25</b>			
Revenue Allowance is applied to hardware and software for the priced configuration.				<b>\$/tpmC: \$51.50</b>			
5-Year Term Maintenance Contract Discount is given when 5-year contract is signed.							
5-Year Maintenance Prepay Discount is given when 5-year maintenance costs are prepaid.							
Software Support Discount For Secondary Systems is given for maintenance costs on Multiple systems in a single I/S shop.							
Results audited by Francois Raab of InfoSizing Inc.							
Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org.							

## **9.0 Clause 8: Audit - Related Items**

*If the benchmark has been independently audited, then the auditor's name, address, phone number, and a brief audit summary report indicating compliance must be included in the Full Disclosure Report. A statement should be included, specifying when the complete audit report will become available and who to contact in order to obtain a copy.*

These TPC Benchmark™ C results have been audited by François Raab of InfoSizing, Inc. The attestation letter is included at the end of this report.

# Appendix A. System Parameters and User Profile

## A.1 System Parameters

Table 4 shows the system parameters changed for these TPC-C measurements.

**Table 4. iSeries 400 System Parameters**

<b>System Parameters</b>	<b>Modified value</b>	<b>Shipped value</b>
QABNORMSW	1	0
QACTJOB	5000	20
QADLACTJ	1000	10
QADLTOTJ	1000	10
QASTLVL	*INTERMED	*BASIC
QAUTOVRT	500	0
QBASACTLVL	32767	6
QCMNRCYLMT	7 1	0 0
QCONSOLE	DSP01	QCONSOLE
QCTLSBSD	QSYS/QCTL	QSYS/QBASE
EDTRCYAP	*OFF	50
QHSTLOGSIZ	32767	5000
QINACTMSGQ	*DSCJOB	*ENDJOB
QJOBMSGQFL	*PRTWRAP	*NOWRAP
QJOBMSGQMX	64	16
QJOBMSGQSZ	16384	16
QJOBMSGQTL	16384	24
QJORECRA	17000000	50000
QLMTSECOFR	0	1
QMAXSIGN	5	3
QPFRADJ	0	2
QPWDEXPITV	186	*NOMAX
QPWDMAXLEN	10	8
QPWDRQDDGT	1	0
QPWDRQDDIF	5	0
QRCLSPSTG	*NOMAX	8
QSECURITY	20	40
QTOTJOB	5000	30
QUTCOFFSET	-500	0

## A.2 Transaction Subsystem Description

The memory on the system was divided into separate pools as shown below.

```
Work with Shared Pools Print Panel                               Page 1
5769SS1 V4R5M0 000526
WILBER 06/15/00 15:24:40
Main storage size (M) . : 131072.00
Pool      Defined      Max      Allocated  Pool  --Paging Option--
Pool      Size (M)    Active   Size (M)   ID    Defined  Current  Text
*MACHINE   7371.6     +++++   7371.6     1     *FIXED  *FIXED  Machine pool
*BASE      37200.3    32767   37200.3    2     *FIXED  USRDFN  Base pool
*SHRPOOL1  14900.0    32767   14900.0    3     *FIXED  USRDFN
*SHRPOOL2  14900.0    32767   14900.0    4     *FIXED  USRDFN
*SHRPOOL3  14900.0    32767   14900.0    5     *FIXED  USRDFN
*SHRPOOL4  14900.0    32767   14900.0    6     *FIXED  USRDFN
*SHRPOOL5  5000.0     32767   5000.0     7     *FIXED  USRDFN
*SHRPOOL6  14900.0    32767   14900.0    8     *FIXED  USRDFN
*SHRPOOL7  7000.0     32767   7000.0     9     *FIXED  USRDFN
* * * * * E N D   O F   L I S T I N G   * * * * *
```

# Appendix B. Database File Definitions

## B.1 AREFFIL: Reference File

```

A* OCO Source Materials
A* The Source code for this program is not published or otherwise
A* divested of its trade secrets, irrespective of what has been
A* deposited with the U.S. Copyright office
A* (C) Copyright IBM Corp. 1993, 1996, 1998

A*B:H2:AREFFIL: Reference File
      R REFRCO      TEXT('TPCC Reference File')
A*This file contains the field definition for most of the other
A*files used in the TPC-C benchmark.  For example to define the
A*warehouse ID field, both the warehouse file (WRHS) and the
A*district file (DSTRCT) reference WRHSID in this file.  This
A*helps when field definitions need to change, the change only
A*needs to be made in one place.
A*
A*UPDATED:10/23/93
A*Changes BALANC and CRDLMT to 12 digit fields as required
A*by the TPC-C V1.1 Specification
A*UPDATED:03/04/96
A*Changes BALANC and CRDLMT to 13 digit fields for efficiency.
A*Added ITMIMAGE perf v3.0 spec.  Converted many fields to binary.
A* wrhsid, distid, custid, itemid, ordid, oline, ytd2, qty2, qty4
A*UPDATED:03/31/98
A*
A*Code Identifiers
A* Note Binary numbers are 4B for two byte shorts and 9B
A* for long integer 32 byte numbers conforms to SQL definitions
A*
A      WRHSID      4B      TEXT('Warehouse ID')
A      COLHGD('W/H ID')
A      DISTID      4B      TEXT('District ID')
A      COLHGD('District')
A      EDTCDE(1)
A      CUSTID      9B      TEXT('Customer ID')
A      COLHGD('Customer')
A      ITEMID      9B      TEXT('Item ID')
A      COLHGD('Item')
A      ITMIMAGE     9B      TEXT('ImageID')
A      COLHGD('Image#')
A      ORDID       9B      TEXT('Order ID')
A      COLHGD('Order #')
A      EDTCDE(4)
A*      OLINE       3P      TEXT('Order Line Number')
A      COLHGD('Order' 'Line #')
A*Name, Address and Descriptor Information
A      LONCNAM     10      TEXT('Name of W/H or District')
A      COLHGD('Location' 'Name')
A      FNAME       16      TEXT('First Name')
A      COLHGD('First' 'Name')
A      MINIT       2       TEXT('Middle Initial')
A      COLHGD('Middle' 'Init')
A      LNAME       16      TEXT('Last Name')
A      COLHGD('Last' 'Name')
A      ADDR1       20      TEXT('Address Line 1')
A      COLHGD('Address' 'Line 1')
A      ADDR2       20      TEXT('Address Line 2')
A      COLHGD('Address' 'Line 2')
A      CITY        20      TEXT('City')
A      STATE       2       TEXT('State')
A      ZIPCD       9A      TEXT('Zip Code')
A      COLHGD('Zip' 'Code')
A      PHONE       16      TEXT('Phone Number')
A* Item name change to 20 characters 3/31/98 Chuck Kingsbury
A* VERSION4      20      TEXT('Item Name')
A      ITMNAM      24      TEXT('Item Name')
A      COLHGD('Item' 'Name')
A      CARRID      2       TEXT('Carrier ID')
A      COLHGD('Carrier' 'Number')
A      LOCAL      1 0     TEXT('Flag to indicate local')
A*Financial and Inventory Numbers
A      TAX         5 4     TEXT('Tax Percentage')
A      EDTWRD('0. ')
A      DSCNT       5 4     TEXT('Discount Percentage')
A      COLHGD('Discount')
A      EDTWRD('0. ')
A      BALANC      13 2    TEXT('Balance Information')
A      COLHGD('Balance')
A      EDTWRD(' $0 . -')
A      CREDIT      2       TEXT('Credit Status-GC or BC')
A      COLHGD('Credit' 'Status')
A      CRDLMT      13 2    TEXT('Credit Limit')
A      COLHGD('Credit' 'Limit')
A      EDTWRD(' $0 . -')
A      AMOUNT7     7P 2    TEXT('Amount')
A      EDTWRD(' $0 . - ')
A      YTD         13 2    TEXT('YTD Amount')
A      EDTWRD(' $0 . - ')
A      YTD2        9B      TEXT('YTD Amount')
A      EDTCDE(4)
A      QTY2        3P 0    TEXT('Quantity')
A      COLHGD('Quantity')
A      EDTCDE(1)
A      QTY4        3P 0    TEXT('Quantity')
A      COLHGD('Quantity')
A      EDTCDE(1)
A      QTY3        3 0     TEXT('Quantity')
A      COLHGD('Quantity')
A      EDTCDE(1)
A      PRICE       5 2     TEXT('Price')
A      EDTWRD('$0 . -')
A*Date and Time Information
A      TIMEDATE    8       TEXT('Internal Time Date')
A      DATE        8S      TEXT('Date')
A      EDTWRD(' - - ')
A      TIME        6S      TEXT('Time')
A      EDTWRD(' : : ')
A*Pad Information
A      CDATA       500     TEXT('Customer Data')
A      COLHGD('Cust' 'Filler')
A      HDATA       24      TEXT('History Information')
A      COLHGD('Hist' 'Filler')
A      IDATA       50      TEXT('Item Data')
A      COLHGD('Item' 'Filler')
A      DISTINFO    24      TEXT('Dist Info')
A      COLHGD('Dist' 'Info')
A*TPCC - Plus Information
A      CPHRNC      10      COLHGD('Phonetic Search')
A      DSPDDID     2 0     TEXT('District ID')
A      COLHGD('District')
A      EDTCDE(4)
A      DSPOID      8 0     TEXT('Order ID')
A      COLHGD('Order #')
A      EDTCDE(4)
A      DSPOLN     2 0     TEXT('Order Line Number')
A      COLHGD('Order' 'Line #')
A      AMOUNT6    6 2     TEXT('Amount')
A      EDTWRD('$0 . -')
A      DSPQTY     3S 0     TEXT('Quantity')

```

```

A      COLHGD('Quantity')
A

```

## B.2 CSTMRLFCRT: Customer Logical File

```

A* CSTMRLFCR: Customer Logical File      Split DB version
A*
A      R CSRCO      PFILE(CSTMRF01 CSTMRF02)
A      K CWID
A      K CDID
A      K CLAST
A      K CFIRST
A      K CID

```

## B.3 CSTMRLFNAM: Customer Names Logical File

```

A* CSTMRLFNA: Customer File Names Logical File      SPLIT DB version
A*
A      R CSRCO      PFILE(CSTMRF01 CSTMRF02)
A      K CWID
A      K CDID
A      K CLAST

```

## B.4 CSTMR: Customer Logical File

```

A* CSTMR: Customer File      SPLIT DB version
A*
A* This is the file definition for the customer data base file.
A* All fields that are stored in the customer file are defined
A* here, the actual field definitions are in AREFFIL (the
A* reference file) and are pulled into this file when it is
A* compiled (CRTPF command executed against this source).
A* The UNIQUE keyword guarantees that there will be no duplicate
A* key values inserted into the file.
A
A      UNIQUE
A      PFILE(CSTMRF01 CSTMRF02)
A      TEXT('CUSTOMER MASTER FILE - TPCC')
A      R CSRCO
A      K CID
A      K CDID
A      K CWID

```

## B.5 CSTMRPF: Customer Physical File

```

A*B:H2:CSTMRPF: Customer File      REF(AREFFIL)
A*
A* This is the file definition for the customer data base file.
A* All fields that are stored in the customer file are defined
A* here, the actual field definitions are in AREFFIL (the
A* reference file) and are pulled into this file when it is
A* compiled (CRTPF command executed against this source).
A*
A* NOTE: in COMMONTPCC H file is a #define for the record length.
A* Changes in record length need to be updated in the H file also.
A*
A      R CSRCO      TEXT('CUSTOMER MASTER FILE - TPCC')
A      CID          R      REFFLD(CUSTID)
A      CDID         R      REFFLD(DISTID)
A      CWID         R      REFFLD(WRHSID)
A      CFIRST      R      REFFLD(FNAME)
A      CINIT       R      REFFLD(MINIT)
A      CLAST       R      REFFLD(LNAME)
A      CLTOD       R      REFFLD(TIMEDATE)
A      COLHGD('Date of' 'Last Order')
A      CADDR1      R      REFFLD(ADDR1)
A      CCREDIT     R      REFFLD(CREDIT)
A      COLHGD('Credit' 'Status')
A      TEXT('Credit Status')
A      CADDR2      R      REFFLD(ADDR2)
A      CDCY        R      REFFLD(DSCNT)
A      CCITY       R      REFFLD(CITY)
A      CSTATE     R      REFFLD(STATE)
A      CZIP        R      REFFLD(ZIPCD)
A      CPHONE     R      REFFLD(PHONE)
A      CBAL        R      REFFLD(BALANC)
A      TEXT('Customer Balance')
A      CCRDLM     R      REFFLD(CRDLMT)
A      TEXT('Credit Limit')
A      CYTD        R      REFFLD(YTD)
A      TEXT('Customer YTD')
A      CPAYCNT     R      REFFLD(QTY4)
A      TEXT('Customer Payments')
A      CDELCNT    R      REFFLD(QTY4)
A      TEXT('Customer Deliveries')
A      CLTIME     R      REFFLD(TIME)
A      COLHGD('Time of' 'Last Order')
A*      CDATA       R      TEXT('Time of DB Build')
A

```

## B.6 DSTRCTLF: District Logical File

```

A* DSTRCTLF: District File (SPLIT DB version)
A*
A      R DSRCD      PFILE(DSTRCT01 DSTRCT02)

```



K DID  
K DWID

## B.7 DSTRCT: District Physical File

```
A*B:H2.DSTRCT: District File REF(AREFFIL)
A*This is the file definition for the district data base file.
A*All fields that are stored in the district file are defined
A*here, the actual field definitions are in AREFFIL (the
A*reference file) and are pulled into this file when it is
A*compiled (CRTPF command executed against this source).
A*The UNIQUE keyword guarantees that there will be not duplicate
A*key values inserted into the file.
A*
A      R DSRCO      TEXT('District Master File - TPCC')
A      DID          REFFLD(DISTID)
A      DWID         REFFLD(WRHSID)
A      DNAME        REFFLD(LOCNAM)
A      COLHGD('District' 'Name')
A      DADDR1       REFFLD(ADDR1)
A      DADDR2       REFFLD(ADDR2)
A      DCITY         REFFLD(CITY)
A      DSTATE        REFFLD(STATE)
A      DZIP          REFFLD(ZIPCD)
A      DTRK          REFFLD(TRX)
A      DVTD          REFFLD(VTID)
A      COLHGD('YTD' 'Balance')
A      TEXT('YTD Balance')
A      REFFLD(ORDID)
A      DNXTOR        REFFLD(ORDID)
A      COLHGD('Next' 'Order #')
A      TEXT('Next Order Number')
A      K DID
A      K DWID
```

## B.8 HSTRY: History Logical File

```
A* HSTRYL: History file (SPLIT DB version)
A*
A      R HSRCD      PFILE(HSTRY01 HSTRY02)
A      K HDID
A      K HWID
A      K HCID
```

## B.9 HSTRYLF: History Logical File

```
A*B:H2.HSTRY: History File REF(AREFFIL)
A      R HSRCD      TEXT('History File for TPCC')
A*This is the file definition for the history data base file.
A*All fields that are stored in the history file are defined
A*here, the actual field definitions are in AREFFIL (the
A*reference file) and are pulled into this file when it is
A*compiled (CRTPF command executed against this source).
A*The UNIQUE keyword is not needed in this file because there
A*are no key fields for this file.
A*
A      HDID         REFFLD(DISTID)
A      HWID         REFFLD(WRHSID)
A      HCID         REFFLD(CUSTID)
A      HCOYD        REFFLD(DISTID)
A      HCWID        REFFLD(WRHSID)
A      HTOD         REFFLD(TIMEDATE)
A      COLHGD('Payment' 'Time Date')
A      HTIME        REFFLD(TIME)
A      HAMT         COLHGD('Payment' 'Time')
A      REFFLD(AMOUNT7)
A      COLHGD('Payment' 'Amount')
A      TEXT('Payment Amount')
A      HDATA        REFFLD(HDATA)
```

## B.10 ITEM: Item Physical File

```
A*B:H2.ITEM: Item File REF(AREFFIL)
A*This is the file definition for the item data base file.
A*All fields that are stored in the item file are defined
A*here, the actual field definitions are in AREFFIL (the
A*reference file) and are pulled into this file when it is
A*compiled (CRTPF command executed against this source).
A*The UNIQUE keyword guarantees that there will be not duplicate
A*key values inserted into the file.
A*
A      R ITRCD      TEXT('Item File for TPCC')
A      IID          REFFLD(ITEMID)
A      IMAGEID       REFFLD(ITEMIMAGE)
A      INAME         REFFLD(ITEMNAM)
A      IPRICE        REFFLD(PRICE)
A      IDATA         REFFLD(PRICE)
A      K IID
```

## B.11 NEWORD: New Order Logical File

```
A* NEWORD: New Order File (SPLIT DB version)
A*
A* This is the file definition for the new order data base file.
A* All fields that are stored in the new order file are defined
A* here, the actual field definitions are in AREFFIL (the
A* reference file) and are pulled into this file when it is
A* compiled (CRTPF command executed against this source).
A* The UNIQUE keyword guarantees that there will not be duplicate
A* key values inserted into the file.
A      R NORCD      PFILE(NEWORDPF01 NEWORDPF02)
```

A K NOWID  
A K NODID  
A K NOOID

TEXT('New Orders File - TPCC')

## B.12 NEWORDLF: New Order Logical File

```
A* NEWORDLF: New Order File Logical file (Split DB version)
A*
A      R NORCD      PFILE(NEWORDPF01 NEWORDPF02)
A      K NOWID  
A      K NODID
```

## B.13 NEWORDPF: New Order Physical File

```
A*B:H2.NEWORDPF: New Order File REF(AREFFIL)
A*This is the file definition for the new order data base file.
A*All fields that are stored in the new order file are defined
A*here, the actual field definitions are in AREFFIL (the
A*reference file) and are pulled into this file when it is
A*compiled (CRTPF command executed against this source).
A      R NORCD      TEXT('New Orders File - TPCC')
A      NOOID        REFFLD(ORDID)
A      NODID        REFFLD(DISTID)
A      NOWID        REFFLD(WRHSID)
```

## B.14 ORDERS: Orders Logical File

```
A* ORDERS: Orders File (Split DB)
A*
A* This is the file definition for the orders data base file.
A* All fields that are stored in the orders file are defined
A* here, the actual field definitions are in AREFFIL (the
A* reference file) and are pulled into this file when it is
A* compiled (CRTPF command executed against this source).
A*
A      R ORRCD      PFILE(ORDERSPF01 ORDERSPF02)
A      OWID         REFFLD(ORDID)
A      ODID         REFFLD(CUSTID)
A      OCID         REFFLD(ORDID)
A      OID          REFFLD(DISTID)
A      OENTTOD       REFFLD(TIMEDATE)
A      OENTTM        REFFLD(TIME)
A      OCARID        REFFLD(CARRID)
A      OLINES        REFFLD(OLINE)
A      OLOCAL        REFFLD(LOCAL)
A      K OWID  
A      K ODID  
A      K OCID  
A      K OID
```

## B.15 ORDERSLF: Orders Logical File

```
A* ORDERSLF: Orders Logical file (Split DB version)
A*
A      R ORRCD      PFILE(ORDERSPF01 ORDERSPF02)
A      K OWID  
A      K ODID  
A      K OID
```

## B.16 ORDERSPF: Orders Physical File

```
A*B:H2.ORDERSPF: Orders File REF(AREFFIL)
A*This is the file definition for the orders data base file.
A*All fields that are stored in the orders file are defined
A*here, the actual field definitions are in AREFFIL (the
A*reference file) and are pulled into this file when it is
A*compiled (CRTPF command executed against this source).
A      R ORRCD      TEXT('Orders Master File for TPCC')
A      OWID         REFFLD(WRHSID)
A      ODID         REFFLD(DISTID)
A      OCID         REFFLD(CUSTID)
A      OID          REFFLD(ORDID)
A      OENTTOD       REFFLD(TIMEDATE)
A      COLHGD('Order' 'Date')
A      TEXT('Order Entry Time Date')
A      REFFLD(TIME)
A      OENTTM        REFFLD(TIME)
A      COLHGD('Order' 'Time')
A      TEXT('Order Entry Time')
A      REFFLD(CARRID)
A      OLINES        REFFLD(OLINE)
A      COLHGD('Number of' 'Order Lines')
A      TEXT('Number of Lines in Order')
A      REFFLD(LOCAL)
A      TEXT('Flag to indicate local order')
```

## B.17 ORDLIN: Order Lines Logical File

```
A* ORDLIN: Order Lines file (SPLIT DB)
A*
```

```

A
A      R OLRCD      UNIQUE
A      TEXT('Order Line File for TPCC')
A      PFILE(ORDLINPF01 ORDLINPF02)
A
A      K OLWID
A      K OLOID
A      K OLOID
A      K OLNBR

```

```

A* WRHSLF: Warehouses logical file (Split DB version)
R WRRCB
K WID
PFILE(WRHS01 WRHS02)

```

## B.18 ORDLINLF: Order Lines Logical File

```

A* ORDLINLF: Order Lines logical file (SPLIT DB)
A*
A      R OLRCD      PFILE(ORDLINPF01 ORDLINPF02)
A      K OLWID
A      K OLOID
A      K OLOID

```

## B.23 WRHS: Warehouse Physical File

```

A*B:H2.WRHSPPF: Warehouse File
REF(AREFFIL)
UNIQUE
TEXT('Warehouse Master File - TPCC')
REFFLD(WRHSID)
REFFLD(WRHSID)
REFFLD(LCNAM)
REFFLD(ADDR1)
REFFLD(ADDR2)
REFFLD(CITY)
REFFLD(STATE)
REFFLD(ZIPCD)
REFFLD(TAX)
REFFLD(YTD)
REFFLD(YTD)
COLHGD('YTD' 'Balance')
TEXT('Warehouse YTD Balance')
A
A      R WRRCB
A      WID
A      WNAME
A      WADDR1
A      WADDR2
A      WCITY
A      WSTATE
A      WZIP
A      WTAX
A      WYTD
A
A      K WID

```

## B.19 ORDLINPF: Order Lines Physical File

```

A*B:H2.ORDLINPF: Order Lines File
REF(AREFFIL)
TEXT('Order Line File for TPCC')
REFFLD(ORDID)
REFFLD(DISTID)
TEXT('Customers District')
REFFLD(WRHSID)
COLHGD('Customers' 'Warehouse')
TEXT('Customers Warehouse')
REFFLD(OLINE)
TEXT('Line Number of Order')
REFFLD(WRHSID)
COLHGD('Supply' 'Warehouse')
TEXT('Supply Warehouse ID')
REFFLD(ITEMID)
TEXT('Item Ordered')
REFFLD(QTY)
COLHGD('Qty' 'Ordered')
TEXT('Quantity Ordered')
REFFLD(AMOUNT)
TEXT('Order Line Amount')
REFFLD(TIMEDATE)
COLHGD('Delivery' 'Time Date')
TEXT('Delivery Time Date')
REFFLD(TIME)
COLHGD('Delivery' 'Time')
TEXT('Delivery Time')
REFFLD(DISTINFO)
TEXT('Dist Information')
A
A      R OLRCD
A      OLOID
A      OLOID
A      OLWID
A      OLNBR
A      OLSPWH
A      OLIID
A      OLQTY
A      OLAMNT
A      OLDLVOTD
A      OLDLVT
A*
A*
A      OLDSTI

```

## B.20 STOCK: Stock Logical File

```

A* STOCK: Stock File (SPLIT DB version)
A*
A      UNIQUE
A      PFILE(STOCKPF01 STOCKPF02)
A      R STRCD
A      K STIID
A      K STWID

```

## B.21 STOCKPF: Stock Logical File

```

A*B:H2.STOCKPF: Stock File
A*
A* NOTE: in COMMONTPCC H file is a #define for the record length.
A* Changes in record length need to be updated in the H file also.
A*
A      REF(AREFFIL)
A      R STRCD      TEXT('Stock File for TPCC')
A      STWID        REFFLD(WRHSID)
A      STIID        REFFLD(ITEMID)
A      STQTY        TEXT('Warehouse Nbr of Stock Item')
A      REFFLD(QTY)
A      COLHGD('Qty in' 'Stock')
A      TEXT('Quantity on hand')
A      REFFLD(DISTINFO)
A      STDI01       TEXT('Dist Info 1')
A      REFFLD(DISTINFO)
A      STDI02       TEXT('Dist Info 2')
A      TEXTS('Dist Info 2')
A      STDI03       REFFLD(DISTINFO)
A      TEXT('Dist Info 3')
A      STDI04       REFFLD(DISTINFO)
A      TEXTS('Dist Info 4')
A      STDI05       REFFLD(DISTINFO)
A      TEXT('Dist Info 5')
A      STDI06       REFFLD(DISTINFO)
A      TEXTS('Dist Info 6')
A      STDI07       REFFLD(DISTINFO)
A      TEXT('Dist Info 7')
A      STDI08       REFFLD(DISTINFO)
A      TEXTS('Dist Info 8')
A      STDI09       REFFLD(DISTINFO)
A      TEXT('Dist Info 9')
A      STDI10       REFFLD(DISTINFO)
A      TEXTS('Dist Info 10')
A      STYTD        REFFLD(YTD)
A      STORDRS      TEXT('Qty ordered YTD')
A      REFFLD(QTY)
A      STREMORD      TEXTS('# Lines ordered')
A      REFFLD(QTY)
A      STDATA        TEXT('# times ordered remotely')
A      REFFLD(IDATA)

```

## B.22 WRHSLF: Warehouse Logical File

# Appendix C. Database Build Programs

## Program Flow For Build of Server, Client, and Database

This list shows the order of invocation of the database build programs. Each level of indentation is a call. If a program is indented more than the previous program, it was called by the previous program. If it is at the same margin as the previous program (or as earlier program), it was called by the same program as the previous (or earlier) program. To make determining the level of call easier, the level number will follow the program enclosed in parenthesis.

```
CRTTPCCDB(1)
CRTDBDTA(1)
CHKWHSPACE (1)
FILLPARTS (1)
    CRTBLDSPCE (2)
        TPCCSPACE(3)
    CRTSTOCKLF (2)
    CSTMRVIEW (2)
    CRTHASH (2)
    CRTORDFILE (2)
        MASTORD (3)
            FILLORD(4)
    FILCUSFILE (2)
        MASTCUSSTK (3)
            FILLCUS (4)
    FILSTKFILE (2)
        MASTCUSSTK (3)
            FILLSTOCK (4)
    CRTHISFILE (2)
        FILLHIST (4)
    CRTITEM (2)
        FILLITEM (4)
    CRTWHFILE (2)
        FILLWH (4)
    CRTDIST (2)
        FILLDIST (4)
    CRTDBIX (2)
        CHKIXSPACE (3)
        CRT2VIEW (3)
    CRTINDEX (2)
        CRTHSTRYLF (3)
        CRT1VIEW (3)
COMMON (1)
COMMONTPCC (1)
CRTTPCCJRN (1)
STRTPCCJRN (1)
```

## C.1 CHKIXSPACE.C:

```
/*-----*/
/*
/* FILE: CHKWHSAPCE
/*
/* PURPOSE: Check if there is enough space available.
/*
/*-----*/

#include <stdlib.h> /* Has atoi() prototype. */
#include <osmsg.h>
#include <commonool.h> /* Has get_ASP_stg() prototype. */
#include <decimal.h>

#define NUMWH_PARM 1
#define ASP_PARM 2
#define IX_PARM 3

#define ORDLIN '1'
#define CSTMRFCRT '2'
#define STOCK '3'
#define ORDERS '4'
#define ORDERSLF '5'
#define CSTMR '6'
#define NEWORD '7'

#define ORDLIN_SIZE_MB 771.214
#define STOCK_SIZE_MB 319.083
#define CSTMRFCRT_SIZE_MB 204.235
#define ORDERS_SIZE_MB 98.456
#define CSTMR_SIZE_MB 93.002
#define ORDERSLF_SIZE_MB 87.032
#define NEWORD_SIZE_MB 17.276

#define STOCKPF_SIZE_MB 3070.027
#define CSTMRPF_SIZE_MB 2004.044
#define ORDERSPF_SIZE_MB 93.605

/*-----*/
/*
/* Procedure: main
/*
/*-----*/
int main(int argc, char ** argv)
{
    int aspid;
    int num_whs;

    decimal(19,3) size_needed_GB, ix_size_GB, PF_size_GB;
    decimal(9,2) num_whs_100;
    unsigned long bytes_avail_MB, bytes_cap_MB;
    decimal(19,3) bytes_avail_GB, bytes_cap_GB;

    char msgtxt[130] = "Not enough space for parallel index build. ";
    char index[11] = "\0";
    char temp[7] = "\0";
    /*-----*/
    /* setup addressability to TPCCBUILD user space
    /*-----*/

    aspid = atoi(argv[ASP_PARM]);
    memcpy(temp,argv[NUMWH_PARM],6); /* copy warehouse count field */
    num_whs = atoi(temp); /* Convert count to binary */
    /*-----*/
    /* Get ASP capacity and space available in GB (1,000,000)
    /*-----*/
    if (0 != get_ASP_stg(aspid, &bytes_avail_MB, &bytes_cap_MB))
    {
        QsendMsg(IBM_MSG_FILE, "CPF9897", "CHKIXSPACE program failed!",
        ESCAPE_MSG, CS_CTLBDY, 1);
        return(-1);
    }
    bytes_avail_GB = (decimal(19,3)) bytes_avail_MB/1000 + .0005;
    bytes_cap_GB = (decimal(19,3)) bytes_cap_MB/1000 + .0005;
    /*-----*/
    /* Total size needed for parallel index build
    /* size needed = index size * 5
    /*-----*/
    PF_size_GB = 0.0; /* initialize file size field */
    num_whs_100 = num_whs; /* get warehouse count */
    num_whs_100 = num_whs_100/100; /* Number of 100 warehouse sets */
    switch (argv[IX_PARM][0])
    {
        case ORDLIN: /* ORDLIN index */
            ix_size_GB = (num_whs_100 * ORDLIN_SIZE_MB + .0005)/1000 + .0005;
            PF_size_GB = (num_whs_100 * STOCKPF_SIZE_MB + .0005)/1000 + .0005;
            memcpy(index,"ORDLIN",6);
            break;
        case CSTMRFCRT: /* CSTMRFCRT index */
            ix_size_GB = (num_whs_100 * CSTMRFCRT_SIZE_MB + .0005)/1000 + .0005;
            PF_size_GB = (num_whs_100 * ORDERSPF_SIZE_MB + .0005)/1000 + .0005;
            memcpy(index,"CSTMRFCRT",10);
            break;
        case STOCK: /* STOCK index */
            ix_size_GB = (num_whs_100 * STOCK_SIZE_MB + .0005)/1000 + .0005;
            PF_size_GB = (num_whs_100 * CSTMRPF_SIZE_MB + .0005)/1000 + .0005;
            memcpy(index,"STOCK",5);
            break;
        case ORDERS: /* ORDERS index */
            ix_size_GB = (num_whs_100 * ORDERS_SIZE_MB + .0005)/1000 + .0005;
            memcpy(index,"ORDERS",6);
            break;
        case ORDERSLF: /* ORDERSLF index */
            ix_size_GB = (num_whs_100 * ORDERSLF_SIZE_MB + .0005)/1000 + .0005;
            memcpy(index,"ORDERSLF",8);
            break;
        case CSTMR: /* CSTMR index */
            ix_size_GB = (num_whs_100 * CSTMR_SIZE_MB + .0005)/1000 + .0005;
            memcpy(index,"CSTMR",5);
            break;
        case NEWORD: /* NEWORD index */
            ix_size_GB = (num_whs_100 * NEWORD_SIZE_MB + .0005)/1000 + .0005;
            memcpy(index,"NEWORD",6);
            break;
    }
    size_needed_GB = ix_size_GB * 5;
    /*
    printf("Space required for next file load of %d(19,3)GB\n", PF_size_GB);
    /*
    if (bytes_avail_GB < size_needed_GB)
    {
        sprintf(msgtxt + 43,
        "ASP %d capacity: %d(19,3)GB, available space: %d(19,3)GB, %s size: %d(19,3)GB,\n
        Required space: %d(19,3)GB",
        aspid, bytes_cap_GB, bytes_avail_GB, index, ix_size_GB, size_needed_GB);
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt,ESCAPE_MSG,CS_CTLBDY,1);
        return(-1);
    }
    else
    {
        /*-----*/
        /* Enough storage for loading DB into this ASP.
        /*-----*/
        memcpy(msgtxt, "Space available for parallel index build. ", 43);
        sprintf(msgtxt + 43,
        "ASP %d capacity: %d(19,3)GB, available space: %d(19,3)GB, %s size: %d(19,3)GB,\n

```

```
Required space: %d(19,3)GB",
        aspid, bytes_cap_GB, bytes_avail_GB, index, ix_size_GB, size_needed_GB);
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, INFO_MSG, CS_CTLBDY, 1);
        if (bytes_avail_GB > (size_needed_GB + PF_size_GB))
        {
            size_needed_GB = size_needed_GB + PF_size_GB;
            printf("Space needed %d(19,3)GB for index and file\n", size_needed_GB);
        }
    }
    return(0);
} /* End: main() */
```

## C.2 CHKWHSAPCE.C:

```
/*-----*/
/*
/* FILE: CHKWHSAPCE
/*
/* PURPOSE: Check if there is enough space available.
/*
/*-----*/

#include <stdlib.h> /* Has atoi() prototype. */
#include <osmsg.h> /* Has get_ASP_stg() prototype. */
#include <decimal.h>

#define NUMWH_PARM 1
#define ASP_PARM 2

#define FILE_SIZE_MB 73.453
/* INDEX_SIZE_MB modified for 2way split - was 13.815 */
#define INDEX_SIZE_MB 15.903

/*-----*/
/*
/* Procedure: main
/*
/*-----*/
int main(int argc, char ** argv)
{
    int aspid;
    int num_whs;

    decimal(19,3) size_needed_MB, size_needed_GB, size_needed;
    unsigned long bytes_avail_MB, bytes_cap_MB;
    decimal(19,3) bytes_avail_GB, bytes_cap_GB;

    char msgtxt[130] = "Not enough space to build TPCC Database. ";
    /*-----*/
    aspid = atoi(argv[ASP_PARM]);
    num_whs = atoi(argv[NUMWH_PARM]);
    /*-----*/
    /* Get ASP capacity and space available in GB (1,000,000)
    /*-----*/
    if (0 != get_ASP_stg(aspid, &bytes_avail_MB, &bytes_cap_MB))
    {
        QsendMsg(IBM_MSG_FILE, "CPF9897", "CHKWHSPACE program failed!",
        ESCAPE_MSG, CS_CTLBDY, 1);
        return(-1);
    }
    bytes_avail_GB = (decimal(19,3)) bytes_avail_MB/1000 + .0005;
    bytes_cap_GB = (decimal(19,3)) bytes_cap_MB/1000 + .0005;
    /*-----*/
    /* Total size needed = table size + index size.
    /*-----*/
    size_needed_MB = num_whs * (FILE_SIZE_MB + INDEX_SIZE_MB + .0005);
    size_needed_GB = size_needed_MB/1000 + .0005;
    if (bytes_avail_GB < size_needed_GB)
    {
        sprintf(msgtxt + 41,
        "ASP %d capacity: %d(19,3)GB, available space: %d(19,3)GB, DB size: %d(19,3)GB",
        aspid, bytes_cap_GB, bytes_avail_GB, size_needed_GB);
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt,ESCAPE_MSG,CS_CTLBDY,1);
        return(-1);
    }
    else
    {
        /*-----*/
        /* Enough storage for loading DB into this ASP.
        /*-----*/
        memcpy(msgtxt, "Space available for TPCC database build. ", 41);
        sprintf(msgtxt + 41,
        "ASP %d capacity: %d(19,3)GB, available space: %d(19,3)GB, DB size: %d(19,3)GB",
        aspid, bytes_cap_GB, bytes_avail_GB, size_needed_GB);
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, INFO_MSG, CS_CTLBDY, 1);
    }
    return(0);
} /* End: main() */
```

## C.3 COMMON.C:

```
#define _COMMON_C
#include <Common.h>

/* Establish a listening port */
int Listen(int port) {
    struct sockaddr_in sin;
    int sd, on=1;

    /* Get a socket descriptor */
    if ((sd=socket(AF_INET,SOCK_STREAM,0))<0) {
        perror("socket() failed");
        exit(-1);
    }

    /* Allow socket descriptor to be reusable */
    if (setsockopt(
        sd,
        SOL_SOCKET,
        SO_REUSEADDR,
        (char *)0,
        sizeof(on)
    )<0) {
        perror("setsockopt() failed");
        close(sd);
        exit(-1);
    }

    /* bind to an address */
    memset(&sin, 0x00, sizeof(struct sockaddr_in));
    sin.sin_family = AF_INET;
    sin.sin_port = htons(port);
    sin.sin_addr.s_addr = htonl(INADDR_ANY);

    if (bind(sd, (struct sockaddr *)&sin, sizeof(sin))<0) {

```

```

        perror("bind() failed");
        close(sd);
        exit(-1);
    }

    if (listen(sd, 500) < 0) {
        perror("listen() failed");
        close(sd);
        exit(-1);
    }

    return sd;
}

/* bind to a port and address */
/* returns the socket descriptor */
int Bind(int ipAddr, int port) {
    struct sockaddr_in sin;
    int rc, on=1;

    /* Get a socket descriptor */
    if ((sd=socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket() failed");
        exit(-1);
    }

    /* Allow socket descriptor to be reuseable */
    if (setsockopt(
        sd,
        SOL_SOCKET,
        SO_REUSEADDR,
        (char *) &on,
        sizeof(on)
    ) < 0) {
        perror("setsockopt() failed");
        close(sd);
        exit(-1);
    }

    /* bind to an address */
    memset(&sin, 0x00, sizeof(struct sockaddr_in));
    sin.sin_family = AF_INET;
    sin.sin_port = htons(port);
    sin.sin_addr.s_addr = htonl(ipAddr);

    if (bind(sd, (struct sockaddr *) &sin, sizeof(sin)) < 0) {
        perror("bind() failed");
        close(sd);
        exit(-1);
    }

    return sd;
}

/* Establish connection to specified address */
int Connect(int ipAddr, int port) {
    struct sockaddr_in sin;
    int rc;

    /* Get a socket descriptor */
    if ((sd=socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket() failed");
        exit(-1);
    }

    memset(&sin, 0x00, sizeof(struct sockaddr_in));
    sin.sin_family = AF_INET;
    sin.sin_port = htons(port);
    sin.sin_addr.s_addr = htonl(ipAddr);

    if (connect(sd, (struct sockaddr *) &sin, sizeof(sin)) < 0) {
        perror("connect() failed");
        close(sd);
        exit(-1);
    }

    return sd;
}

#ifdef SEMAPHORE
int CreateSemaphore(Semaphore_t *s, unsigned int count) {
    _Mutex_Create_T attr;
    int rc;

    s->xCount=count;
    memset(&attr, 0, sizeof(attr));
    attr.Mutex_Type=Shared;
    attr.Name_Option=Not_Named;
    attr.Keep_Valid=MUTEX_DESTROY_MUTEX;
    attr.Recursive_Option=MUTEX_NO_RECURSION;

    memset(&s->xTemplate, 0, sizeof(s->xTemplate));

    if (rc=CRTMX(&s->xMutex, &attr)) >= 0
        if ((rc=msgget(IPC_PRIVATE, S_IRUSR|S_IWUSR|S_IROTH|S_IWOTH)) >= 0) {
            s->xSleep=rc;
            rc=0;
        }

    return rc;
}

void WaitSemaphore(Semaphore_t *s) {
    int myCount, msg;

    _LOCKMIX(&s->xMutex, &s->xTemplate);
    myCount==s->xCount;
    _UNLKMITX(&s->xMutex);

    /* Need to wait the count? */
    if (s->xCount < 0)
        msgrcv(s->xSleep, &msg, 0, 0, 0);
}

void SignalSemaphore(Semaphore_t *s) {
    _LOCKMIX(&s->xMutex, &s->xTemplate);

    if (s->xCount < 0) {
        int msg=1;

        msgsnd(s->xSleep, &msg, 0, 0);
    }

    ++s->xCount;
    _UNLKMITX(&s->xMutex);
}

void DestroySemaphore(Semaphore_t *s) {
    _Mutex_Destroy_Opt_T mDo=Destroy;
    _DESRMIX(&s->xMutex, &mDo);
}
#endif

```

## C.4 COMMON.H:

```

#ifdef _COMMON_H
#define _COMMON_H

```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <spawn.h>
#include <qp0z1170.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <mh/mattod>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/msg.h>
#include <netinet/in.h>
#include <errno.h>
#include <unistd.h>
#include <iconv.h>
#include <sys/uio.h>
#include <decimal.h>
#ifdef SEMAPHORE
typedef struct Semaphore {
    _Mutex_T xMutex;
    _Mutex_Lock_T xTemplate;
    int xCount, xSleep;
} Semaphore_t;
#endif
#define BufferSize 3078

extern int Listen(int port);
extern int Connect(int ipAddr, int port);
extern int Bind(int ipAddr, int port);

#ifdef _COMMON_H

```

## C.5 COMMONTOOL.H:

```

#ifdef COMMONTOOL_H
#define COMMONTOOL_H

/*-----*/
/* Header File Name: COMMONTOOL
/*-----*/
/* Description: Miscellaneous functions.
/*-----*/

#include <stdio.h> /* FILE declare. */
#include <qsysinc/mih/matrmd>

/*-----*/
/* Local declares.
/*-----*/
typedef struct ASP_info {
    short ASP_num; /* number of the ASP
    int num_megabytes; /* number of "megabytes" in ASP.
} ASP_info;

#define MEG_BYTES 1000000 /* 1,000,000 bytes of DASD
/* Not 1,048,576 (1024*1024)

/*-----*/
/* NAME: get ASP_info
/* Purpose: Gets the size available in an ASP.
/*-----*/
int get ASP_info(int last ASP, ASP_info *data);
int mat ASP(MATRMD_Template_T *mdata, int aspid);
int get ASP_stg(int aspid, unsigned long *bytes_avail_MB,
    unsigned long *bytes_cap_MB);

/*-----*/
/* NAME: clientGet
/* Purpose: Get the client info.
/*-----*/
int clientGet(char *serv400, char *client, char *clientLib, char *serverLib,
    char *dbLib, char *clientSys);

/* -----*/ End of COMMONTOOL.H <-----*/
#endif /* if COMMONTOOL_H */

```

## C.6 COMMONTPCC.C:

```

/*-----*/
/* File name: COMMONTPCC
/* Purpose: common functions
/*-----*/

/*-----*/
/* Start include files.
/*-----*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <QSYSINC/MIH/MATTOD>
#include <QSYSINC/MIH/MIDTTM>
#include <qp0z1170.h>
#include <string.h>
#include <commontpcc.h>
#include <tpccspace.h>
#include <os4wait.h>
#include <os4msg.h>

/*-----*/
/* Spawn declares
/*-----*/
#include <spawn.h>
#include <sys/stat.h>
#include <qp0wpid.h>
#include <errno.h>
#include <unistd.h> /* Has unlink() */

/*-----*/
/* Function: CreateCust_lastname
/* Build cust last name from predefined syllables.
/* PARS: num => num used to build last name
/* nameStr => address to store last name
/*-----*/
void CreateCust_lastname(int num, char *nameStr)
{
    static char *syble[] = {

```

```

        "BAR", "OUGH", "ABLE", "PRI", "PRES",
        "ESE", "ANTI", "CALLY", "ATION", "EING");

strcpy(nameStr, sylble(num/100));
strcat(nameStr, sylble((num/10)&10));
strcat(nameStr, sylble(num&10));
return;
} /* end of CreateCust_lastname() */

.....
/*
 * Function: CreateStr
 * Create string of random alphanumeric characters.
 * PARSMS: min => minimum length of string
 *          max => maximum length of string
 *          newStr => address to store created string
 * If string is not variable length, then set iMin and iMax to
 * the actual fixed length.
 */
void CreateStr(int min, int max, char *newStr)
{
    int i, newlen;
    char Alpha[63] =
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNQRSTUUVWXYZ0123456789";

    /*-----*/
    newlen = RANDOM(min, max); /* Determine new string length. */
    for (i=0; i < newlen; i++)
    {
        newStr[i] = Alpha[(RANDOM(0,61))];
    }
    newStr[newlen] = '\0';
    return;
} /* end of CreateStr() */

.....
/*
 * Function: CreateStrPad
 * Create string of random alphanumeric characters.
 * Pad remaining length to max with ' '
 * PARSMS: min => minimum length of string
 *          max => maximum length of string
 *          newStr => address to store created string
 */
void CreateStrPad(int min, int max, char *newStr)
{
    int i, newlen;
    char Alpha[63] =
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNQRSTUUVWXYZ0123456789";

    /*-----*/
    newlen = RANDOM(min, max); /* Determine new string length. */
    for (i=0; i < newlen; i++)
    {
        newStr[i] = Alpha[(RANDOM(0,61))];
    }
    newStr[newlen] = '\0';
    memset(newStr+newlen+1, ' ', (max-newlen+1));
    return;
} /* end of CreateStrPad() */

.....
/*
 * Function: CreateStrPad_extra
 * Create string of random alphanumeric characters.
 * Insert extra text in String 10% of the time.
 * Pad remaining length to max with ' '
 * PARSMS: min => minimum length of string
 *          max => maximum length of string
 *          newStr => address to store created string
 */
void CreateStrPad_extra(int min, int max, char *newStr, char *text)
{
    int i, newlen;
    int work/* rndom value for cheking extra */
    int length/* string length */
    int max_end/* the maximum end of the string */
    char Alpha[63] =
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNQRSTUUVWXYZ0123456789";

    /*-----*/
    newlen = RANDOM(min, max); /* Determine new string length. */
    for (i=0; i < newlen; i++)
    {
        newStr[i] = Alpha[(RANDOM(0,61))];
    }
    newStr[newlen] = '\0';
    memset((newStr+newlen+1), ' ', (max-newlen+1));
    work = RANDOM(0, 999);
    if (work < 100) /* check for 10 percent of the time */
    {
        /* Over part of string with special text. */
        length = strlen(text);
        max_end = newlen - length;
        _CPBYTES(&newStr[(RANDOM(0,max_end))], text, strlen(text));
    }
    return;
} /* end of CreateStrPad_extra() */

.....
/*
 * Function: CreateZip
 * Create zip code string.
 * PARSMS: Returns zip code string
 * USE the CREATE_ZIP macro defined in the COMMONTPCC Header file!!
 */
void CreateZip(char *zipstr)
{
    sprintf(zipstr, "%04d1111", (RANDOM(0, 9999)));
    return;
} /* end of CreateZip() */

/* Rerdefine max generated random number. Default is 32,767. */
#undef RAND_MAX
#define RAND_MAX 99999
.....
/*
 * Function: MaxRand
 * Create random number larger than 32767.
 * NOTE: This is the rand() function used by AS/400 with one
 * slight modification. In the rand() function, rand_next is
        /* shifted right by 16 rather than the 14 used here. Shifting
        /* by 16 produces a max value of 65535.
        /*
        /* PARSMS:
        /*
        /*-----*/
        int MaxRand(unsigned long int *rand_next)
        {
            *rand_next = *rand_next * 1103515245 + 12345;
            return((( *rand_next >> 14) & ((unsigned long) (RAND_MAX+1))));
        } /* end of MaxRand() */

.....
/*
 * Function: GetSeed
 * Get a random number to use as a seed value
 * NOTE: use micro seconds of the time as the random number base.
 */
unsigned long int get_seed(void)
{
    struct timeval set_value;
    _MI_Time set_time;

    gettimeofday(&set_value, &set_time, QP0Z_CVTTIME_TO_TIMESTAMP);
    /* Get the time and convert to sec and micro sec */
    return (unsigned long int) set_value.tv_usec;
}

.....
/*
 * Function: spawn_FILL_job
 * Purpose:
 */
int spawn_FILL_job(char *dblib, char *dbfile, int file_id, int inst_num,
                  int str_wh,
                  int num_wh, char *temp_file, char *work_path,
                  char *call_level)
{
    int rc, times;
    int fd count = 0;
    char strwh_string[8];
    char numwh_string[8];
    char instance_string[3];
    char *env_arg[1];
    char *wp_arg[9];
    pid_t pid;
    inheritance_t inherit;

    char msgtxt[60];

    /*-----*/
    sprintf(strwh_string, "%06d", str_wh);
    sprintf(numwh_string, "%06d", num_wh);
    sprintf(instance_string, "%01d", inst_num);

    /* Set parms for the spawn. */
    /*-----*/
    wp_arg[0] = "dummy";
    wp_arg[1] = dblib;
    wp_arg[2] = dbfile;
    wp_arg[3] = strwh_string;
    wp_arg[4] = numwh_string;
    wp_arg[5] = call_level;
    wp_arg[6] = instance_string;
    wp_arg[7] = temp_file;
    wp_arg[8] = NULL;

    env_arg[0] = NULL;
    inherit.flags = 0;
    inherit.pgroup = SPAWN_NEWPGROUP;

    /* Get the number of jobs available for us to use. */
    /*-----*/
    times = 0;
    while ((0 == get_available_jobs(dblib)) && (-1 != rc))
    {
        /*-----*/
        /* Wait and try again... */
        /*-----*/
        tpcc_Wait(5, 0, NULL, NULL); /* Wait 5 seconds. */

        ++times;
        if (2880 == times) /* wait for 4 hours for an available job. */
        {
            sprintf(msgtxt,
                "No available %s jobs. Tired of waiting for one. Quitting!",
                get_DB_filename(file_id));
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
            rc = -1; /* Exit from loop. */
        }
    }

    times = 0;
    while ((0 != add_started_jobs(dblib, file_id, inst_num)) && (-1 != rc))
    {
        /*-----*/
        /* wait and try again... */
        /*-----*/
        tpcc_Wait(3, 0, NULL, NULL); /* Wait 3 seconds. */

        ++times;
        if (100 == times) /* Total of 5 minutes wait time. */
        {
            sprintf(msgtxt, "I can't add to the number of %s jobs I am starting!",
                get_DB_filename(file_id));
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_PGMBDI, 1);
            return(-1);
        }

        pid = spawn(work_path, fd_count, NULL, &inherit,
                  (char **)(&wp_arg), (char **)(&env_arg));

        if (pid < 0)
        {
            sprintf(msgtxt, "%s.02s = %s' not spawned.", work_path, wp_arg[1]);
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
        }

        rc = 0;
        times = 0;
        while ((0 != add_error_jobs(dblib, HISTORY_JOBS, inst_num)) && (-1 != rc))
        {
            /*-----*/
            /* Wait and try again... */
            /*-----*/
            tpcc_Wait(3, 0, NULL, NULL); /* Wait 3 seconds. */

            ++times;
            if (100 == times) /* Total of 5 minutes wait time. */
            {
                sprintf(msgtxt, "I can't set 'error' jobs value for %s files!",
                    get_DB_filename(file_id));
            }
        }
    }
}

```

```

        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
        rc = -1;
    }
    return(-1);
}
return(0);
} /* End of: spawn_FILL_job() */

/*-----*/
/* Function: wait_on_spawned_jobs */
/* Purpose: jobs have been spawned to fill a database file.
   This function will check that the jobs for that
   file_id have completed.
/* Params: dblib = database library name
   file_id = from TPCCSPACE, the int representing the file.
   num_spawned = number of jobs YOU think have been spawned
/*-----*/
int wait_on_spawned_jobs(char *dblib, int file_id, int inst_num,
                        int num_spawned)
{
    int rc, times;
    int str_jobs, done_jobs, succj, errj;
    char msgtxt[100];

    rc = 0;
    if (num_spawned < 1)
    {
        while ((0 == (str_jobs = get_started_jobs(dblib, file_id, inst_num))) &&
              (-1 != rc))
        {
            /*-----*/
            /* Wait and try again... */
            /*-----*/
            tpcc_Wait(3, 0, NULL, NULL); /* Wait 3 seconds. */

            ++times;
            if (60 == times)
            {
                sprintf(msgtxt,
                        "Warning. Number of started %s jobs is 0.",
                        get_DB_filename(file_id));
                QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
                return(-1);
            }
        } /* End: while loop. */

        if (-1 == str_jobs)
        {
            sprintf(msgtxt,
                    "Error getting the number of started %s jobs!",
                    get_DB_filename(file_id));
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
            return(-1);
        }
    } /* end: num_spawned < 1 */

    done_jobs = 0;
    rc = 0;
    while (done_jobs < num_spawned)
    {
        /*-----*/
        /* Wait so we are not constantly checking for completion. */
        /*-----*/
        tpcc_Wait(15, 0, NULL, NULL); /* Wait 15 seconds. */

        times = 0;
        if (-1 == (succj = get_successful_jobs(dblib, file_id, inst_num)))
        {
            sprintf(msgtxt, "I can't get the number of successful %s jobs!",
                    get_DB_filename(file_id));
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
            return(-1);
        }

        times = 0;
        if (-1 == (errj = get_error_jobs(dblib, file_id, inst_num)))
        {
            sprintf(msgtxt, "I can't get the number of %s jobs in error!",
                    get_DB_filename(file_id));
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
            return(-1);
        }

        done_jobs = succj + errj;
    } /* end: (done_jobs < num_spawned) */

    return(0);
} /* End of: wait_on_spawned_jobs() */

```

## C.7 COMMONTPCC.H:

```

#ifndef _COMMONTPCC_H
#define _COMMONTPCC_H
/*-----*/
/* Header file name: COMMONTPCC */
/* Purpose: common functions for TPCC DB Build
/*-----*/
/* DEFINES */
#define BASE_TIME 'b'
#define MAX_DISTRICTS 10
#define CUSTPERDIST 3000
#define ORDERSPERDIST CUSTPERDIST
#define MAXITEMS 100000

#define MX_JOBS 60
#define IO_BLOCK_SIZE 257000
#define NUM_IO_BLOCKS 10
#define MSG_LENGTH 1
#define BLD_MSGQ_NAME_10 "TPCCMSG "

/* The Following is so we don't have to have the file created in
   order to compile the program.
/*-----*/
#define CUSTOMER_RECORD_SIZE 667
#define STOCK_RECORD_SIZE 306

/*-----*/
/* TYPEDEFS */
/*-----*/
typedef struct {
    int bytes_prov;
    int bytes_avail;
    char except_id[7];
    char reserved[1];
}

```

```

        char except_data[50];
    } error_structure;

/*-----*/
/* Prototypes */
/*-----*/
void CreateCust_lastname(int num, char *name);
void CreateStr(int min, int max, char *string);
void CreateStrPad(int min, int max, char *string);
void CreateStrPad_extra(int min, int max, char *string, char *text);
void CreateZip(char *zipcode);
unsigned long int get_seed(void);
int MaxRand(unsigned long int *rand_next);

/*-----*/
/* Function: spawn_FILL_job */
/* Purpose: spawn a fill job for a TPCC database file.
/* Params: dblib = database library name
/*-----*/
int spawn_FILL_job(char *dblib, char *dbfile, int file_id,
                  int inst_num, int str_wh,
                  int num_wh, char *temp_file, char *work_path,
                  char *call_level);

/*-----*/
/* Function: wait_on_spawned_jobs */
/* Purpose: jobs have been spawned to fill a database file.
   This function will check that the jobs for the
   file_id have completed.
/* Params: dblib = database library name
   file_id = from TPCCSPACE, the int representing the file.
   inst_number = the instance number 0 when not a split
   database and 0 or 1 when a split database
   num_spawned = number of jobs YOU think have been spawned
/*-----*/
int wait_on_spawned_jobs(char *dblib, int file_id, int inst_num,
                        int num_spawned);

/*-----*/
/* Macros */
/*-----*/

/* Macro: RANDOM */
/* Purpose: generate a random number.
/* RANDOM is for existing code.
/* RANDOM2 is for code that wants checking to ensure that the two
   values (a & b) do not result in an attempt to divide by
   0.
#define RANDOM(a,b) (rand() % (b-a+1))+a

#define RANDOM2(a, b, r2) \
{ \
    int rr = b-a+1; \
    if (0 == rr) ++rr; \
    r2 = (rand() % rr) + a; \
}

/* Macro: GET_CWD_IDS */
/* Purpose: get the correct customer, warehouse, district ids for
   the Customer file.
/* PARAMS: (input)
   recnum: (int) relative record number
   whs: (int) number of warehouses per customer
/* (output)
   cid: (int) customer id
   wid: (int) warehouse id
   did: (int) district id
/* Example:
   10 districts, 3 warehouses
1. 1 1 1 ((1-1)/10)+1=1, ((1-1)/10)*3+1=1, ((1-1)/(10*3))+1=1
2. 1 1 2 ((2-1)/10)+1=2, ((2-1)/10)*3+1=1, ((2-1)/(10*3))+1=1
3. 1 1 3
4. 1 1 4
5. 1 1 5
6. 1 1 6
7. 1 1 7
8. 1 1 8
9. 1 1 9
10. 1 1 10((10-1)/10)+1=10, ((10-1)/10)*3+1=1, ((10-1)/(10*3))+1=1
11. 1 2 1((11-1)/10)+1= 1, ((11-1)/10)*3+1=2, ((11-1)/(10*3))+1=1
12. 1 2 2
13. 1 2 3
14. 1 2 4
15. 1 2 5
16. 1 2 6
17. 1 2 7
18. 1 2 8
19. 1 2 9
20. 1 2 10((20-1)/10)+1=10, ((20-1)/10)*3+1=2, ((20-1)/(10*3))+1=1
21. 1 3 1((21-1)/10)+1= 1, ((21-1)/10)*3+1=3, ((21-1)/(10*3))+1=1
22. 1 3 2
23. 1 3 3
24. 1 3 4
25. 1 3 5
26. 1 3 6
27. 1 3 7
28. 1 3 8
29. 1 3 9
30. 1 3 10((30-1)/10)+1=10, ((30-1)/10)*3+1=3, ((30-1)/(10*3))+1=1
31. 2 1 1((31-1)/10)+1= 1, ((31-1)/10)*3+1=1, ((31-1)/(10*3))+1=2
32. 2 1 2
33. 2 1 3
34. 2 1 4
35. 2 1 5
36. 2 1 6
37. 2 1 7
38. 2 1 8
39. 2 1 9
40. 2 1 10((40-1)/10)+1=10, ((40-1)/10)*3+1=1, ((40-1)/(10*3))+1=2
41. 2 2 1((41-1)/10)+1= 1, ((41-1)/10)*3+1=2, ((41-1)/(10*3))+1=2
*/

#define GET_CWD_IDS(recnum, whs, cid, wid, did) \
{ \
    did = ((recnum - 1) % MAX_DISTRICTS) + 1; \
    wid = (((recnum - 1) / MAX_DISTRICTS) % whs) + 1; \
    cid = ((recnum - 1) / (MAX_DISTRICTS * whs)) + 1; \
}

/* Macro: CREATE_ZIP */

```

```

/*
/* Purpose: Create zip code string.
/*
/* Param: char string the Zip code will be placed into.
/*
-----*/
#define CREATE_ZIP(zipstring) \
{ \
    sprintf(zipstring, "%04d1111", (RANDOM(0, 9999))); \
}

/*-----*/
/* Macro: NEW RAND
/*
/* Purpose: return integer value.
/*
/* Aint is constant chosen according to size of range [x..y]
/*
/* PARS: Aint = integer value
/* Xmin = minimum value in range
/* Ymax = maximum value in range
/*-----*/
#define NEW RAND(Aint, Xmin, Ymax) \
(((RANDOM(0,Aint) | RANDOM(Xmin,Ymax)) + 7) & (Ymax-Xmin+1)) + Xmin)
/* Refer to Page 20 TPCBenchmarkKMC - Standard Specification, Revision */

/*-----*/
/* Macro: IO_BLOCKS_PER_JOB
/*
/* Purpose: calculate the number of IO blocks per stock or customer
/* job
/* The maximum number of IO blocks is the #define value.
/* The purpose of this macro is to determine if we will
/* use all the IO blocks allowed by the #define value.
/*
/* PARS: nbpj = number IO blocks per job (OUTPUT)
/* num_rec_per_IO_block =
/* final_rec = last record number
/* njobs = number of jobs
/*-----*/
#define IO_BLOCKS_PER_JOB(nbpj, num_rec_per_IO_block, final_rec, njobs) \
nbpj = (final_rec / num_rec_per_IO_block) / njobs; \
if (0 != (final_rec % num_rec_per_IO_block) / njobs) \
{ \
    ++nbpj; \
} \
if (NUM_IO_BLOCKS < nbpj) \
{ \
    nbpj = NUM_IO_BLOCKS; \
}

/*-----*/ End of COMMONTPCC.H <-----*/
#endif

```

## C.8 CRT1VIEW.CL:

```

/*-----*/
/* NAME: CRT1VIEW
/*
/* PURPOSE: create one logical view from input file name.
/*
/* Note that Split DB file names have source members with
/* the number 2 in their name.
/* Create the district or warehouse logical files
/*-----*/
PGM PARM(&FILE &DBLIB &NUMPARTS)

DCL &FILE TYPE(*CHAR) LEN(6)
DCL &DBLIB TYPE(*CHAR) LEN(10)
DCL &NUMPARTS TYPE(*CHAR) LEN(2)

DCL &FILESRCLEF TYPE(*CHAR) LEN(10)

/*-----*/
/* Note if number of parts is 1 these files are not created */
MONMSG (CPF7302 CPF0001) EXEC(GOTO ERREXIT)
IF COND(&NUMPARTS *EQ '01') THEN(GOTO ERREXIT)
IF COND(&FILE *EQ 'WRHS ') THEN(
    CHGVAR &FILESRCLEF VALUE(WRHSF0000))
ELSE CMD(+
DO)
    IF COND(&FILE *EQ 'DSTRCT') THEN(+
    CHGVAR &FILESRCLEF VALUE(DSTRCTLF00))
    ELSE CMD(+
    DO)
        IF COND(&FILE *EQ 'HSTRY ') THEN(+
        CHGVAR &FILESRCLEF VALUE(HSTRYLF000))
        ELSE CMD(GOTO ERREXIT)
    ENDDO
ENDDO
CHGVAR %SST(&FILESRCLEF 9 2) VALUE(&NUMPARTS)

/* Create PARTIAL key view. */
CHKOBJ OBJ(&DBLIB/&FILE) OBJTYPE(*FILE)
MONMSG CPF9801 EXEC(+
DO)
    IF COND(&FILE *EQ 'HSTRY ') THEN(CRTLF +
    FILE(&DBLIB/&FILE) SRCMBR(&FILESRCLEF) +
    MAINT(*REBLD) WAITRCD(*NOMAX) LVLCHK(*NO))
    ELSE CMD(+
    CRTLF FILE(&DBLIB/&FILE) SRCMBR(&FILESRCLEF) +
    WAITRCD(2) LVLCHK(*NO))
ENDDO

SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('CRT1VIEW completed successfully!') MSGTYPE(*COMP)
GOTO ENDPGM

ERREXIT: SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('CRT1VIEW failed!') MSGTYPE(*ESCAPE)
ENDPGM: ENDPGM

```

## C.9 CRT2VIEW.CL:

```

/*-----*/
/* NAME: CRT2VIEW
/*
/* PURPOSE: create two logical views: one based on the file name
/* and the other ending with LF.
/*
/* Note that Split DB file names have source members with
/* has the partition number as the last 2 characters.
/* Characters 9 and 10 the name is filed with zeros.
/* All names have 6 characters. ORDERS, NEWORD, and
/* ORDLIN.
/*-----*/
PGM PARM(&FILE &DBLIB &NUMPARTS)

DCL &FILE TYPE(*CHAR) LEN(6)

```

```

DCL &DBLIB TYPE(*CHAR) LEN(10)
DCL &NUMPARTS TYPE(*CHAR) LEN(2)

DCL &FILELEF TYPE(*CHAR) LEN(10)
DCL &FILESRC TYPE(*CHAR) LEN(10)
DCL &FILESRCLEF TYPE(*CHAR) LEN(10)

/*-----*/
MONMSG (CPF7302 CPF0001) EXEC(GOTO ERREXIT)

/* setup the first part of the names */
IF COND(%SST(&FILE 6 1) = ' ') THEN(+
GOTO ERREXIT)
CHGVAR VAR(&FILESRC) VALUE(&FILE)
CHGVAR VAR(&FILESRCLEF) VALUE(&FILE)
CHGVAR VAR(%SST(&FILESRCLEF 7 2)) VALUE('LF')
CHGVAR VAR(&FILELEF) VALUE(&FILESRCLEF)
IF COND(&NUMPARTS *GT '01') THEN(+
DO)
    CHGVAR VAR(%SST(&FILESRC 7 2)) VALUE('00')
    CHGVAR VAR(%SST(&FILESRC 9 2)) VALUE(&NUMPARTS)
    CHGVAR VAR(%SST(&FILESRCLEF 9 2)) VALUE(&NUMPARTS)
ENDDO

/* Create the UNIQUE view over the physical file. */
TAKEACTION:
CHKOBJ OBJ(&DBLIB/&FILE) OBJTYPE(*FILE)
MONMSG CPF9801 EXEC(CRTLF FILE(&DBLIB/&FILE) SRCMBR(&FILESRC) +
WAITRCD(*NOMAX) LVLCHK(*NO))

/* Create PARTIAL key view. */
CHKOBJ OBJ(&DBLIB/&FILELEF) OBJTYPE(*FILE)
MONMSG CPF9801 EXEC(CRTLF FILE(&DBLIB/&FILELEF) SRCMBR(&FILESRCLEF) +
WAITRCD(*NOMAX) LVLCHK(*NO))

SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('CRT2VIEW completed successfully!') MSGTYPE(*COMP)
GOTO ENDPGM

ERREXIT: SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('CRT2VIEW failed!') MSGTYPE(*ESCAPE)
ENDPGM: ENDPGM

```

## C.10 CRTBLDSPCE.C:

```

/*-----*/
/* File name: CRTBLDSPCE
/*
/* Purpose: Build space.
/*-----*/

/*-----*/
/* Start include files.
/*-----*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <qstrvus.h>
#include <qstrctus.h>
#include <qspstrus.h>
#include <qsgen.h>
#include <time.h>
#include <cmh/cpybytes.h>
#include <qcnfig.h>
#include <usrspc.h>
#include <qcmdexc.h>
#include <qsmmsg.h>
#include <commontpcc.h>
#include <tpccspacei.h>

#define DBLIB_PARM 1
#define STRWH_PARM 2
#define NUMWH_PARM 3
#define SPLIT_PARM 4

#define CNFIG_SPACE "TPCCBLDCPU"

/*-----*/
/* Globals
/*-----*/
int number_processors;

/*-----*/
void init_job_data(job_data *jobs_ptr)
{
    jobs_ptr->max_allowed = number_processors + 1;
    jobs_ptr->planned = 0;
    jobs_ptr->running = 0;
    jobs_ptr->completed = 0;
    jobs_ptr->error = 0;
}

/*-----*/
int create_user_space(control_space **control_ptr, char *space_name)
{
    Qus_Generic_Header_0100_t *space;
    char initial_char; /* user space parameter creation */
    char replace[10];
    char text_desc[50];
    error_structure err;

    err.bytes_prov = 0;
    _CPYBYTES(replace, "YES", 10);
    cpyblap(text_desc, 50, "Job info for DB build.", 22, ' ');
    initial_char = 0; /* Initialize to binary zero, user not active. */

    QUSCRTUS(space_name, "PF", sizeof(control_space), &initial_char,
    "ALL", text_desc, replace, &err);

    err.bytes_prov = 0; /* set number of bytes and error reporting mode */

    QUSPTRUS(space_name, &space, &err); /* get the pointer */

    *control_ptr = (control_space *)space->User_Area; /* get the address. */

    /*-----*/
    /* Set max and available jobs as the total of processors for each */
    /* of the DB jobs plus 1 for each for luck. This should be */
    /* more than we ever need. */
    /*-----*/
    (*control_ptr)->available_jobs = (number_processors * 4) + 4;
    (*control_ptr)->max_jobs = (number_processors * 4) + 4;
    (*control_ptr)->actual_cpu_cnt = number_processors;

    (*control_ptr)->working[0].start_whs = 1; /* set starting to 1 */
    (*control_ptr)->working[0].numbeF_whs = 0; /* set for zero warehouses */

    init_job_data(&(*control_ptr)->working[0].orders);
    init_job_data(&(*control_ptr)->working[0].stock);
    init_job_data(&(*control_ptr)->working[0].customer);
    init_job_data(&(*control_ptr)->working[0].history);

    (*control_ptr)->working[1].start_whs = 1; /* set starting to 1 */
    (*control_ptr)->working[1].numbeF_whs = 0; /* set for zero warehouses */

    init_job_data(&(*control_ptr)->working[1].orders);
    init_job_data(&(*control_ptr)->working[1].stock);
}

```



```

init_job_data(&(*control_ptr->working[1].customer);
init_job_data(&(*control_ptr->working[1].history );

return(0);
) /* end of create_user_space */

/*-----*/
int main(int argc, char **argv)
{
    int start_who;
    int numb_who;
    int split_low; /* number of warehouse in the low section */
    unsigned int lib_len;
    char *slash_pos;

    char cmd[100];
    char dblib[11]; /* full program string */
    char pgm_name[11]; /* full program string */
    char pgm_lib[11]; /* the program library name. */
    char dataspace[21]; /* data space name */
    char space_name[21]; /* space name used to transfer data. */
    char output_block[7]; /* output block for writing. */
    char msgq[20];
    char msgxt[132];
    lib_len = slash_pos = argv[0];
    error_code err; /* error for check pointer */
    control_space *control_ptr; /* pointer to user space. */
    Qcnfig *data_ptr; /* configuration data pointer */
    Qus_Generic_Header_0100_t *space; /* user space header */

    ExcData_t volatile exd = {0, "HANDLE "};

    /*-----*/
    /* Get the library name that we are running in now. */
    /*-----*/
    slash_pos = strchr(argv[0], '/');
    lib_len = slash_pos - argv[0];
    _CPYBYTES(pgm_lib, argv[0], lib_len);
    pgm_lib[lib_len] = '\0';

    ++slash_pos; /* Skip the '/' */
    _CPYBYTES(pgm_name, slash_pos, 10);
    pgm_name[triml(pgm_name, ' ')] = '\0';

    /* TPCC message queue to send our build messages to. */
    _CPYBYTES(msgq, "TPCCMSG", 10);
    _CPYBYTES(&msgq[10], argv[DBLIB_PARAM], 10);

    dblib[10] = '\0';
    _CPYBYTES(dblib, argv[DBLIB_PARAM], 10);
    dblib[triml(dblib, ' ')] = '\0';

    /*-----*/
    /* Get config values. */
    /*-----*/
    sprintf(cmd, "TPCCTOOLS/QCNFIG RUNLIB(%s) SPNAME(%s) RESET(*YES)",
            dblib, CNFIG_SPACE);

    #pragma exception_handler(QexceptHdlr, exd, 0, _C2_MH_ESCAPE)
    QCMDEXC(cmd, strlen(cmd));
    #pragma disable_handler

    if (0 != exd.error)
    {
        QsendMsg(IBM_MSG_FILE, "CPF9897", "QCNFIG program call failed.",
                ESCAPE_MSG, CS_CTRLEDY, 1);
    }

    _CPYBYTES(dataspace, CNFIG_SPACE, 10);
    _CPYBYTES(&dataspace[10], argv[DBLIB_PARAM], 10);
    err.bytes_prov = 0;

    QUSPTRUS(dataspace, &space, &err); /* get the pointer. */

    data_ptr = (Qcnfig*)(space->User_Area); /* get the address. */
    if (data_ptr->prcmittsk[0] == '0')
    {
        number_processors = data_ptr->CPUcnt;
        sprintf(msgxt, "CRTBLDSPCE: number of processors = %d. HMT is OFF.",
                number_processors);
    }
    else {
        number_processors = 2 * data_ptr->CPUcnt;
        sprintf(msgxt,
                "CRTBLDSPCE: number of processors = %d. HMT ON, so processors = %d.",
                data_ptr->CPUcnt, number_processors);
    }
    Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgxt, INFO_MSG, msgq, 1, NULL);

    /*-----*/
    /* Delete the CNFIG user space. */
    /*-----*/
    sprintf(cmd, "DLTUSRSPC %s/%s", dblib, CNFIG_SPACE);

    #pragma exception_handler(QexceptHdlr, exd, 0, _C2_MH_ESCAPE)
    QCMDEXC(cmd, strlen(cmd));
    #pragma disable_handler

    _CPYBYTES(space_name, USERSPACE, 10); /* get first part of the name */
    _CPYBYTES(&space_name[10], argv[DBLIB_PARAM], 10);
    space_name[20] = '\0';

    create_user_space(&control_ptr, space_name);

    start_who = atoi(argv[STRWH_PARAM]);
    numb_who = atoi(argv[NUMWH_PARAM]);

    if (argv[SPLIT_PARAM][0] == 'Y')
    {
        split_low = numb_who / 2;

        control_ptr->max_instance = 2; /* set number of instance to 2 */
        control_ptr->working[0].start_who = start_who;
        control_ptr->working[0].number_who = split_low;
        control_ptr->working[1].start_who = start_who + split_low;
        control_ptr->working[1].number_who = split_low + (numb_who % 2);
    }
    else {
        control_ptr->max_instance = 1;
        control_ptr->working[0].start_who = start_who;
        control_ptr->working[0].number_who = numb_who;
    }
    return;
} /* end: main() */

```

## C.11 CRTDBDTA.CL:

```

/*-----*/
/*
/* Program: CRTDBDTA
/* Purpose: create TPCC data areas if they do not exist already.
/*
/*-----*/
PGM PARM(&DBLIB)

DCL &DBLIB TYPE(*CHAR) LEN(10)
DCL &PGMLIB TYPE(*CHAR) LEN(10)
/*-----*/

```

```

MONMSG MSGID(CPF0000 MCH0000) EXEC(GOTO ERREXIT)

SNDFPMMSG MSGID(CPF9897) +
MSGDTA('CRTDBDTA: creating data areas...') +
MSGF(QCPFMSG TOPGMQ(*EXT) MSGTYPE(*STATUS))

RTV0BJD OBJ(CRTDBDTA) OBJTYPE(*PGM) RTNLIB(&PGMLIB)

CRTDTAARA &PGMLIB/TPCCDBLIB TYPE(*CHAR) LEN(10) VALUE(&DBLIB) +
TEXT('Data Base Library')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCDBVRS TYPE(*CHAR) LEN(10) VALUE('32') +
TEXT('Data Base Version')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCJRNLIB TYPE(*CHAR) LEN(10) +
VALUE(JRNRCVLIB) TEXT('Journal receiver library')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCWRHS TYPE(*DEC) LEN(6) VALUE(0) +
TEXT('Number of warehouses')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCAPLLIB TYPE(*CHAR) LEN(10) VALUE(' ') +
TEXT('Application Library')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCRNLIB TYPE(*CHAR) LEN(10) VALUE(' ') +
TEXT('Run performance data library')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCUSERS TYPE(*DEC) LEN(7) VALUE(0) +
TEXT('Number of real users in TPCC')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCSAVLIB TYPE(*CHAR) LEN(10) +
VALUE(TPCCSFMDJ) TEXT('Save library name')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

/*-----*/
CRTDTAARA &DBLIB/TPCCSLJOB TYPE(*DEC) LEN(5) +
VALUE(25) TEXT('Number of users per Stock Level Job')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCDLVJOB TYPE(*DEC) LEN(5) +
VALUE(40) TEXT('Number of users per Delivery Job')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCNGRCV TYPE(*CHAR) LEN(10) +
VALUE(*SYSNM) TEXT('Manage journal receiver')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCNJJOB TYPE(*DEC) LEN(5) +
VALUE(50) TEXT('Number of users per New Order Job')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCPAYJOB TYPE(*DEC) LEN(5) +
VALUE(25) TEXT('Number of users per Payment Job')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCRCVLSZ TYPE(*CHAR) LEN(10) +
VALUE(*NONE) TEXT('Journal receiver size option')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCRCVTRH TYPE(*DEC) LEN(10) +
VALUE(150000) TEXT('Journal receiver threshold')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCOJOB TYPE(*DEC) LEN(5) +
VALUE(8) TEXT('Number of users per Order Status Job')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

/*-----*/
CRTDTAARA &DBLIB/HWRPEXDEFN TYPE(*CHAR) LEN(10) +
VALUE(HDWTST05) TEXT('HWR PEX definition name')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/HWRPTIME TYPE(*DEC) LEN(5) VALUE(1800) +
TEXT('HWR data time')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TSKPEXDEFN TYPE(*CHAR) LEN(10) +
VALUE(TASKSWT) TEXT('Task Switch PEX definition name')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TSKPTIME TYPE(*DEC) LEN(5) +
VALUE(60) TEXT('Task switch data time')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPRPEXDEFN TYPE(*CHAR) LEN(10) +
VALUE(TPROFRC) TEXT('TPROF PEX definition name')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPRPTIME TYPE(*DEC) LEN(5) VALUE(1800) +
TEXT('TPROF data time')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPSTPEXDEFN TYPE(*CHAR) LEN(10) VALUE(TPST) +
TEXT('TPST PEX definition name')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPSTTIME TYPE(*DEC) LEN(5) VALUE(1800) +
TEXT('TPST data time')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/SAVEMEDIA TYPE(*CHAR) LEN(10) VALUE(*SAVF) +
TEXT('Save media')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/USR1PEXDEFN TYPE(*CHAR) LEN(10) +
TEXT('User PEX definition name 1')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/USR1TIME TYPE(*DEC) LEN(5) VALUE(0) +
TEXT('User data time 1')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/USR2PEXDEFN TYPE(*CHAR) LEN(10) +
TEXT('User PEX definition name 2')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/USR2TIME TYPE(*DEC) LEN(5) VALUE(0) +
TEXT('User data time 2')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/USR3PEXDEFN TYPE(*CHAR) LEN(10) +
TEXT('User PEX definition name 3')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/USR3TIME TYPE(*DEC) LEN(5) VALUE(0) +
TEXT('User data time 3')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/USR4PEXDEFN TYPE(*CHAR) LEN(10) +
TEXT('User PEX definition name 4')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/USR4TIME TYPE(*DEC) LEN(5) VALUE(0) +
TEXT('User data time 4')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/USR5PEXDEFN TYPE(*CHAR) LEN(10) +

```

```

TEXT('User PEX definition name 5')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/USRSTIME TYPE(*DEC) LEN(5) VALUE(0) +
TEXT('User data time 5')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/PFRMONTIME TYPE(*DEC) LEN(5) VALUE(3840) +
TEXT('Performance Monitor data time')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCRPTDSC TYPE(*CHAR) LEN(40) VALUE(TPCC) +
TEXT('Collection/Report description')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/CNFIG TYPE(*CHAR) LEN(5) VALUE(*MON) +
TEXT('Configuration type')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/CNTVALUE TYPE(*CHAR) LEN(10) VALUE(*MINUTES) +
TEXT('Counter value')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/LOGOPT TYPE(*CHAR) LEN(5) VALUE(*ON) +
TEXT('Data logging')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/MEASURE TYPE(*CHAR) LEN(10) VALUE(*FOREVER) +
TEXT('Measurement time to run')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/RSPLG TYPE(*CHAR) LEN(10) VALUE(*RSPLG) +
TEXT('Response log')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/RUN TYPE(*CHAR) LEN(4) VALUE(*YES) +
TEXT('Run a measurement')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCSRVR TYPE(*CHAR) LEN(10) VALUE(*UNKNOWN) +
TEXT('Transaction system name')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TPCCACTLIB TYPE(*CHAR) LEN(10) VALUE(*TPCCDATA) +
TEXT('Transaction library name')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/TRNMIX TYPE(*CHAR) LEN(4) VALUE(*MIX) +
TEXT('Transaction mixture')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/WARMUP TYPE(*CHAR) LEN(7) VALUE(*NONE) +
TEXT('Warmup time')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/WHRNG1 TYPE(*DEC) LEN(10 0) VALUE(1) +
TEXT('Starting warehouse number')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

CRTDTAARA &DBLIB/WHRNG2 TYPE(*DEC) LEN(10 0) VALUE(0) +
TEXT('Ending warehouse number')
MONMSG CPF1023 EXEC(RCVMSG RMV(*YES))

SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('CRTDBDTA completed successfully!') MSGTYPE(*COMP)
GOTO ENDPGM

ERREXIT: SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('CRTDBDTA failed!') MSGTYPE(*ESCAPE)
ENDPGM: ENDPGM

```

## C.12 CRTDBIX.CL:

```

PGM PARM(&IXD &DBLIB &ASPD &TEMPSTG &NUMPARTS &WHS)
DCL VAR(&IXD) TYPE(*DEC) LEN(1)
DCL VAR(&IX) TYPE(*CHAR) LEN(1)
DCL VAR(&DBLIB) TYPE(*CHAR) LEN(10)
DCL VAR(&WMSD) TYPE(*DEC) LEN(6)
DCL VAR(&WHS) TYPE(*CHAR) LEN(6)
DCL VAR(&ASPD) TYPE(*DEC) LEN(2)
DCL VAR(&NUMPARTS) TYPE(*DEC) LEN(2)
DCL VAR(&NUMPARTSC) TYPE(*CHAR) LEN(2)
DCL VAR(&ASP) TYPE(*CHAR) LEN(2)
DCL VAR(&TEMPSTG) TYPE(*CHAR) LEN(1)
DCL VAR(&PGLIB) TYPE(*CHAR) LEN(10)
DCL VAR(&BLDJOBQ) TYPE(*CHAR) LEN(10) VALUE(TPCCBLDJOBQ)
DCL VAR(&MSGQ) TYPE(*CHAR) LEN(10) VALUE(INDEXMSGQ)
CHGVAR VAR(&IX) VALUE(&IXD)
CHGVAR VAR(&WHS) VALUE(&WMSD)
CHGVAR VAR(&ASP) VALUE(&ASPD)
CHGVAR VAR(&NUMPARTSC) VALUE(&NUMPARTS)
RTVDBJID OBJ(CRTDBIX) OBJTYPE(*PGM) RTNLIB(&PGLIB)
/* Decide whether to use ASP1 or ASP2 for index builds */
/* ASP2 requires special modules loaded in SLIC code */
/* ***** */
IF COND(&TEMPSTG *EQ 'D') THEN(+
DO)
CALL PGM(CHKIXSPACE) PARM(&WHS &ASP &IX)
MONMSG MSGID(CPF9897) EXEC(GOTO CMDLBL(NONPAR))
ENDDO
ELSE CMD(+
DO)
/* Check the temp storage in the system ASP */
CALL PGM(CHKIXSPACE) PARM(&WHS '01' &IX)
MONMSG MSGID(CPF9897) EXEC(GOTO CMDLBL(NONPAR))
ENDDO
/* ***** */
/* Parallel index build */
/* ***** */
SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('Starting parallel index build') +
MSGTYPE(*INFO)
CHGQRYA DEGREE(*NRX)
CHGQRYA DEGREE(*NRXTASKS 9) /*
WRKSYSSTS OUTPUT(*PRINT) RESET(*YES)
IF COND(&IXD = 1) THEN(+
DO)
CALL PGM(&PGLIB/CRT2VIEW) PARM('ORDLIN' +
&DBLIB &NUMPARTSC)
GOTO PARENDACT
ENDDO
IF COND(&IXD = 2) THEN(+
DO)
CALL PGM(&PGLIB/CSTMVIEW) PARM(&DBLIB +
&NUMPARTSC)
GOTO PARENDACT
ENDDO
IF COND(&IXD = 3) THEN(+
DO)
CALL PGM(&PGLIB/CRTSTOCKLF) PARM(&DBLIB +
&NUMPARTSC)
GOTO PARENDACT
ENDDO
IF COND(&IXD = 4) THEN(+
DO)
CALL PGM(&PGLIB/CRT2VIEW) PARM('ORDERS' +
&DBLIB &NUMPARTSC)

```

```

GOTO PARENDACT
ENDDO
IF COND(&IXD = 5) THEN(+
DO)
CALL PGM(&PGLIB/CRT2VIEW) PARM('NEWORD' +
&DBLIB &NUMPARTSC)
GOTO PARENDACT
ENDDO
PARENDACT: WRKSYSSTS OUTPUT(*PRINT) RESET(*NO)
CHGQRYA DEGREE(*NONE)
GOTO ENDPGM
/* ***** */
/* Non-Parallel index build */
/* ASP does not have enough DASD space (5K index size) */
/* ***** */
NONPAR: SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('NOT ENOUGH SPACE FOR A PARALLEL +
BUILD') MSGTYPE(*INFO)
CHGQRYA DEGREE(*NONE)
WRKSYSSTS OUTPUT(*PRINT) RESET(*YES)
IF COND(&IXD = 1) THEN(+
DO)
CALL &PGLIB/CRT2VIEW PARM('ORDLIN' &DBLIB +
&NUMPARTSC)
GOTO NORMEXIT
ENDDO
IF COND(&IXD = 2) THEN(+
DO)
SBMJOB CMD(CALL &PGLIB/CSTMVIEW PARM(&DBLIB &NUMPARTSC) +
JOB(CSTMVIEW) JOBQ(&PGLIB/TPCCBLDJOBQ) MSGQ(&DBLIB/&MSGQ))
GOTO NORMEXIT
ENDDO
IF COND(&IXD = 3) THEN(+
DO)
CALL PGM(&PGLIB/CRTSTOCKLF) PARM(&DBLIB &NUMPARTSC)
GOTO NORMEXIT
ENDDO
IF COND(&IXD = 4) THEN(+
DO)
CALL &PGLIB/CRT2VIEW PARM('ORDERS' &DBLIB +
&NUMPARTSC)
GOTO NORMEXIT
ENDDO
IF COND(&IXD = 5) THEN(+
DO)
CALL &PGLIB/CRT2VIEW PARM('NEWORD' &DBLIB +
&NUMPARTSC)
GOTO NORMEXIT
ENDDO
NORMEXIT:
ENDPGM: ENDPGM

```

## C.13 CRTDIST.CL:

```

/* ***** */
/* Program: CRTDIST */
/* ***** */
/* Purpose: Create and fill the DSTRCT file. */
/* ***** */
/* Parms: Database library name */
/* File name */
/* Starting warehouse id (null terminated) */
/* Number of warehouses (null terminated) */
/* ***** */
PGM PARM(&DBLIB &DISTFILE &STRWH &NUMWH &RECOMPILE)
DCL &DBLIB TYPE(*CHAR) LEN(10)
DCL &DISTFILE TYPE(*CHAR) LEN(10)
DCL &STRWH TYPE(*DEC) LEN(6)
DCL &NUMWH TYPE(*DEC) LEN(6)
DCL &RECOMPILE TYPE(*CHAR) LEN(1)
DCL &STRWHC TYPE(*CHAR) LEN(6)
DCL &NUMWHC TYPE(*CHAR) LEN(6)
DCL &STRWHNULL TYPE(*CHAR) LEN(7)
DCL &NUMWHNULL TYPE(*CHAR) LEN(7)
DCL &NULLCH TYPE(*CHAR) LEN(1) VALUE(X'00')
DCL &DISTDEF TYPE(*CHAR) LEN(10) VALUE(DSTRCT)
DCL &NUM1 TYPE(*DEC) LEN(6 0)
DCL &NUM2 TYPE(*DEC) LEN(6 0)
DCL &PGLIB TYPE(*CHAR) LEN(10)
DCL &MSGQ TYPE(*CHAR) LEN(10) VALUE(TPCCMSGQ)
DCL &MDATA TYPE(*CHAR) LEN(49) +
VALUE('CRTDIST program is filling the db file...')
MONMSG MSGID(CPF0000 MCH0000) EXEC(GOTO ERREXIT)
/* Other pgms will be called from library we are running in. */
RTVDBJID OBJ(CRTDIST) OBJTYPE(*PGM) RTNLIB(&PGLIB)
/* ***** */
/* Create the DISTRICT and WRHS files and programs.*/
/* ***** */
CHGVAR VAR(&SST(&MDATA 32 7)) VALUE(&DISTFILE)
SNDPGMMSG MSGID(CPF9897) MSGDTA(&MDATA) +
MSGF(QSYS/QCPFMMSG) TOPMQ(*EXT) MSGTYPE(*STATUS)
SNDPGMMSG MSGID(CPF9897) MSGDTA(&MDATA) +
MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) TOMSQ(&DBLIB/&MSGQ)
DLTF &DBLIB/&DISTFILE
MONMSG MSGID(CPF2105) EXEC(RCVMSG RMV(*YES))
CRTPF FILE(&DBLIB/&DISTFILE) SRCMBR(DSTRCT) +
SIZE(*NOMAX) WAITRCD(2) LVLRCD(*NO)
IF (&RECOMPILE *EQ 'Y') THEN(DO)
OVRDBF FILE(&DISTDEF) TOFILE(&DBLIB/&DISTFILE)
CRTCMOD &PGLIB/FILLDIST
CRTPGM PGM(&PGLIB/FILLDIST) +
MODULE(&PGLIB/FILLDIST &PGLIB/COMMONTPCC &PGLIB/TPCCSPACE) +
ENTMOD(&PGLIB/FILLDIST) BNDSRVPGM(TPCCTOOLS/TPCCTOOLS)
DLTOVR FILE(&DISTDEF)
ENDDO
CHGVAR &STRWHC VALUE(&STRWH)
CHGVAR &STRWHNULL VALUE(&STRWHC *CAT &NULLCH)
CHGVAR &NUMWHC VALUE(&NUMWH)
CHGVAR &NUMWHNULL VALUE(&NUMWHC *CAT &NULLCH)
CALL &PGLIB/FILLDIST PARM(&DBLIB &DISTFILE &STRWHNULL &NUMWHNULL)
SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*COMP) +
MSGDTA('CRTDIST program completed successfully!') +
TOMSQ(&DBLIB/TPCCMSGQ)
GOTO ENDPGM
ERREXIT: SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('CRTDIST failed!') MSGTYPE(*ESCAPE)
SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +

```

```
-----MSGDTA('CRTDIST failed!') TOMSGQ(&DBLIB/&MSGQ)
ENDPGM: ENDPGM
```

## C.14 CRTHASH.C:

```

/*-----*/
/*
/* Program: CRTHASH
/*
/* Params: argv[1] = Database library name
/*          argv[2] = Number of warehouses
/*          argv[3] = Physical file name
/*          argv[4] = Logical file name
/*          argv[5] = Create userspace flag '0' = no '1' = yes
/*
/*-----*/
#include <stdlib.h>
#include <stdio.h>

/* The following is from the TPCCTOOLS project.
#include <os4msg.h>

/*-----*/
/* internal defines
/*-----*/

#define DBLIB_PARM 1
#define WH_PARM 2
#define PFFILE_PARM 3
#define LFFILE_PARM 4
#define CREATE_PARM 5
#define TOTAL_PARM 5

typedef _Packed struct key_struct {
    char name[10];
    long value;
} key_ranges_structure[5];

long qdbrtha(char flag, char *hash_name, char *pf, char *pf_lib,
            char *lf, char *lf_lib,
            key_ranges_structure key_ranges, char *expression);

/*-----*/
void main(int argc, char *argv[])
{
    int wh;
    char whc[7];
    char pgm_lib[11];
    char create;
    char db_lib[11];
    char lf_arg[11];
    char pf_arg[11];
    char msgtxt[35];
    key_ranges_structure key_ranges;

    /*-----*/
    if (argc < TOTAL_PARM)
    {
        printf(msgtxt, "Expected %s parms, received %d parm",
              (TOTAL_PARM - 1), (argc-1));
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLDY, 1);
        exit;
    }

    if (0 == strcmp(argv[PFFILE_PARM], "ITEM", 4))
    {
        key_ranges[0].value = 100000;
    }
    else {
        memcpy(whc, argv[WH_PARM], 6);
        whc[6] = '\0';
        wh = atoi(whc);
        if (0 == wh)
        {
            printf(msgtxt, "Error: the warehouse parameter has a length of 0.");
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLDY, 1);
            exit;
        }
        if (0 == strcmp(argv[PFFILE_PARM], "STOCK", 5))
        {
            key_ranges[0].value = 100000;
            key_ranges[1].value = wh;
        }
        else {
            key_ranges[0].value = 3000; /* Customer ID */
            key_ranges[1].value = 10; /* District ID */
            key_ranges[2].value = wh;
        }
    }

    pgm_lib[0] = '\0'; /* Not used. */
    _CPYBYTES(db_lib, argv[DBLIB_PARM], 10);
    db_lib[10] = '\0';
    _CPYBYTES(pf_arg, argv[PFFILE_PARM], 10);
    pf_arg[10] = '\0';
    _CPYBYTES(lf_arg, argv[LFFILE_PARM], 10);
    lf_arg[10] = '\0';

    if ('N' == *(argv[CREATE_PARM]))
        create = '0';
    else
        create = '1';

    if (-1 == qdbrtha(create, lf_arg, pf_arg, db_lib, lf_arg,
                    "DPT", ""))
    {
        QsendMsg(IBM_MSG_FILE, "CPF9897", "Error received from QDBCRTHA program.",
              ESCAPE_MSG, CS_CTLDY, 1);
    }
    QsendMsg(IBM_MSG_FILE, "CPF9897", "CRTHASH completed successfully.",
          COMP_MSG, CS_CTLDY, 1);
    return;
} /* End of: main() */

```

## C.15 CRTHISFILE.CL:

```

/*-----*/
/*
/* NAME: CRTHISFILE
/*
/* PURPOSE: create and fill a History database file.
/*
/*-----*/
PGM PARM(&DBLIB &DBFILE &STRWH &NUMWHS &RECOMPILE)

DCL &DBLIB TYPE(*CHAR) LEN(10)
DCL &DBFILE TYPE(*CHAR) LEN(10)
DCL &DBFILEDEF TYPE(*CHAR) LEN(10) VALUE(HSTRY)
DCL &STRWH TYPE(*DEC) LEN(6 0)
DCL &NUMWHS TYPE(*DEC) LEN(6 0)
DCL &RECOMPILE TYPE(*CHAR) LEN(1)

DCL &NUM1 TYPE(*DEC) LEN(6 0)
DCL &NUM2 TYPE(*DEC) LEN(6 0)

```

```

DCL &STRWHC TYPE(*CHAR) LEN(6)
DCL &NUMWHC TYPE(*CHAR) LEN(6)
DCL &STRWHNULL TYPE(*CHAR) LEN(7)
DCL &NUMWHNULL TYPE(*CHAR) LEN(7)
DCL &NULLLCH TYPE(*CHAR) LEN(1) VALUE(X'00')

DCL &PGMLIB TYPE(*CHAR) LEN(10)
DCL &REUSEREC TYPE(*CHAR) LEN(4) VALUE(*YES)
DCL &ORDSIZ TYPE(*DEC) LEN(9 0)

DCL &MDATA1 TYPE(*CHAR) LEN(43) +
    VALUE('CRTHISFILE: filling the db file...')
DCL &MDATA TYPE(*CHAR) LEN(58) +
    VALUE('CRTHISFILE: calling FILLHIST pgm to fill db file...')

/*-----*/
MONMSG MSGID(CPF0000 CEE9901 MCH0000) EXEC(GOTO ERREXIT)

RTVBJ OBJ(CRTHISFILE) OBJTYPE(*PGM) RTNLIB(&PGMLIB)
MONMSG CPF9811 EXEC(DO)
SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*DIAG) +
    MSGDTA('Add the library with CRTHISFILE pgm to lib list!')
GOTO ERREXIT
ENDDO

CRTMSGQ MSGQ(&DBLIB/TPCCMSG)
MONMSG CPF2112 EXEC(RCVMSG RMV(*YES))

CHGVAR VAR(&SST(&MDATA1 25 7)) VALUE(&DBFILE)
SNDPGMMSG MSGID(CPF9897) MSGDTA(&MDATA1) +
    MSGF(QSYS/QCPFMMSG) TOPMQ(*EXT) MSGTYPE(*STATUS)

IF COND(&NUMWHS < 3000) THEN(CHGVAR &REUSEREC VALUE(*NO))

CALL TPCCTOOLS/SETTOOLLIB
CALL &PGMLIB/SETBLDLIBL /* So we can do a CRTPF. */

ADDLIB &DBLIB
MONMSG MSGID(CPF2103) EXEC(RCVMSG RMV(*YES))

DLTF &DBLIB/&DBFILE
MONMSG MSGID(CPF2105) EXEC(RCVMSG RMV(*YES))

CHGVAR &ORDSIZ VALUE(&NUMWHS * 36000)
CRTPF FILE(&DBLIB/&DBFILE) SRCMBR(&DBFILEDEF) +
    SIZE(&ORDSIZ 30000 30000) ALLOCATE(*YES) +
    WAITRCD(*NOMAX) SHARE(*YES) DLTCT(50) +
    REUSEDLT(&REUSEREC) LVLCHK(*NO)

/*-----*/
/* Now that the HSTRY file has been created...
/* Create programs to fill the HSTRY file.
/*-----*/
IF (&RECOMPILE *EQ 'Y') THEN(DO)

OVRDBF FILE(&DBFILEDEF) TOFILE(&DBLIB/&DBFILE) OVRSCOPE(*JOB)
CRTCMOD MODULE(&PGMLIB/FILLHIST) SRCFILE(QCSRC)

CRTPGM PGM(&PGMLIB/FILLHIST) +
    MODULE(&PGMLIB/FILLHIST) &PGMLIB/COMMONTPCC &PGMLIB/TPCCSPACE) +
    ENTMOD(&PGMLIB/FILLHIST) Sndsrvpgm(TPCCTOOLS/TPCCTOOLS)

DLTOVR FILE(&DBFILEDEF) LVL(*JOB)
ENDDO

CHGVAR &STRWHC VALUE(&STRWH)
CHGVAR &STRWHNULL VALUE(&STRWHC *CAT &NULLLCH)
CHGVAR &NUMWHC VALUE(&NUMWHS)
CHGVAR &NUMWHNULL VALUE(&NUMWHC *CAT &NULLLCH)

/*-----*/
/* Call pgm to fill HSTRY file.
/*-----*/
CHGVAR VAR(&SST(&MDATA 42 6)) VALUE(&DBFILE)
SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +
    MSGDTA(&MDATA) TOMSGQ(&DBLIB/TPCCMSG)

/* WRKSYSSTS OUTPUT(*PRINT) RESET(*YES) */

CALL &PGMLIB/FILLHIST +
    PARM(&DBLIB &DBFILE &STRWHNULL &NUMWHNULL '000' '000')

/* WRKSYSSTS OUTPUT(*PRINT) RESET(*NO) */

SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*COMP) +
    MSGDTA('CRTHISFILE program completed successfully!') +
    TOMSGQ(&DBLIB/TPCCMSG)
GOTO ENDPGM

ERREXIT: SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*DIAG) +
    MSGDTA('CRTHISFILE failed!') TOMSGQ(&DBLIB/TPCCMSG)
SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
    MSGDTA('CRTHISFILE failed!') MSGTYPE(*ESCAPE)

ENDPGM: ENDPGM

```

## C.16 CRTHSTRYLF.CL:

```

/*-----*/
/*
/* NAME: CRTHSTRYLF
/*
/* PURPOSE: create logical view from history file.
/*          has delayed maintenance option
/*          Note that Split DB file names have source members with
/*          the partition number in their name.
/*
/*-----*/
PGM PARM(&DBLIB &NUMPARTS)

DCL &FILE TYPE(*CHAR) LEN(10) VALUE(HSTRY)
DCL &LIB TYPE(*CHAR) LEN(10)
DCL &NUMPARTS TYPE(*CHAR) LEN(2)

DCL &FILESRCFL TYPE(*CHAR) LEN(10) VALUE(HSTRYLF000)

/*-----*/
MONMSG (CPF7302 CPF0001) EXEC(GOTO ERREXIT)
IF COND(&NUMPARTS *EQ '01') THEN(GOTO ERREXIT)
CHGVAR &SST(&FILESRCFL 9 2) VALUE(&NUMPARTS)
/* Create PARTIAL key view. */
CHKOBJ OBJ(&DBLIB/&FILE) OBJTYPE(*FILE) +
MONMSG MSGID(CPF9801) EXEC(CRTLF FILE(&DBLIB/&FILE) +
    SRCMBR(&FILESRCFL) MAIN(*REBLD) +
    WAITRCD(*NOMAX) LVLCHK(*NO))

SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
    MSGDTA('CRTRVIEW completed successfully!') MSGTYPE(*COMP)
GOTO ENDPGM

ERREXIT: SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
    MSGDTA('CRTRVIEW failed!') MSGTYPE(*ESCAPE)
ENDPGM: ENDPGM

```

## C.17 CRTINDEX.CL:

```

/*-----*/

```

```

/*
/* Program: CRTINDEX
/*
/* Create an index with the given name
/* Call check index build status then build the index in line
/* of submit the index build
/* Params: Where to put pgm objects
/* Database library name
/* Database library asp
/* Number of warehouses
/* Number of partitions
/* Build files and index mixed
-----*/
PGM PARM(&INDEX &DBLIB &DBASPCN &NUMMHS &NUMPARTSC &PARIND)

DCL &INDEX TYPE(*CHAR) LEN(6)
DCL &DBLIB TYPE(*CHAR) LEN(10)
DCL &DBASPCN TYPE(*CHAR) LEN(3) /* DB ASP number w/null */
DCL &NUMMHS TYPE(*DEC) LEN(6)
DCL &PARIND TYPE(*CHAR) LEN(1)

DCL &PGMLIB TYPE(*CHAR) LEN(10)
DCL &MSGQ TYPE(*CHAR) LEN(10) VALUE(INDEXMSGQ)
DCL &NUMPARTSC TYPE(*CHAR) LEN(2)

/* Other pgms will be called from library we are running in. */
RTVOBJD OBJ(CRTINDEX) OBJTYPE(*PGM) RTNLIB(&PGMLIB)

/*-----*/
IF COND(&INDEX *EQ 'CSTMR ') THEN (+
DO)
SBMJOB CMD(CALL &PGMLIB/CSTMRVIEW PARM(&DBLIB &NUMPARTSC) +
JOB(CSTMRINDEX) JOBQ(&PGMLIB/TPCBLDJOBQ) MSGQ(&DBLIB/&MSGQ))
GOTO NORMEXIT
ENDDO
IF COND(&INDEX *EQ 'DSTRCT') THEN (+
DO)
SBMJOB CMD(CALL &PGMLIB/CRTVIEW PARM('DSTRCT' &DBLIB +
&NUMPARTSC) +
JOB(DSTRCTINDEX) JOBQ(&PGMLIB/TPCBLDJOBQ) MSGQ(&DBLIB/&MSGQ))
GOTO NORMEXIT
ENDDO
IF COND(&INDEX *EQ 'HSTRY ') THEN (+
DO)
SBMJOB CMD(CALL PGM(&PGMLIB/CRTHSTRYLF) PARM( +
&DBLIB &NUMPARTSC) JOB(HSTRYINDEX) +
JOBQ(&PGMLIB/TPCBLDJOBQ) MSGQ(&DBLIB/&MSGQ))
GOTO NORMEXIT
ENDDO
IF COND(&INDEX *EQ 'NEWORD') THEN (+
DO)
SBMJOB CMD(CALL &PGMLIB/CRT2VIEW PARM('NEWORD' &DBLIB +
&NUMPARTSC) +
JOB(NEWORDINDEX) JOBQ(&PGMLIB/TPCBLDJOBQ) MSGQ(&DBLIB/&MSGQ))
GOTO NORMEXIT
ENDDO
IF COND(&INDEX *EQ 'ORDERS') THEN (+
DO)
SBMJOB CMD(CALL &PGMLIB/CRT2VIEW PARM('ORDERS' &DBLIB +
&NUMPARTSC) +
JOB(ORDERSINDEX) JOBQ(&PGMLIB/TPCBLDJOBQ) MSGQ(&DBLIB/&MSGQ))
GOTO NORMEXIT
ENDDO
IF COND(&INDEX *EQ 'ORDLIN') THEN (+
DO)
SBMJOB CMD(CALL &PGMLIB/CRT2VIEW PARM('ORDLIN' &DBLIB +
&NUMPARTSC) +
JOB(ORDLININDEX) JOBQ(&PGMLIB/TPCBLDJOBQ) MSGQ(&DBLIB/&MSGQ))
GOTO NORMEXIT
ENDDO
IF COND(&INDEX *EQ 'STOCK ') THEN (+
DO)
SBMJOB CMD(CALL &PGMLIB/CRTSTOCKLF PARM(&DBLIB +
&NUMPARTSC) +
JOB(STOCKINDEX) JOBQ(&PGMLIB/TPCBLDJOBQ) MSGQ(&DBLIB/&MSGQ))
GOTO NORMEXIT
ENDDO
IF COND(&INDEX *EQ 'WRHS ') THEN (+
DO)
SBMJOB CMD(CALL PGM(&PGMLIB/CRTVIEW) PARM('WRHS ' +
&DBLIB &NUMPARTSC) JOB(WRHSINDEX) +
JOBQ(&PGMLIB/TPCBLDJOBQ) MSGQ(&DBLIB/&MSGQ))
GOTO NORMEXIT
ENDDO
NORMEXIT:
ENDPGM

```

## C.18 CRTITEM.CL:

```

/*-----*/
/*
/* Program: CRTITEM
/*
/* Purpose: Create the ITEM PF and fill it.
/*
/* Params: Database library name
/*
-----*/
PGM PARM(&DBLIB &NUMPARTS)

DCL &DBLIB TYPE(*CHAR) LEN(10)
DCL &NUMPARTS TYPE(*DEC) LEN(2 0)
DCL &ITEMFILE TYPE(*CHAR) LEN(10)
DCL &NEWFILE TYPE(*CHAR) LEN(10)
DCL &PGMLIB TYPE(*CHAR) LEN(10)
DCL &CURRPART TYPE(*CHAR) LEN(2)
DCL &CURRPART TYPE(*DEC) LEN(2 0)
DCL &MSGQ TYPE(*CHAR) LEN(10) VALUE(TPCCMSG)

MONMSG MSGID(CPF0000 MCH0000) EXEC(GOTO ERREXIT)

/* Other pgms will be called from library we are running in. */
RTVOBJD OBJ(CRTITEM) OBJTYPE(*PGM) RTNLIB(&PGMLIB)

/*-----*/
/* Create the ITEM file and program.
/*-----*/
DLTF &DBLIB/ITEMLF
MONMSG MSGID(CPF2105) EXEC(RCVMSG RMV(*YES))

IF COND(&NUMPARTS *EQ 1) THEN(DO)
CHGVAR &ITEMFILE VALUE(ITEM)
ENDDO
ELSE DO
CHGVAR &ITEMFILE VALUE(ITEM01)
ENDDO

DLTF &DBLIB/&ITEMFILE
MONMSG MSGID(CPF2105) EXEC(RCVMSG RMV(*YES))
CRTPF FILE(&DBLIB/&ITEMFILE) SRCMBR(ITEM) +
SIZE(*NOMAX) WAITRCD(*NOMAX) LVLCHK(*NO)

OVRDBF FILE(ITEM) TOFILE(&DBLIB/&ITEMFILE)

CRTCMD &PGMLIB/FILLITEM
CRTPGM &PGMLIB/FILLITEM +
MODULE(&PGMLIB/FILLITEM &PGMLIB/TPCCSPACE &PGMLIB/COMMONTPCC) +
ENDSRVPGM(TPCCTOOLS/TPCCTOOLS)

```

```

DLTOVR FILE(ITEM)

CALL &PGMLIB/FILLITEM PARM(&DBLIB &ITEMFILE)
IF COND(&NUMPARTS *GT 1) THEN(DO)
CHGVAR &CURRPART VALUE(2)
CHGVAR &NEWFILE VALUE(&ITEMFILE)
CRTCLPGM PGM(&DBLIB/TRIGGERFIN)
CRTCLPGM PGM(&DBLIB/TRIGGERSET)
CRTCLPGM PGM(&DBLIB/TRIGGERIT)
ITEMLOOP:
CHGVAR &CURRPART VALUE(&CURRPART)
CHGVAR &SSST(&NEWFILE 5 2) VALUE(&CURRPART)
CRTDPOBJ OBJ(&ITEMFILE) FROMLIB(&DBLIB) +
OBJTYPE(*FILE) NEWOBJ(&NEWFILE) DATA(*YES)
RNMN FILE(&NEWFILE) MBR(&ITEMFILE) NEWMBR(&NEWFILE)
SNDPGMSG &NEWFILE
ADDPFTRG FILE(&NEWFILE) TRGTIME(*BEFORE) +
TRSEVENT(*INSERT) PGM(&DBLIB/TRIGGERSET)
ADDPFTRG FILE(&NEWFILE) TRGTIME(*BEFORE) +
TRSEVENT(*DELETE) PGM(&DBLIB/TRIGGERSET)
ADDPFTRG FILE(&NEWFILE) TRGTIME(*BEFORE) +
TRSEVENT(*UPDATE) PGM(&DBLIB/TRIGGERSET)
ADDPFTRG FILE(&NEWFILE) TRGTIME(*AFTER) +
TRSEVENT(*INSERT) PGM(&DBLIB/TRIGGERFIN)
ADDPFTRG FILE(&NEWFILE) TRGTIME(*AFTER) +
TRSEVENT(*DELETE) PGM(&DBLIB/TRIGGERFIN)
ADDPFTRG FILE(&NEWFILE) TRGTIME(*AFTER) +
TRSEVENT(*UPDATE) PGM(&DBLIB/TRIGGERFIN)
CHGVAR &CURRPART VALUE(&CURRPART + 1)
IF COND(&CURRPART *LE &NUMPARTS) THEN(GOTO +
CMDLBL(ITEMLOOP))
ENDDO

GOTO ENDPGM

ERREXIT: SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('CRTITEM failed!') MSGTYPE(*ESCAPE)
SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +
MSGDTA('CRTITEM failed!') TOMSGQ(&DBLIB/&MSGQ)
ENDPGM: ENDPGM

```

## C.19 CRTORDFILE.CL:

```

/*-----*/
/*
/* NAME: CRTORDFILE
/*
/* PURPOSE: create and fill the following Database files:
/* - ORDERSPF
/* - ORDLINPF
/* - NEWORDPF
/*-----*/
PGM PARM(&DBLIB &ORDERSNAME &ORDLINNAME &NEWORDNAME &STRWH &NUMWH +
&RECCOMPILE)

DCL &DBLIB TYPE(*CHAR) LEN(10)
DCL &ORDERSNAME TYPE(*CHAR) LEN(10)
DCL &ORDLINNAME TYPE(*CHAR) LEN(10)
DCL &NEWORDNAME TYPE(*CHAR) LEN(10)
DCL &STRWH TYPE(*DEC) LEN(6 0)
DCL &NUMWH TYPE(*DEC) LEN(6 0)
DCL &RECCOMPILE TYPE(*CHAR) LEN(1)

DCL &STRWHC TYPE(*CHAR) LEN(6)
DCL &STRWHNULL TYPE(*CHAR) LEN(7)
DCL &NUMWHC TYPE(*CHAR) LEN(6)
DCL &NUMWHNULL TYPE(*CHAR) LEN(7)

DCL &NUM1 TYPE(*DEC) LEN(6 0)
DCL &NUM2 TYPE(*DEC) LEN(6 0)

DCL &ORDERSNAMN TYPE(*CHAR) LEN(11)
DCL &ORDLINNAMN TYPE(*CHAR) LEN(11)
DCL &NEWORDNAMN TYPE(*CHAR) LEN(11)
DCL &ORDERSFILE TYPE(*CHAR) LEN(10) VALUE(ORDERSPF)
DCL &ORDLINFILE TYPE(*CHAR) LEN(10) VALUE(ORDLINPF)
DCL &NEWORDFILE TYPE(*CHAR) LEN(10) VALUE(NEWORDPF)

DCL &PGMLIB TYPE(*CHAR) LEN(10)
DCL &NULLCH TYPE(*CHAR) LEN(1) VALUE('X'00')

DCL &REUSEREC TYPE(*CHAR) LEN(4) VALUE(*YES)
DCL &ORDSIZ TYPE(*DEC) LEN(9 0)
DCL &ORDLINSIZ TYPE(*DEC) LEN(10 0)
DCL &ORDLINSIZW TYPE(*DEC) LEN(10 0)
DCL &NEWORDSIZ TYPE(*DEC) LEN(9 0)

DCL &MDATA TYPE(*CHAR) LEN(43) +
VALUE('CRTORDFILE: filling all the ORDER files...')

/*-----*/
MONMSG MSGID(CPF0000 MCH0000 CZM0000) EXEC(GOTO ERREXIT)

RTVOBJD OBJ(CRTORDFILE) OBJTYPE(*PGM) RTNLIB(&PGMLIB)
MONMSG CPF9811 EXEC(DO)
SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*DIAG) +
MSGDTA('Add the library with CRTORDFILE pgm to lib list!')
GOTO ERREXIT
ENDDO

SNDPGMSG MSGID(CPF9897) MSGF(QCPFMMSG) MSGTYPE(*STATUS) TOPGMQ(*EXT) +
MSGDTA('CRTORDFILE: starting...')

ADDLIB &DBLIB
MONMSG CPF2103 EXEC(RCVMSG RMV(*YES))

CRTMSGQ MSGQ(&DBLIB/TPCCMSG)
MONMSG CPF2112 EXEC(RCVMSG RMV(*YES))

IF COND(&NUMWH < 3000) THEN(CHGVAR &REUSEREC VALUE(*NO))

DLTF &DBLIB/&ORDERSNAME
MONMSG CPF2105 EXEC(RCVMSG RMV(*YES))

DLTF &DBLIB/&ORDLINNAME
MONMSG CPF2105 EXEC(RCVMSG RMV(*YES))

DLTF &DBLIB/&NEWORDNAME
MONMSG CPF2105 EXEC(RCVMSG RMV(*YES))

CHGVAR &ORDSIZ VALUE(&NUMWH * 36000)
CHGVAR &ORDLINSIZ VALUE(&NUMWH * 360000)
CHGVAR &ORDLINSIZW VALUE(&ORDLINSIZ)
/* Save the required size for extra filling. */
IF COND(&ORDLINSIZ > 2147483646) THEN(DO)
IF COND(&ORDLINSIZ < 3024000000) THEN(DO)
CHGVAR &ORDLINSIZW VALUE(2147479552)
GOTO CRTFILES
ENDDO
IF COND(&ORDLINSIZ < 3168000000) THEN(DO)
CHGVAR &ORDLINSIZW VALUE(2147479552)
GOTO CRTFILES
ENDDO
IF COND(&ORDLINSIZ < 3312000000) THEN(DO)
CHGVAR &ORDLINSIZW VALUE(2147479554)
GOTO CRTFILES
ENDDO
IF COND(&ORDLINSIZ < 3456000000) THEN(DO)
CHGVAR &ORDLINSIZW VALUE(2147479555)
GOTO CRTFILES
ENDDO

```

```

ENDDO
IF COND(&ORDLINSIZ < 3600000000) THEN(DO)
  CHGVAR &ORDLINSIZ VALUE(2147479556)
  GOTO CRTFILES
ENDDO
IF COND(&ORDLINSIZ < 3744000000) THEN(DO)
  CHGVAR &ORDLINSIZ VALUE(2147479557)
  GOTO CRTFILES
ENDDO
IF COND(&ORDLINSIZ < 3898000000) THEN(DO)
  CHGVAR &ORDLINSIZ VALUE(2147479558)
  GOTO CRTFILES
ENDDO
IF COND(&ORDLINSIZ < 4032000000) THEN(DO)
  CHGVAR &ORDLINSIZ VALUE(2147479559)
  GOTO CRTFILES
ENDDO
IF COND(&ORDLINSIZ < 4176000000) THEN(DO)
  CHGVAR &ORDLINSIZ VALUE(2147479560)
  GOTO CRTFILES
ENDDO
IF COND(&ORDLINSIZ < 4294800000) THEN(DO)
  CHGVAR &ORDLINSIZ VALUE(2147479561)
  GOTO CRTFILES
ENDDO
IF COND(&ORDLINSIZ < 4032000000) THEN(DO)
  CHGVAR &ORDLINSIZ VALUE(2147479562)
  GOTO CRTFILES
ENDDO
ENDDO
CRTFILES:
CHGVAR &NEWORDSIZ VALUE(&NUMWH * 11000)

CRTPF FILE(&DBLIB/&ORDERSNAME) SRCMBR(&ORDERSFILE) +
  SIZE(&ORDSIZ 30000 1000) ALLOCATE(*YES) +
  WAITRCD(*NOMAX) SHARE(*YES) DLTPTC(50) +
  REUSEDLT(&REUSEREC) LVLCHK(*NO)

CRTPF FILE(&DBLIB/&ORDLINNAME) SRCMBR(&ORDLINFILE) +
  SIZE(&ORDLINSIZ 00000 00000) +
  ALLOCATE(*YES) WAITRCD(*NOMAX) +
  SHARE(*YES) DLTPTC(50) +
  REUSEDLT(&REUSEREC) LVLCHK(*NO)

CRTPF FILE(&DBLIB/&NEWORDNAME) SRCMBR(&NEWORDFILE) +
  SIZE(&NEWORDSIZ 30000 30000) +
  ALLOCATE(*YES) WAITRCD(*NOMAX) +
  SHARE(*YES) DLTPTC(50) +
  REUSEDLT(&REUSEREC) LVLCHK(*NO)

IF (&RECOMPILE *EQ 'Y') THEN(DO)

OVRDDB &ORDERSFILE TOFILE(&DBLIB/&ORDERSNAME) OVRSCOPE(*JOB)
MONMSG CPF2103 EXEC(RCVMSG RMV(*YES))
OVRDDB &ORDLINFILE TOFILE(&DBLIB/&ORDLINNAME) OVRSCOPE(*JOB)
MONMSG CPF2103 EXEC(RCVMSG RMV(*YES))
OVRDDB &NEWORDFILE TOFILE(&DBLIB/&NEWORDNAME) OVRSCOPE(*JOB)
MONMSG CPF2103 EXEC(RCVMSG RMV(*YES))

/*-----*/
/* Now that the files have been created... */
/* Create programs to fill the orders file. */
/*-----*/
CALL TPCCTOOLS/SETTOOLIBL /* So we get correct header file. */

CRTCMOD &PGLIB/FILLORD
CRTCMOD &PGLIB/MASTORD

CRTPGM &PGLIB/FILLORD +
  MODULE(&PGLIB/FILLORD) &PGLIB/COMMONTPCC &PGLIB/TPCCSPACE +
  ENTMOD(&PGLIB/FILLORD) ENDSRVPGM(TPCCTOOLS/TPCCTOOLS)
CRTPGM &PGLIB/MASTORD +
  MODULE(&PGLIB/MASTORD) &PGLIB/COMMONTPCC &PGLIB/TPCCSPACE +
  ENTMOD(&PGLIB/MASTORD) ENDSRVPGM(TPCCTOOLS/TPCCTOOLS)

DLTOVR FILE(&NEWORDFILE) LVL(*JOB)
MONMSG CPF9841
DLTOVR FILE(&ORDLINFILE) LVL(*JOB)
MONMSG CPF9841
DLTOVR FILE(&ORDERSFILE) LVL(*JOB)
MONMSG CPF9841
ENDDO

CHGVAR &STRWHC VALUE(&STRWH)
CHGVAR &STRWHNULL VALUE(&STRWHC *CAT &NULLCH)
CHGVAR &NUMWHC VALUE(&NUMWH)
CHGVAR &NUMWHNULL VALUE(&NUMWHC *CAT &NULLCH)
CHGVAR &ORDERSNAMN VALUE(&ORDERSNAME *CAT &NULLCH)
CHGVAR &ORDLINNAMN VALUE(&ORDLINNAME *CAT &NULLCH)
CHGVAR &NEWORDNAMN VALUE(&NEWORDNAME *CAT &NULLCH)

/*-----*/
/* Call pgm to fill ORDERS, ORDLINP and NEWORDERS file. */
/*-----*/
SNDPGMSG MSGID(CPF9897) MSGDTA(&MDATA) +
  MSGF(QSYS/QCPFMMSG) TOPGMQ(*EXT) MSGTYPE(*STATUS)
SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +
  MSGDTA(&MDATA) TOMSGQ(&DBLIB/TPCCMSG)

/* WRKSYSSTS OUTPUT(*PRINT) RESET(*YES) */

CALL &PGLIB/MASTORD PARM(&DBLIB/&ORDERSNAMN &ORDLINNAMN +
  &NEWORDNAMN &STRWHNULL &NUMWHNULL)
/* WRKSYSSTS OUTPUT(*PRINT) RESET(*NO) */

SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*COMP) +
  MSGDTA('CRTTORDFILE program completed successfully!') +
  TOMSGQ(&DBLIB/TPCCMSG)
GOTO ENDPGM

ERREXIT: SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*ESCAPE) +
  MSGDTA('CRTTORDFILE failed!')
SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +
  MSGDTA('CRTTORDFILE failed!') TOMSGQ(&DBLIB/TPCCMSG)
ENDPGM: ENDPGM

DCL &JRNLIB TYPE(*CHAR) LEN(10)
DCL &JRNASP TYPE(*DEC) LEN(2 0)
DCL &MNGRCV TYPE(*CHAR) LEN(10)
DCL &RCVSIZEOPT TYPE(*CHAR) LEN(10)
DCL &THRESHOLD TYPE(*DEC) LEN(10)
DCL &WHS TYPE(*DEC) LEN(6 0)
DCL &NUMPARTS TYPE(*DEC) LEN(2 0)
DCL &NUMPARTSC TYPE(*CHAR) LEN(2)
DCL &SAVEDB TYPE(*CHAR) LEN(1)
DCL &MEDIA *CHAR LEN(5)
DCL &SAVLIB *CHAR LEN(10)
DCL &SAVASP *DEC LEN(2 0)
DCL &DEVNAME *CHAR LEN(10)

DCL &WHSC TYPE(*CHAR) LEN(6)
DCL &WHSCN TYPE(*CHAR) LEN(7)
DCL &DBASPC TYPE(*CHAR) LEN(2)
DCL &DBASPCN TYPE(*CHAR) LEN(3)

DCL &NULL TYPE(*CHAR) LEN(1) VALUE('X'00')
DCL &DATE TYPE(*CHAR) LEN(6)
DCL &PGLIB TYPE(*CHAR) LEN(10)
DCL &MSGQ TYPE(*CHAR) LEN(10) VALUE(TPCCMSG)
DCL &BLDSBS TYPE(*CHAR) LEN(10) VALUE(TPCCBLDSBS)
DCL &RCVSIZEMOD TYPE(*CHAR) LEN(32)

DCL &MSGDATA TYPE(*CHAR) LEN(65) +
  VALUE('CRTTPCCDB: starting warehouse id = 1 and the total equals .')

/*-----*/
/* Start of executable code */
/*-----*/
MONMSG MSGID(CPF2556 CPF1023 CPF1054 CPF2103 CPF2105 CPA0701 CPF9898)

/* Monitor ERROR msgs, then exit. */
MONMSG (CPF0001 CPF0801 CPF1338 CPF2161 CPF9897 CPF9999) EXEC(GOTO ERREXIT)

IF COND(&WHS < 1) THEN(DO)
  SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*DIAG) +
  MSGDTA('Number of warehouses must be greater or equal +
  to the number of partitions. ')
  GOTO ERREXIT
ENDDO
IF COND(&NUMPARTS > &WHS) THEN( +
  DO)
  SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*ESCAPE) +
  MSGDTA('Number of warehouses must be greater or equal +
  to the number of partitions. ')
  GOTO ERREXIT
ENDDO
CHOSYSVAL SYSVAL(QRYDEGREE) VALUE(*NONE)
/* set the system value so the index build does not blow up */
/*-----*/
SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*STATUS) +
  MSGDTA('CRTTPCCDB is starting...') TOPGMQ(*EXT)

RTVSYSVAL SYSVAL(QDATE) RTNVAR(&DATE)

CHKOBJ OBJ(QSYS/&DBLIB) OBJTYPE(*LIB)
MONMSG CPF9801 EXEC(DO)
  RCVMSG RMV(*YES)
  GOTO CRTLIB
ENDDO
SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +
  MSGDTA('Database library exist delete first then call CRTTPCCDB')
GOTO ENDPGM
CRTLIB: CRTLIB &DBLIB TEXT('Data base library ' *CAT &DATE) ASP(&DBASP)
MONMSG CPF2111 EXEC(RCVMSG RMV(*YES))

/* MSGQ for Status and Info messages. */
CRTMSGQ MSGQ(&DBLIB/&MSGQ)
MONMSG CPF2112 EXEC(CLRMSGQ MSGQ(&DBLIB/&MSGQ))

CRTMSGQ MSGQ(&DBLIB/TPCCBLMSG)
MONMSG CPF2112 EXEC(CLRMSGQ MSGQ(&DBLIB/TPCCBLMSG))

CHGVAR VAR(&SST(&MSGDATA 59 6)) VALUE(&WHS)
SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +
  MSGDTA(&MSGDATA) TOMSGQ(&DBLIB/&MSGQ)

/* Other pgms will be called from library we are running in. */
RTVBJD OBJ(CRTTPCCDB) OBJTYPE(*PGM) RTNLIB(&PGLIB)

/*-----*/
ENDSBS &BLDSBS OPTION(*IMMED)
DLTSSD &PGLIB/&BLDSBS
MONMSG CPF2105 EXEC(RCVMSG RMV(*YES))
DLTJOB &PGLIB/TPCCBLJOB
MONMSG CPF2105 EXEC(RCVMSG RMV(*YES))
CRTSSD &PGLIB/&BLDSBS POOLS((1 *BASE)) TEXT('TPCCBLD Jobs Subsystem')

DLTCLS &PGLIB/TPCCBLDCLS
MONMSG CPF2105 EXEC(RCVMSG RMV(*YES))
CRTCLS &PGLIB/TPCCBLDCLS RUNPT(21) TIMESLICE(2000) PURGE(*NO)

DLTJOBQ JOBQ(&PGLIB/TPCCBLDJOBQ)
MONMSG CPF2105 EXEC(RCVMSG RMV(*YES))
CRTJOBQ JOBQ(&PGLIB/TPCCBLDJOBQ)
ADDJOBQE SBDQ(&PGLIB/&BLDSBS) JOBQ(&PGLIB/TPCCBLDJOBQ) MAXACT(*NOMAX)

CRTJOBQ JOBQ(&PGLIB/TPCCBLDJOBQ) JOBQ(&PGLIB/TPCCBLDJOBQ) +
  RTGDTA(&PGLIB) INLLIB(&PGLIB) QTEM QGPL) +
  LOG(4 0 *SECLVL) INQMSGRPY(*DDT)
ADDRTE SBDQ(&PGLIB/&BLDSBS) SEQNBR(99) CMPVAL(*ANY) PGM(QCMD) +
  CLS(&PGLIB/TPCCBLDCLS)
STRSBS &PGLIB/&BLDSBS

/*-----*/
/* Create the TPCC data areas needed. */
/*-----*/
SBMJOB CMD(CALL PGM(&PGLIB/CRTDBDTP) PARM(&DBLIB)) +
  JOB(CRTDBDTP) JOBQ(&PGLIB/TPCCBLDJOBQ) MSGQ(&DBLIB/TPCCBLMSG)

/*-----*/
/* Before we start filling the DB, check if enough space */
/* exists in the specified ASP. */
/*-----*/
CHGVAR &WHSC VALUE(&WHS)
CHGVAR &WHSCN VALUE(&WHSC *CAT &NULL)
CHGVAR &DBASPC VALUE(&DBASP)
CHGVAR &DBASPCN VALUE(&DBASPC *CAT &NULL)

CALL &PGLIB/CHKWHSPACE PARM(&WHSCN &DBASPCN)

RCVMSG MSGQ(&DBLIB/TPCCBLMSG) MSGTYPE(*ANY) WAIT(*MAX)
/*-----*/
/* Update TPCC data areas values. */
/*-----*/
CHGDTAARA &PGLIB/TPCCDBLIB VALUE(&DBLIB)
CHGDTAARA &DBLIB/TPCCJRNLIB VALUE(&JRNLIB)
CHGDTAARA &DBLIB/TPCCWRHS VALUE(&WHS)
CHGDTAARA &DBLIB/TPCCDBVRS VALUE('34')
/* set the version to 3.4 for the database with new stock */

/*-----*/
/* Create Database files and fill them. */
/*-----*/
CALL &PGLIB/FILLPARTS PARM(&DBLIB &DBASP &WHS &NUMPARTS +
  &PARSUP)
/*-----*/
/* Post-Database file and pgm creation... */
/*-----*/

CHKOBJ OBJ(&DBLIB/TPCCMNGRCV) OBJTYPE(*DTAARA)

```

## C.20 CRTTPCCDB.CL:

```

MONMSG MSGID(CPF9801) EXEC(CRTDTAARA +
DTAARA(4DBLIB/TPCCMNGRCV) TYPE(*CHAR) +
LEN(10) VALUE(*SYSTEM))
CHGDTAARA DTAARA(4DBLIB/TPCCMNGRCV) VALUE(4MNGRCV)
CHKOBJ OBJ(4DBLIB/TPCCRCVSI2) OBJTYPE(*DTAARA)
MONMSG MSGID(CPF9801) EXEC(CRTDTAARA +
DTAARA(4DBLIB/TPCCRCVSI2) TYPE(*CHAR) +
LEN(10) TEXT('TPCC Journal receiver size +
option'))
CHGDTAARA DTAARA(4DBLIB/TPCCRCVSI2) VALUE(4RCVSIZEOPT)
CHKOBJ OBJ(4DBLIB/TPCCRCVTRH) OBJTYPE(*DTAARA)
MONMSG MSGID(CPF9801) EXEC(CRTDTAARA +
DTAARA(4DBLIB/TPCCRCVTRH) TYPE(*DEC) +
LEN(10) TEXT('Journal receiver threshold'))
CHGDTAARA DTAARA(4DBLIB/TPCCRCVTRH) VALUE(4THRESHOLD)

IF COND(4SAVEDB *EQ *Y) THEN(DO)
CHGVAR VAR(4NUMPARTS) VALUE(4NUMPARTS)
CHKOBJ OBJ(4JRNLIB) OBJTYPE(*LIB)
MONMSG MSGID(CPF9801) EXEC(+
CRTLIB LIB(4JRNLIB) ASP(4JRNASP))
SAVTPCC DBLIB(4DBLIB) +
MEDIA(4MEDIA) +
SAVFLIB(4SAVFLIB) SAVFASP(4SAVFASP) +
DEVN(4DEVNAME)
ENDDO
ELSE
CMD(DO)
CHGVAR VAR(4SST(4RCVSIZEMOD 1 2)) VALUE(' ')
CHGVAR VAR(4SST(4RCVSIZEMOD 3 10)) VALUE(4RCVSIZEOPT)
CALL 4PGLIB/STRTPCCJRN PARM(4DBLIB 4JRNLIB 4JRNASP +
4MNGRCV 4RCVSIZEMOD 4THRESHOLD)
CALL 4PGLIB/STRTPCCJRN PARM(4DBLIB)
ENDDO
RMVLIB(4DBLIB)
MONMSG CPF2104 EXEC(RCVMSG RMV(*YES))
ENDSBS 4BLDSBS OPTION(*IMMED)

/*-----*/
SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*COMP) +
MSGDTA('CRTTPCCDB completed successfully!')
SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +
MSGDTA('CRTTPCCDB completed successfully!') +
TOMSGQ(4TPCCDATM/4MSGQ)
MONMSG MSGID(CPF2403) EXEC(RCVMSG RMV(*YES))
MONMSG CPF2469 EXEC(RCVMSG RMV(*YES))
SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +
MSGDTA('CRTTPCCDB completed successfully!') +
TOMSGQ(4DBLIB/4MSGQ)
MONMSG MSGID(CPF2403) EXEC(RCVMSG RMV(*YES))
MONMSG CPF2469 EXEC(RCVMSG RMV(*YES))

GOTO ENDPGM

ERREXIT: SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +
MSGDTA('CRTTPCCDB failed!') TOMSGQ(4DBLIB/4MSGQ)
/* Ignore errors so we don't get in an infinite loop. */
MONMSG CPF0000

SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('CRTTPCCDB failed!') MSGTYPE(*ESCAPE)

ENDPGM: ENDPGM

```

## C.21 CRTTPCCJRN.CL:

```

/*-----*/
/* Program: CRTTPCCJRN */
/*-----*/
PGM PARM(4DBLIB 4JRNLIB 4JRNASP 4MNGRCV 4RCVSIZEOPT +
4THRESHOLD)

DCL 4DBLIB TYPE(*CHAR) LEN(10)
DCL 4JRNLIB TYPE(*CHAR) LEN(10)
DCL 4JRNASP TYPE(*DEC) LEN(2)
DCL 4MNGRCV TYPE(*CHAR) LEN(10)
DCL 4NUMPARTS TYPE(*DEC) LEN(2 0)
DCL 4CURRPART TYPE(*DEC) LEN(2 0)
DCL 4CURRPARTC TYPE(*CHAR) LEN(2)
DCL 4JRNNAME TYPE(*CHAR) LEN(10) VALUE('TPCCJRN ')
DCL 4JRNRCVNAME TYPE(*CHAR) LEN(10)
DCL 4JRNLIBC TYPE(*CHAR) TYPE(*CHAR) LEN(32)
DCL 4RCVSIZEOPT1 TYPE(*CHAR) LEN(10) VALUE(' ')
DCL 4RCVSIZEOPT2 TYPE(*CHAR) LEN(10) VALUE(' ')
DCL 4RCVSIZEOPT3 TYPE(*CHAR) LEN(10) VALUE(' ')
DCL 4THRESHOLD TYPE(*DEC) LEN(10)
DCL 4TESTASP TYPE(*DEC) LEN(2 0)

DCL 4DBASP TYPE(*DEC) LEN(2 0) /* Must be (2 0) */

/*-----*/
MONMSG MSGID(CPF9801 CPF2105 CPF2125)
MONMSG (CPF0001 CPF0801 CPF9897 CEE9901) EXEC(GOTO ERREXIT)

SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('CRTTPCCJRN: creating journals, etc ...') +
TOPGMQ(*EXT) MSGTYPE(*STATUS)

CHKOBJ OBJ(4JRNLIB) OBJTYPE(*LIB)
MONMSG MSGID(CPF9801) EXEC(+
DO)
CRTLIB LIB(4JRNLIB) ASP(4JRNNAME) +
TEXT('TPCC library to hold the journal receivers')
GOTO NORMA
ENDDO
RTVLIB LIB(4JRNLIB) ASP(4TESTASP)
IF COND(4JRNASP *NE 4TESTASP) THEN(+
DO)
DLTLIB LIB(4JRNLIB)
CRTLIB LIB(4JRNLIB) ASP(4JRNNAME) +
TEXT('TPCC library to hold the journal receivers')
ENDDO
NORMA:
/*-----*/
/* Get the number of parts */
/*-----*/
RTVMBRD FILE(4DBLIB/WRHS) NBRDTAMBS(4NUMPARTS)
CHGJOB INQMSGRPY(*SYSRPLY)
CHGVAR VAR(4RCVSIZEOPT1) VALUE(4SST(4RCVSIZEOPT 3 10))
IF COND(4RCVSIZEOPT1 *NE *NONE) THEN(+
DO)
IF COND(4SST(4RCVSIZEOPT 13 1) *EQ **) THEN(+
DO)
CHGVAR VAR(4RCVSIZEOPT2) VALUE(4SST(4RCVSIZEOPT 13 10))
IF COND(4SST(4RCVSIZEOPT 23 1) *EQ **) THEN(+
DO)
CHGVAR VAR(4RCVSIZEOPT3) VALUE(4SST(4RCVSIZEOPT 23 10))
ENDDO
ELSE
CMD(DO)
IF COND(4RCVSIZEOPT1 *EQ *MAXOPT1) THEN(+
CHGVAR VAR(4RCVSIZEOPT2) VALUE(*MINFKLEN))
ENDDO
ENDDO
IF COND(4NUMPARTS *EQ 1) THEN(DO)
ENDJRNPF FILE(*ALL) JRN(4DBLIB/TPCCJRN)
MONMSG MSGID(CPF9801) EXEC(RCVMSG RMV(*YES))
ENDDO

ELSE DO
CHGVAR VAR(4CURRPART) VALUE(1)
ENDJRNLOOP: CHGVAR VAR(4CURRPARTC) VALUE(4CURRPART)
CHGVAR VAR(4SST(4JRNNAME 8 2)) VALUE(4CURRPART)
ENDJRNPF FILE(*ALL) JRN(4DBLIB/4JRNNAME)
MONMSG MSGID(CPF9801) EXEC(RCVMSG RMV(*YES))
CHGVAR VAR(4CURRPART) VALUE(4CURRPART + 1)
IF COND(4CURRPART *LE 4NUMPARTS) THEN(GOTO ENDJRNLOOP)
ENDDO
CRTMSGQ MSGQ(4DBLIB/TPCCJRNMSG)
MONMSG CPF2112 EXEC(CLRMSGQ MSGQ(4DBLIB/TPCCJRNMSG))

DLTJRN JRN(4DBLIB/TPCCJRN*)
DLTRJRNRCV JRNRCV(4JRNLIB/TPC*)
MONMSG MSGID(CPF2117) EXEC(DO)
SNDPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('Receiver library in use!') MSGTYPE(*INFO)
GOTO ERREXIT
ENDDO

IF COND(4NUMPARTS *EQ 1) THEN(DO)
CRTJRNRCV JRNRCV(4JRNLIB/TPCCJRNRCV) ASP(4JRNASP) +
THRESHOLD(4THRESHOLD) TEXT('TPCC Journal Receiver')
ENDDO
ELSE DO
CHGVAR VAR(4CURRPART) VALUE(1)
CHGVAR VAR(4SST(4JRNRCVNAME 1 4)) VALUE(4TPCJ)
CHGVAR VAR(4SST(4JRNRCVNAME 7 4)) VALUE(4NRVC)
CRTJRNRCVLIB: CHGVAR VAR(4CURRPART) VALUE(4CURRPART)
CHGVAR VAR(4SST(4JRNRCVNAME 5 2)) VALUE(4CURRPARTC)
CRTJRNRCV JRNRCV(4JRNLIB/4JRNRCVNAME) ASP(4JRNASP) +
THRESHOLD(4THRESHOLD) +
TEXT('TPCC Journal Receiver (Split DB)')
CHGVAR VAR(4CURRPART) VALUE(4CURRPART + 1)
IF COND(4CURRPART *LE 4NUMPARTS) THEN(GOTO CRTJRNRCVLIB)
ENDDO

/* Get the DBLIB's ASP id. */
RTVJOB OBJ(4DBLIB) OBJTYPE(*LIB) ASP(4DBASP)
IF COND(4SST(4RCVSIZEOUT3 1 1) *NE *) THEN(+
DO)
IF COND(4MNGRCV *EQ *SYSTEM) THEN(DO)
IF COND(4NUMPARTS *EQ 1) THEN(DO)
CRTJRN JRN(4DBLIB/TPCCJRN) +
JRNRCV(4JRNLIB/TPCCJRNRCV) ASP(4DBASP) +
MSGQ(4DBLIB/TPCCJRNMSG) MNGRCV(*SYSTEM) +
DLTRCV(*YES) RCVSIZEOPT(4RCVSIZEOUT1 +
4RCVSIZEOUT2 4RCVSIZEOUT3) TEXT('TPCC Journal')
ENDDO
ELSE DO
CHGVAR VAR(4CURRPART) VALUE(1)
CRTJRNYSM: CHGVAR VAR(4CURRPART) VALUE(4CURRPART)
CHGVAR VAR(4SST(4JRNNAME 8 2)) VALUE(4CURRPARTC)
CHGVAR VAR(4SST(4JRNRCVNAME 5 2)) VALUE(4CURRPARTC)
CRTJRN JRN(4DBLIB/4JRNNAME) +
JRNRCV(4JRNLIB/4JRNRCVNAME) ASP(4DBASP) +
MSGQ(4DBLIB/TPCCJRNMSG) MNGRCV(*SYSTEM) +
DLTRCV(*NO) RCVSIZEOPT(4RCVSIZEOUT1 +
4RCVSIZEOUT2 4RCVSIZEOUT3) TEXT('TPCC +
Journal (Split DB)')
CHGVAR VAR(4CURRPART) VALUE(4CURRPART + 1)
IF COND(4CURRPART *LE 4NUMPARTS) THEN(GOTO CRTJRNYSM)
ENDDO /* end else */
ENDDO /* end IF COND(4MNGRCV *EQ *SYSTEM) THEN(DO) */
IF COND(4NUMPARTS *EQ 1) THEN(DO)
CRTJRN JRN(4DBLIB/TPCCJRN) +
JRNRCV(4JRNLIB/TPCCJRNRCV) ASP(4DBASP) +
MSGQ(4DBLIB/TPCCJRNMSG) MNGRCV(*USER) +
DLTRCV(*NO) RCVSIZEOPT(4RCVSIZEOUT1 +
4RCVSIZEOUT2 4RCVSIZEOUT3) TEXT('TPCC Journal')
ENDDO
ELSE DO
CHGVAR VAR(4CURRPART) VALUE(1)
CRTJRNYSM2: CHGVAR VAR(4CURRPART) VALUE(4CURRPART)
CHGVAR VAR(4SST(4JRNNAME 8 2)) VALUE(4CURRPARTC)
CHGVAR VAR(4SST(4JRNRCVNAME 5 2)) VALUE(4CURRPARTC)
CRTJRN JRN(4DBLIB/4JRNNAME) +
JRNRCV(4JRNLIB/4JRNRCVNAME) ASP(4DBASP) +
MSGQ(4DBLIB/TPCCJRNMSG) MNGRCV(*SYSTEM) +
DLTRCV(*NO) RCVSIZEOPT(4RCVSIZEOUT1 +
4RCVSIZEOUT2) TEXT('TPCC +
Journal (Split DB)')
CHGVAR VAR(4CURRPART) VALUE(4CURRPART + 1)
IF COND(4CURRPART *LE 4NUMPARTS) THEN(GOTO CRTJRNYSM2)
ENDDO /* end else */
ENDDO /* end IF COND(4MNGRCV *EQ *SYSTEM) THEN(DO) */
IF COND(4NUMPARTS *EQ 1) THEN(DO)
CRTJRN JRN(4DBLIB/TPCCJRN) +
JRNRCV(4JRNLIB/TPCCJRNRCV) ASP(4DBASP) +
MSGQ(4DBLIB/TPCCJRNMSG) MNGRCV(*USER) +
DLTRCV(*NO) RCVSIZEOPT(4RCVSIZEOUT1 +
4RCVSIZEOUT2) TEXT('TPCC Journal')
ENDDO
ELSE DO
CHGVAR VAR(4CURRPART) VALUE(1)
CRTJRNYSR2: CHGVAR VAR(4CURRPART) VALUE(4CURRPART)
CHGVAR VAR(4SST(4JRNNAME 8 2)) VALUE(4CURRPARTC)
CHGVAR VAR(4SST(4JRNRCVNAME 5 2)) VALUE(4CURRPARTC)
CRTJRN JRN(4DBLIB/4JRNNAME) +
JRNRCV(4JRNLIB/4JRNRCVNAME) ASP(4DBASP) +
MSGQ(4DBLIB/TPCCJRNMSG) MNGRCV(*USER) +
DLTRCV(*NO) RCVSIZEOPT(4RCVSIZEOUT1 +
4RCVSIZEOUT2) TEXT('TPCC +
Journal (Split DB)')
CHGVAR VAR(4CURRPART) VALUE(4CURRPART + 1)
IF COND(4CURRPART *LE 4NUMPARTS) THEN(GOTO CRTJRNYSR2)
ENDDO /* else */
ENDDO /* elseif COND(4MNGRCV *EQ *SYSTEM) THEN(DO) */
GOTO EXIT
ENDDO /* end of rcvsizeout2 set */
/* last case only rcvsizeout1 has data */
IF COND(4MNGRCV *EQ *SYSTEM) THEN(DO)
IF COND(4NUMPARTS *EQ 1) THEN(DO)
CRTJRN JRN(4DBLIB/TPCCJRN) +

```

```

JRNRCV (&JRNLIB/TPCCJRNRCV) ASP (&DBASP) +
MSGQ (&DBLIB/TPCCJRNMSG) MNGRCV (*SYSTEM) +
DLTRCV (*NO) RCVSIZOPT (&RCVSIZOUT1) +
TEXT ('TPCC Journal')
ENDDO
ELSE DO
CHGVAR VAR (&CURRPART) VALUE (1)
CRTJRNYS1: CHGVAR VAR (&CURRPART) VALUE (&CURRPART)
CHGVAR VAR (&SST (&JRNNAME 8 2)) VALUE (&CURRPART)
CHGVAR VAR (&SST (&JRNNAME 5 2)) VALUE (&CURRPART)
CRTJRN JRN (&DBLIB/&JRNNAME) +
JRNRCV (&JRNLIB/&JRNNAME) ASP (&DBASP) +
MSGQ (&DBLIB/TPCCJRNMSG) MNGRCV (*SYSTEM) +
DLTRCV (*NO) RCVSIZOPT (&RCVSIZOUT1) +
TEXT ('TPCC Journal')
Journal (Split DB)
CHGVAR VAR (&CURRPART) VALUE (&CURRPART + 1)
IF COND (&CURRPART *LE &NUMPARTS) THEN (GOTO CRTJRNYS1)
ENDDO /* end of else */
ENDDO /* end of IF COND (&MNGRCV *EQ *SYSTEM) THEN (DO) */
ELSE DO
IF COND (&NUMPARTS *EQ 1) THEN (DO)
CRTJRN JRN (&DBLIB/TPCCJRN) +
JRNRCV (&JRNLIB/TPCCJRNRCV) ASP (&DBASP) +
MSGQ (&DBLIB/TPCCJRNMSG) MNGRCV (*USER) +
DLTRCV (*NO) RCVSIZOPT (&RCVSIZOUT1) +
TEXT ('TPCC Journal')
ENDDO
ELSE DO
CHGVAR VAR (&CURRPART) VALUE (1)
CRTJRNYSR1: CHGVAR VAR (&CURRPART) VALUE (&CURRPART)
CHGVAR VAR (&SST (&JRNNAME 8 2)) VALUE (&CURRPART)
CHGVAR VAR (&SST (&JRNNAME 5 2)) VALUE (&CURRPART)
CRTJRN JRN (&DBLIB/&JRNNAME) +
JRNRCV (&JRNLIB/&JRNNAME) ASP (&DBASP) +
MSGQ (&DBLIB/TPCCJRNMSG) MNGRCV (*USER) +
DLTRCV (*NO) RCVSIZOPT (&RCVSIZOUT1) +
TEXT ('TPCC Journal')
Journal (Split DB)
CHGVAR VAR (&CURRPART) VALUE (&CURRPART + 1)
IF COND (&CURRPART *LE &NUMPARTS) THEN (GOTO CRTJRNYSR1)
ENDDO /* end of else */
ENDDO /* end of IF COND (&MNGRCV *EQ *SYSTEM) THEN (DO) */
EXIT:
/*-----*/
/*
/* Update the database data arars with the new Journal
/* information.
/*
/*
/*
/*
/*-----*/
CHKOBJ OBJ (&DBLIB/TPCCJRNLIB) OBJTYPE ('DTAARA)
MONMSG MSGID (CPF9801) EXEC (CRTDTAARA +
DTAARA (&DBLIB/TPCCJRNLIB) TYPE (*CHAR) +
LEN (10))
CHGDTAARA &DBLIB/TPCCJRNLIB VALUE (&JRNLIB)
CHKOBJ OBJ (&DBLIB/TPCCMNGRCV) OBJTYPE ('DTAARA)
MONMSG MSGID (CPF9801) EXEC (CRTDTAARA +
DTAARA (&DBLIB/TPCCMNGRCV) TYPE (*CHAR) +
LEN (10))
CHGDTAARA DTAARA (&DBLIB/TPCCMNGRCV) VALUE (&MNGRCV)
CHKOBJ OBJ (&DBLIB/TPCCRCVSI2) OBJTYPE ('DTAARA)
MONMSG MSGID (CPF9801) EXEC (CRTDTAARA +
DTAARA (&DBLIB/TPCCRCVSI2) TYPE (*CHAR) +
LEN (10) TEXT ('TPCC Journal receiver size +
option'))
CHGDTAARA DTAARA (&DBLIB/TPCCRCVSI2) VALUE (&RCVSIZOUT1)
CHKOBJ OBJ (&DBLIB/TPCCRCVTRH) OBJTYPE ('DTAARA)
MONMSG MSGID (CPF9801) EXEC (CRTDTAARA +
DTAARA (&DBLIB/TPCCRCVTRH) TYPE (*DEC) +
LEN (10) TEXT ('Journal receiver threshold'))
CHGDTAARA DTAARA (&DBLIB/TPCCRCVTRH) VALUE (&THRESHOLD)
DLTMSGQ MSGQ (&DBLIB/TPCCJRNMSG)
MONMSG CPF0000
/*-----*/
SNDPGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) +
MSGDTA ('CRTTPCCJRN completed successfully!') MSGTYPE (*COMP)
GOTO ENDPGM
ERREXIT: SNDPGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) +
MSGDTA ('CRTTPCCJRN failed!') MSGTYPE (*ESCAPE)
ENDPGM: ENDPGM

```

## C.22 CRTWHFILE.CL:

```

/*-----*/
/*
/* Program: CRTWHFILE
/*
/* Purpose: Create and fill the Warehouse file.
/*
/* Params: Database library name
/* File name
/* Starting warehouse id (null terminated)
/* Number of warehouses (null terminated)
/*-----*/
PGM PARM (&DBLIB &WHFILE &STRWH &NUMWH &RECOMPFILE)
DCL &DBLIB TYPE (*CHAR) LEN (10)
DCL &WHFILE TYPE (*CHAR) LEN (10)
DCL &STRWH TYPE (*DEC) LEN (6)
DCL &NUMWH TYPE (*DEC) LEN (6)
DCL &RECOMPFILE TYPE (*CHAR) LEN (1)
DCL &STRWHC TYPE (*CHAR) LEN (6)
DCL &NUMWHC TYPE (*CHAR) LEN (6)
DCL &STRWHNULL TYPE (*CHAR) LEN (7)
DCL &NUMWHNULL TYPE (*CHAR) LEN (7)
DCL &NULLCH TYPE (*CHAR) LEN (1) VALUE (X'00')
DCL &WHFILEDEF TYPE (*CHAR) LEN (10) VALUE (WRHS)
DCL &NUM1 TYPE (*DEC) LEN (6)
DCL &NUM2 TYPE (*DEC) LEN (6)
DCL &PGMLIB TYPE (*CHAR) LEN (10)
DCL &MSGQ TYPE (*CHAR) LEN (10) VALUE (TPCCMSG)
DCL &MDATA TYPE (*CHAR) LEN (49) +
VALUE ('CRTWHFILE program is filling the WRHS db file...')
MONMSG MSGID (CPF0000 MCH0000) EXEC (GOTO ERREXIT)
/* Other pgms will be called from library we are running in. */
RTV0BJD OBJ (CRTWHFILE) OBJTYPE (*PGM) RTNLIB (&PGMLIB)
/*-----*/
/* Create the WRHS file and programs.
/*-----*/
CHGVAR VAR (&SST (&MDATA 34 5)) VALUE (&WHFILE)
SNDPGMMSG MSGID (CPF9897) MSGDTA (&MDATA) +
MSGF (QSYS/QCPFMMSG) TOPGMQ (*EXT) MSGTYPE (*STATUS)
SNDPGMMSG MSGID (CPF9897) MSGDTA (&MDATA) +
MSGF (QSYS/QCPFMMSG) MSGTYPE (*INFO) TOMSGQ (&DBLIB/&MSGQ)

```

## C.22 CRTWHFILE.CL:

```

DCL &DBLIB &WHFILE
MONMSG MSGID (CPF2105) EXEC (RCVMSG RMV (*YES))
CRTYPE FILE (&DBLIB/&WHFILE) SRCMBR (WRHS) +
SIZE (*NMAX) WAITRCD (2) LVLCHK (*NO)
IF (&RECOMPFILE *EQ 'Y') THEN (DO)
OVRDBF FILE (&WHFILEDEF) TOFILE (&DBLIB/&WHFILE)
CRTCMOD &PGMLIB/FILLWH
CRTPGM PGM (&PGMLIB/FILLWH) +
MODULE (&PGMLIB/FILLWH &PGMLIB/COMMONTPCC &PGMLIB/TPCCSPACE) +
ENTMOD (&PGMLIB/FILLWH) BNDSRVPGM (TPCCTOOLS/TPCCTOOLS)
DLTOVR FILE (&WHFILEDEF)
ENDDO
CHGVAR &STRWHC VALUE (&STRWH)
CHGVAR &STRWHNULL VALUE (&STRWHC *CAT &NULLCH)
CHGVAR &NUMWHC VALUE (&NUMWH)
CHGVAR &NUMWHNULL VALUE (&NUMWHC *CAT &NULLCH)
CALL &PGMLIB/FILLWH PARM (&DBLIB &WHFILE &STRWHNULL &NUMWHNULL)
SNDPGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*COMP) +
MSGDTA ('CRTWHFILE program completed successfully!') +
TOMSGQ (&DBLIB/TPCCMSG)
GOTO ENDPGM
ERREXIT: SNDPGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) +
MSGDTA ('CRTWHFILE failed!') MSGTYPE (*ESCAPE)
SNDPGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*INFO) +
MSGDTA ('CRTWHFILE failed!') TOMSGQ (&DBLIB/&MSGQ)
ENDPGM: ENDPGM

```

## C.23 CSTMRVIEW.CL:

```

/*-----*/
/*
/* NAME: CSTMRVIEW
/*
/* PURPOSE: create logical views
/*
/*-----*/
PGM PARM (&DBLIB &NUMPARTS)
DCL &DBLIB TYPE (*CHAR) LEN (10)
DCL &NUMPARTS TYPE (*CHAR) LEN (2)
DCL &FILE1 TYPE (*CHAR) LEN (10)
DCL &FILE2 TYPE (*CHAR) LEN (10)
DCL &FILE3 TYPE (*CHAR) LEN (10)
IF COND (&NUMPARTS *EQ '01') THEN (DO)
CHGVAR &FILE1 VALUE (CSTMRLFCRT)
CHGVAR &FILE2 VALUE (CSTMRLFNAM)
CHGVAR &FILE3 VALUE (CSTMTR)
ENDDO
ELSE DO
CHGVAR &FILE1 VALUE (CSTMRCRT)
CHGVAR &FILE2 VALUE (CSTMTRAM)
CHGVAR &FILE3 VALUE (CSTMTR000)
CHGVAR &SST (&FILE1 9 2) VALUE (&NUMPARTS)
CHGVAR &SST (&FILE2 9 2) VALUE (&NUMPARTS)
CHGVAR &SST (&FILE3 9 2) VALUE (&NUMPARTS)
ENDDO
/* CREATE THE LAST/FIRST NAME VIEW OVER CSTMTRPF. */
CHKOBJ OBJ (&DBLIB/CSTMRLFCRT) OBJTYPE (*FILE)
MONMSG CPF9801 EXEC (CRTLF FILE (&DBLIB/CSTMRLFCRT) SRCMBR (&FILE1) +
WAITRCD (2) LVLCHK (*NO))
/* CREATE THE VIEW FOR ACCESS BY LAST NAME. */
CHKOBJ OBJ (&DBLIB/CSTMRLFNAM) OBJTYPE (*FILE)
MONMSG CPF9801 EXEC (CRTLF FILE (&DBLIB/CSTMRLFNAM) SRCMBR (&FILE2) +
WAITRCD (2) LVLCHK (*NO))
/* CREATE THE UNIQUE VIEW FOR CSTMTRPF. */
CHKOBJ OBJ (&DBLIB/CSTMTR) OBJTYPE (*FILE)
MONMSG CPF9801 EXEC (CRTLF FILE (&DBLIB/CSTMTR) SRCMBR (&FILE3) +
WAITRCD (2) LVLCHK (*NO))
SNDPGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) +
MSGDTA ('CSTMTRVIEW completed successfully!') MSGTYPE (*COMP)
ENDPGM

```

## C.24 FILCUSFILE.CL:

```

/*-----*/
/*
/* NAME: FILCUSFILE
/*
/* PURPOSE: fill the following Database files:
/* - CSTMTRPF or for split DB: CSTMTRPF01 to CSTMTRPF32
/*-----*/
PGM PARM (&DBLIB &DBFILE &STRWH &NUMWHS &RECOMPFILE)
DCL &DBLIB TYPE (*CHAR) LEN (10)
DCL &DBFILE TYPE (*CHAR) LEN (10)
DCL &STRWH TYPE (*DEC) LEN (6)
DCL &NUMWHS TYPE (*DEC) LEN (6)
DCL &RECOMPFILE TYPE (*CHAR) LEN (1)
DCL &NUM1 TYPE (*DEC) LEN (6)
DCL &NUM2 TYPE (*DEC) LEN (6)
DCL &STRWHC TYPE (*CHAR) LEN (6)
DCL &NUMWHC TYPE (*CHAR) LEN (6)
DCL &STRWHNULL TYPE (*CHAR) LEN (7)
DCL &NUMWHNULL TYPE (*CHAR) LEN (7)
DCL &NULLCH TYPE (*CHAR) LEN (1) VALUE (X'00')
DCL &PGMLIB TYPE (*CHAR) LEN (10)
DCL &MSGQ TYPE (*CHAR) LEN (10) VALUE (TPCCMSG)
DCL &STOCKSRC TYPE (*CHAR) LEN (10) VALUE (STOCKRPF)
DCL &STOCKFILE TYPE (*CHAR) LEN (10) VALUE (STOCKRPF)
DCL &MDATA TYPE (*CHAR) LEN (44) +
VALUE ('FILCUSFILE: filling the db file...')
/*-----*/
MONMSG MSGID (CPF0000 MCH0000) EXEC (GOTO ERREXIT)
RTV0BJD OBJ (FILCUSFILE) OBJTYPE (*PGM) RTNLIB (&PGMLIB)
MONMSG CPF9811 EXEC (DO)
SNDPGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*DIAG) +
MSGDTA ('Add the library with FILCUSFILE pgm to lib list!')
GOTO ERREXIT
ENDDO
SNDPGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) TOPGMQ (*EXT) +
MSGDTA ('FILCUSFILE program starting...') MSGTYPE (*STATUS)

```

```

CRTMSGQ MSGQ(&DBLIB/&MSGQ)
MONMSG CPF2112 EXEC(RCVMSG RMV(*YES))

CALL TPCCTOOLS/SETTOOLIBL

ADDLIBL &DBLIB
MONMSG CPF2103 EXEC(RCVMSG RMV(*YES))

/*-----*/
IF (&RECCOMPILE *EQ 'Y') THEN(DO)

OVRDBF &CSTMRSRC TOFILE(&DBLIB/&DBFILE) OVRSCOPE(*JOB)
MONMSG CPF2103 EXEC(RCVMSG RMV(*YES))

CRTCMOD &PGMLIB/FILLCUS

CRTPGM &PGMLIB/FILLCUS +
MODULE(&PGMLIB/FILLCUS &PGMLIB/COMMONTPCC +
&PGMLIB/TPCCSPACE) +
ENTMOD(&PGMLIB/FILLCUS) BNDSRVPGM(TPCCTOOLS/TPCCCTOOLS)

DLTOVR FILE(&CSTMRSRC) LVL(*JOB)
ENDDO

CHGVAR VAR(&ST(&MDATA 25 9)) VALUE(&DBFILE)
SNDRPGMSG MSGID(CPF9897) MSGDTA(&MDATA) +
MSGF(QSYS/QCPFMMSG TOPGM(*EXT) MSGTYPE(*STATUS)
SNDRPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG MSGTYPE(*INFO) +
MSGDTA(&MDATA) TOMSGQ(&DBLIB/&MSGQ))

CHGVAR &NUMWHSC VALUE(&NUMWHSC)
CHGVAR &NUMWHNULL VALUE(&NUMWHSC *CAT &NULLCH)
CHGVAR &STRWHC VALUE(&STRWH)
CHGVAR &STRWHNULL VALUE(&STRWHC *CAT &NULLCH)

/* WRKSYSSTS OUTPUT(*PRINT) RESET(*YES) */

CALL &PGMLIB/MASTCUSSTK PARM(&DBLIB &STRWHNULL &NUMWHNULL &DBFILE)

/* WRKSYSSTS OUTPUT(*PRINT) RESET(*NO) */

SNDRPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG MSGTYPE(*COMP) +
MSGDTA('FILLCUSFILE program completed successfully!'))
SNDRPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG MSGTYPE(*COMP) +
MSGDTA('FILLCUSFILE program completed successfully!')) +
TOMSGQ(&DBLIB/&MSGQ)

GOTO ENDPGM

ERREXIT: SNDRPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG +
MSGDTA('FILLCUSFILE failed!')) MSGTYPE(*ESCAPE)
SNDRPGMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG MSGTYPE(*INFO) +
MSGDTA('FILLCUSFILE failed!')) TOMSGQ(&DBLIB/&MSGQ)

ENDPGM: ENDPGM

```

## C.25 FILLCUS.C:

```

/*-----*/
/*
File name: FILLCUS
Purpose: Fill the Customer file.
NOTE: The logic is the same as in the FILLSTOCK file.
PARMS: database library name
Fill Q name
write Q name
Fill space name
file id
instance number
starting warehouse number
number of warehouses
our job number in the scheme of things
total number of jobs to fill data
*/
/*-----*/
/* Includes.
*/
#include <stdlib.h>
#include <errno.h>
#include <recio.h>
#include <string.h>
#include <xxfsbk.h>
#include <mlib.h>
#include <micomput.h>
#include <decimal.h>
#include <qsaptrus.h>
#include <qwtchjg.h>
#include <mih/cpybytes.h>
#include <mih/cpyblap.h>
#include <mih/triml.h>
#include <mih/mattod.h>
#include <qsysinc/mih/deg>
#include <qsysinc/mih/rsivsp>
#include <os4msg.h>
#include <filldata.h>
#include <commentpcc.h>
#include <tpccspace.h>

/* MAPINC generates structures for the database file.
*/
#pragma mapinc("cstmrpf", "libl/cstmrpf(*all)", "both", "d_P", "tpcc")
#include "cstmrpf"

/* Defines
*/
#define DBLIB_PARM 1
#define FILL_Q_PARM 2
#define WRITE_Q_PARM 3
#define FILL_SPACE_PARM 4
#define FILE_ID_PARM 5
#define INST_NUMBER_PARM 6
#define STRWH_PARM 7
#define NUMWH_PARM 8
#define JOB_NUMBER_PARM 9
#define NUMBER_JOBS_PARM 10

/* Global declares
*/
int file_id;
int inst_number;

char dblib[11];
char msgtxt[100];
char msgq_g[20];
char pgmname_g[11];
char pgmlib_g[11];

SYSPTR write_q_ptr = NULL;
ENQ_Msg_Prefix_T ENQ_msg_prefix;

/*-----*/

```

```

/* Function: error_cleanup
*/
void error_cleanup(void)
{
char Q_msg_text[MSG_LENGTH];

add_error_jobs(dblib, file_id, inst_number);
if (NULL != write_q_ptr)
{
sprintf(Q_msg_text, "Q"); /* Quit. */
enq(write_q_ptr, &ENQ_msg_prefix, Q_msg_text);
}

printf(msgtxt, "%s program failed!", pgmname_g);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDRY, 1);
return;
} /* End: error_cleanup() */

/*-----*/
/* Function: fill_customer
*/
void fill_customer(unsigned long record_number,
unsigned long num_record_per_IO_block,
unsigned long max_record,
char *data_space, int StartWH, int NumWH)
{
int Cid, wid, Dist;
int iX;
int index;

unsigned long record_count; /* Number of records filled in block. */
unsigned long current_record; /* Current record being processed. */

char tstr[21];
char chNumStr[11] = "0123456789";

tpcc_CSRCD_both_t *CustRec;
MI_Time CurrTime; /* Current time from MATTOD. */

/*-----*/
/* Start executable code.
*/
current_record = record_number; /* Find the current record */
record_count = 0;
while ((record_count < num_record_per_IO_block) &&
(current_record <= max_record))
{
index = record_count * sizeof(tpcc_CSRCD_both_t); /* Get the position */
CustRec = (tpcc_CSRCD_both_t *)data_space[index];
wid = ((current_record - 1) % NumWH) + 1;
CustRec->CWID = wid + StartWH; /* note StartWH starts at 0 */
CustRec->CDID = Dist = (((current_record - 1) / NumWH) % 10) + 1;
CustRec->CID = Cid = ((current_record - 1) / (10 * NumWH)) + 1;

/* Build cfirst */
memset(CustRec->CFIRST, ' ', 16);
if ((1 == wid) && (1 == Dist) && (Cid==1))
{
sprintf(CustRec->CFIRST, "C load = %d", 7);
CustRec->CFIRST[strlen(CustRec->CFIRST)] = ' '; /* put in space */
} else {
iX = RANDOM(0, 99);
strncpy(CustRec->CFIRST, szFName[iX], strlen(szFName[iX]));
}

/* Build cinit. */
strncpy(CustRec->CINIT, "OE", 2);

/* Build clast */
if (Cid < 1001)
{
CreateCust_lastname((Cid-1), tstr);
} else {
CreateCust_lastname((NEW_RAND(255, 0, 999)), tstr);
}
cpyblap(CustRec->CLAST, 16, tstr, strlen(tstr), ' ');

CreateStrPad(10, 20, CustRec->CADDR1);
CreateStrPad(10, 20, CustRec->CADDR2);

/* build city */
iX = RANDOM(0, 99);
cpyblap(CustRec->CCITY, 20, CITY_ARR[iX], strlen(CITY_ARR[iX]), ' ');

/* build state */
strncpy(CustRec->CSTATE, state_arr[iX], 2);

/* build zip code */
CREATE_ZIP(CustRec->CZIP);

/* build phone # */
for (iX = 0; iX < 16; ++iX)
{
CustRec->CPHONE[iX] = chNumStr[(RANDOM(0, 9))];
}

/* Build credit limit */
CustRec->CCRDLIM = 50000.00;

/* Build credit status */
if ((RANDOM(0, 9) > 0)
{
CustRec->CCREDIT[0] = 'G';
} else {
CustRec->CCREDIT[0] = 'B';
}
CustRec->CCREDIT[1] = 'C';
/* build Cust balance */
CustRec->CBAL = -10.00;
/* build Cust YTD payments */
CustRec->CYTD = 10.00;
/* build Cust pay count & delivery count */
CustRec->CPAYCNT = 1;
CustRec->CDELCLNT = 0;

/* build Cust Discount */
CustRec->CDCT = ((RANDOM(0, 5000)) / 10000.00);

/* Build Cust Data */
CreateStrPad(300, 500, CustRec->CDATA);

/* Build Cust Date & Time */
mattod(CurrTime);
CurrTime[7] = BASE_TIME;
memcpy(CustRec->CLTOD, CurrTime, sizeof(MI_Time));

++record_count;
++current_record;
} /* end of while loop. */

return;
} /* end of fill_customer */

/*-----*/
/*
Function: main
*/

```



```

int main(int argc, char **argv)
{
    short  block, job_number;
    int    StartWH;
    int    NumWH;
    int    number_jobs, number_blocks_per_job;
    error_structure errCode;
    unsigned int lib_len;

    char  *slash_pos;

    char  fill_space[21];
    char  *fill_space_ptr; /* Always points to start of fill space. */
    char  *fill_space_ptr2; /* Points to different blocks with space. */

    char  fill_Q[11];
    char  write_Q[11];
    _SYSPTR fill_Q_ptr;

    unsigned long rec_number; /* Relative record number. */
    unsigned long num_record_per_IO_block;
    unsigned long record_skip_size;
    unsigned long last_record = 0;

    char  Q_msg_text[MSG_LENGTH];
    _DEQ_Msg_Prefix_T DEQ_msg_prefix;

    struct {
        Qus_Job_Change_Information_t jci;
        Qus_JOB0100_t x;
        int priority;
    } chgI;

    ExcData_t volatile exD = {0, "HANDLE "};

    /*-----*/
    /* Start executable code. */
    /*-----*/

    /*-----*/
    /* Set up the exception handler. Handle all escapes & FC excps. */
    /* Branch to: ERR when an error occurs. */
    /*-----*/
    #pragma exception_handler \
    (ERR, 0, 0, _C2_MH_ESCAPE | _C2_MH_FUNCTION_CHECK, _CTLA_HANDLE)

    errCode.bytes_prov = 0;

    /*-----*/
    /* Get the library name that we are running in. */
    /*-----*/
    slash_pos = strchr(argv[0], '/');
    lib_len = slash_pos - argv[0];
    CPYBYTES(pgmlib_gl, argv[0], lib_len);
    pgmlib_gl[lib_len] = '\0';

    ++slash_pos; /* Skip the '/' that follows the pgmlib field. */
    CPYBYTES(pgmname_gl, slash_pos, 10);
    pgmname_gl[triml(pgmname_gl, " ")] = '\0';

    /*-----*/
    /* TPC message queue to send our build messages to. */
    /*-----*/
    CPYBYTES(msgq_gl, BLD_MSGQ_NAME, 10);
    _CPYBYTES(4(msgq_gl[10]), argv[DBLIB_PARM], 10);

    dblib[10] = '\0';
    CPYBYTES(dblib, argv[DBLIB_PARM], 10);
    dblib[triml(dblib, " ")] = '\0';

    _CPYBYTES(fill_space, argv[FILL_SPACE_PARM], 10);
    _CPYBYTES(fill_space+10, argv[DBLIB_PARM], 10);

    job_number = atoi(argv[JOB_NUMBER_PARM]);

    /*-----*/
    sprintf(msgtxt, "FILLCUS: filling user space %.10s in library %s...",
            fill_space, dblib);
    Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, INFO_MSG, msgq_gl, 1, NULL);

    StartWH = atoi(argv[STRWH_PARM]) - 1;
    instn_number = atoi(argv[INST_NUMBER_PARM]);
    /*-----*/
    /* Calculate the number of records and the starting record. */
    /* Number of records is the (io block size) / record size. */
    /*-----*/
    NumWH = atoi(argv[NUMWH_PARM]);
    file_id = atoi(argv[FILE_ID_PARM]); /* Customer or Stock file? */

    num_record_per_IO_block = IO_BLOCK_SIZE / sizeof(tpcc_CSRCD_both_t);
    last_record = NumWH * MAX_DISTRICTS * CUSTPERDIST;

    /*-----*/
    /* Set Queuing values. */
    /*-----*/
    DEQ_msg_prefix.Wait_Forever = 1; /* Wait forever. */
    ENQ_msg_prefix.Msg_Len = MSG_LENGTH;

    fill_Q[10] = '\0';
    CPYBYTES(fill_Q, argv[FILL_Q_PARM], 10);
    fill_Q[triml(fill_Q, " ")] = '\0';
    write_Q[10] = '\0';
    CPYBYTES(write_Q, argv[WRITE_Q_PARM], 10);
    write_Q[triml(write_Q, " ")] = '\0';

    if (NULL == (fill_Q_ptr = rslvsp(Usrq, fill_Q, dblib, AUTH_ALL)))
    {
        sprintf(msgtxt, "Could not resolve to user queue, %s, in library %s.",
                fill_Q, dblib);
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
        error_cleanup();
        return(-1);
    }
    if (NULL == (write_Q_ptr = rslvsp(Usrq, write_Q, dblib, AUTH_ALL)))
    {
        sprintf(msgtxt, "Could not resolve to user queue, %s, in library %s.",
                write_Q, dblib);
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
        error_cleanup();
        return(-1);
    }

    QUSPTRUS(fill_space, &fill_space_ptr, &errCode); /* Get the ptr */
    fill_space_ptr2 = fill_space_ptr;

    /* Set up random number generator */
    jrand(get_seed());

    /*-----*/
    /* Find the starting record number. */
    /*-----*/
    number_jobs = atoi(argv[NUMBER_JOBS_PARM]);
    rec_number = (job_number * num_record_per_IO_block) + 1;
    record_skip_size = num_record_per_IO_block * number_jobs;

    block = 1;
    IO_BLOCKS_PER_JOB(number_blocks_per_job, num_record_per_IO_block,
                      last_record, number_jobs)

    /*-----*/
    /* Change our job priority so we run a little slower than the
    /* master job that called us. */
    /*-----*/

```

```

chgI.jci.Number_Fields_Entered = 1;
chgI.x.Length_Field_Info = sizeof(Qus_JOB0100_t) + sizeof(int);
chgI.x.Key_Field = 1802;
chgI.x.Type_Of_Data = 'B';
_CPYBYTES(chgI.x.Reserved, " ", 3);
chgI.x.Length_Data = 4;
chgI.priority = 50;

OWTCHGJB("JOB0100", &chgI, &errCode);

if (errCode.bytes_avail > 0)
{
    sprintf(msgtxt, "Unable to change FILLCUS job's priority.");
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
}

/*-----*/
while (rec_number <= last_record)
{
    /*-----*/
    /* Wait until the storage area is available for us to fill. */
    /*-----*/
    deq(4(DEQ_msg_prefix, Q_msg_text, fill_Q_ptr);

    if ('Q' == Q_msg_text[0])
    {
        add_successful_jobs(dblib, file_id, instn_number);
        sprintf(msgtxt,
                "Ending Fill job. 'Quit' message received on user queue.");
        QsendMsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, INFO_MSG, msgq_gl, 1, NULL);
        return;
    }

    /*-----*/
    /* Start filling... */
    /*-----*/
    fill_customer(rec_number, num_record_per_IO_block, last_record,
                 fill_space_ptr2, StartWH, NumWH);

    sprintf(Q_msg_text, "%02d", block); /* Convert to char. */
    enq(write_Q_ptr, &ENQ_msg_prefix, Q_msg_text);

    if (block == number_blocks_per_job)
    {
        block = 1;
        fill_space_ptr2 = fill_space_ptr; /* Reset to beginning of space. */
    }
    else {
        ++block;
        fill_space_ptr2 = fill_space_ptr2 + IO_BLOCK_SIZE;
    }
    rec_number += record_skip_size;
} /* End: while < last record. */

add_successful_jobs(dblib, file_id, instn_number);

sprintf(msgtxt,
        "FILLCUS: job number %d successfully filled user space %.10s.",
        job_number, fill_space);
Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, COMP_MSG, msgq_gl, 1, NULL);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, COMP_MSG, CS_CTLBOV, 1);
return;

#pragma disable_handler

/*-----*/
/* Any exceptions causes us to branch here. */
/*-----*/
ERR:
    error_cleanup();
    return(-1);
} /* end of main */

```

## C.26 FILLDATA.H:

```

#ifndef FILLDATA_H
#define FILLDATA_H
/** START HEADER FILE SPECIFICATIONS ***/
/* Header file name: FILLDATA */
/* Purpose: data used to fill Customer and History files. */
/** END HEADER FILE SPECIFICATIONS ***/

#define CITY_LENGTH 20
#define STATE_LENGTH 2

/* There are 100 names. */

char *szFName[] = {
    "FrankIford", "FranTarkenton", "GailStomymight",
    "JohnWayne", "JaneSeedickRun", "JeannieWithLight",
    "DavesNotHere", "MaryMartin", "JefferyTheWaiter",
    "CarrieFromQueen", "StephanKing", "KaraKing",
    "AnnieOakley", "BillClements", "FannyFarmer",
    "AllanLuden", "BerniceLatitude", "MelindaFisher",
    "ElaineBoosler", "FrankieAvalon", "Francine",
    "GaleGordon", "JohnBoyWalton", "JeanneChiley",
    "DavidBowie", "KurtiFranollie", "PhillipOfWales",
    "KerryCWilliams", "Stephanie", "AnnaSueBrighton",
    "BillyBob", "FayDunawayWith", "PamelaSueMartin",
    "AlanAlda", "BernardPffe", "LindaDayGeorge",
    "RobertConrad", "Franklin", "FrancisTheMule",
    "GayleMcDonald", "JanMurray", "JoanneCassidy",
    "MarkTheSpot", "Geoffrey", "PhyllisDiller",
    "DoogieHowser", "Anastasia", "AlvinChipmunk",
    "AlexHearon", "MelindaMcWilliam", "EllynBurnsnye",
    "JayeMorgan", "JoeFromKokomo", "JoeyHeatherton",
    "Gilligan", "TheSkipper2", "CarryBishop",
    "BenjaminFranklin", "AbrahamLincoln", "GeorgeWBush",
    "IncentPrice", "Fotisiekaber", "RichieCunningham",
    "Alexander", "PeterTheGreater", "MickeyMouse",
    "DonaldDuck", "BabyHuey", "CrystalGayle",
    "SnidelyWhiplash", "HastalaVista", "GeorgeWashington",
    "KareemAbdulJabar", "WhitneyHouston", "BobSmith",
    "PerryMason", "PaulDrake", "DellaMainStreet",
    "MissPiggy", "TheCookieMonster", "KermitTheFrog",
    "BluesBrothers", "BeaverCleaver", "Anriqueta",
    "Crenshaw", "AndyTaylor", "ClaytonWilliams",
    "BillClinton", "Elizabeth", "MichaelJordan",
    "PaulRevere", "JohnGlenn", "WashingtonCarver",
    "DanJohnson", "BubbaTexan", "GGordonLiddy",
    "ScottSimpson", "SamNight", "Montgomery",
    "NolanRyan", "AlbertEinstein" };

/* There are 100 cities. */
char *CITY_ARR[] = {
    "Saint_Paul", "Saint_Louis",
    "Concord_Grapes", "Trenton_New_Jersey",
    "Bismark_Doughnut", "Cheyenne_Wyoming",
    "Juneau_Alaska", "Honolulu_Hawaii",
    "Phoenix_The_Big_Bird", "Topeka_Kansas",
    "Montgomery", "Elmwood_Illinois",

```

```

"Madison Dolly " "Lansing Michigan "
"Frankfurt Germany " "Des Moines Iowa "
"Byron_Minnesota " "Pierre South Dakota "
"Dover Cliffs " "Helena Montana "
"San Antonio " "Saint Petersburg "
"Caribu Maine " "Sault_Saint_Marie "
"Los Angeles " "Minneapolis St Paul "
"Dallas Fort Worth " "Kalamazoo Michigan "
"The Land Of Oz " "Lexington Kentucky "
"Washington " "District Of Columbia "
"New Orleans " "Boise Idaho "
"Kensington Maryland " "Denver Colorado "
"Blue Mountain " "Riverside Ca "
"New York City " "Nashville Tennessee "
"Mayberry RD " "Berkeley California "
"Maryville Missouri " "San Francisco "
"Indianapolis " "Saint Joseph Mo "
"Rochester Minnesota " "Fort Worth "
"Holland Michigan " "South Orange NJ "
"Stillwater " "Eugene Oregon "
"Alexandria Virginia " "Salt Lake City "
"Manitowoc Wisconsin " "Philadelphia "
"Sioux Falls " "West Lafayette "
"New Brunswick NJ " "Rochester New York "
"Oakland Ca " "Las Vegas Nevada "
"Birmingham Alabama " "Arkadelphia Arkansas "
"San Juan Puerto Rico " "Omaha Nebraska "
"Walla Walla " "Baraboo Wisconsin "
"Atlanta Georgia " "Grand Forks ND "
"Tempe Arizona " "Knoxville TN "
"Fort Leavenworth " "Boston Massachusetts "
"Danville Virginia " "Baltimore Maryland "
"New Haven Ct " "Claremont California "
"Orango Michigan " "Providence "
"Jacksonville " "Columbia SC "
"London England " "Paris France "
"Chicago Illinois " "Albuquerque "
"Raleigh NC " "Kansas City Missouri "
"Tacoma Washington " "Oronoco Mn "
"Charleston WV " "Newark Delaware "
"Burlington Vermont " "Damariscotta Maine "
"Colorado Springs Co " "Mcgockoches Texas "
"Barbourville Ky " "Boca Raton Fl "
"Mary Of The Woods " "Heston And Isleworth" };

/*-----*/
/* There are 100 states. */
/*-----*/
char *state_arr[] = {
    "AL", "AK", "AZ", "CA", "CO", "CT", "DE", "FL", "GA",
    "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME", "MD",
    "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH", "NM",
    "NJ", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI", "SC",
    "SD", "TN", "TX", "UT", "VA", "WA", "WI", "WV",
    "AC", "AG", "AI", "AM", "AP", "AV", "AW", "CN", "CS", "DI",
    "DM", "DN", "DS", "ED", "EM", "EN", "HN", "HO", "IH", "IM",
    "IR", "IW", "JN", "KA", "KO", "LE", "LI", "NI", "NT", "NZ",
    "OC", "OM", "ON", "PR", "RA", "RO", "SK", "SM", "TC", "TG",
    "TV", "TV", "VI", "VN", "VW", "XT", "YK", "YN", "YW", "ZA" };

/*-----*/
/* There are 3 sets of characters that are concatenated */
/* to make the district information fields for Stock. */
/* Also used by FILLORD file. */
/*-----*/
char *DistIn1[] = {
    "abcdefghijklmnopqrstuvwxyz",
    "whatdoyouno", "notmuchthen", "whydoyouask",
    "Idonotknow", "sowheresit", "inthecities", "andthetowns" };

char *DistIn2[] = {
    "01", "02", "03", "04", "05", "06", "07", "08", "09", "10" };

char *DistIn3[] = {
    "mnopqrstuvw", "whydoyoudo", "interactive",
    "batchwork", "systemview", "performance",
    "Ineverwork", "Ialwaysplay", "nexttolast", "nowthefinal" };

/*-----*/ End of FILLDATA.H <-----*/
#endif

```

## C.27 FILLDIST.C:

```

/*-----*/
/* Name: FILLDIST */
/* Purpose: fill the district file. */
/*-----*/

#include <stdlib.h>
#include <string.h>
#include <decimal.h>
#include <micomput.h>
#include <mlib.h>
#include <recio.h>
#include <time.h>
#include <mlh/cpyblap.h>
#include <osmsg.h>
#include <commontpcc.h>
#include <filldata.h>

/*-----*/
/* Defines */
/*-----*/
#define DBLIB_PARM 1
#define DISTFILE_PARM 2
#define STRWH_PARM 3
#define NUMWH_PARM 4

/* MAPINC generates structures for the database files */
#pragma mapinc("dstrct", "LIBL/dstrct(*all)", "both", "d_p", "tpcc")
#include "dstrct"

/*-----*/
/* Name: main */
/*-----*/
int main(int argc, char **argv)
{
    short int strWH, endWH, curWH, iDist;
    int iX;

    char dblib[11];
    char distfile[11];
    char msgtxt[50];
    char msgq_gl[20];

    char curwhs[5];
    char curdist[3];
    char daddr[21];
    char tempatr[13];
    char tstr[5];

    _RFILE *Dist_File;
    tpcc_DSRCDC_both_t DRec;

```

```

time_t ltime;

/* Set up random number generator */
time(&ltime);
srand(ltime); /* seed the random num generator */

dblib[10] = '\0';
_CPYBYTES(dblib, argv[DBLIB_PARM], 10);
dblib[triml(dblib, ' ')] = '\0';

distfile[10] = '\0';
_CPYBYTES(distfile, argv[DISTFILE_PARM], 10);
distfile[triml(distfile, ' ')] = '\0';

strWH = atoi(argv[STRWH_PARM]);
endWH = (atoi(argv[NUMWH_PARM])) + strWH;

/* TPCC message queue to send our build messages to. */
_CPYBYTES(msgq_gl, "TPCCMSG", 10);
_CPYBYTES(&msgq_gl[10], argv[DBLIB_PARM], 10);

if (NULL == (Dist_File = _Ropen(distfile, "rr*")))
{
    sprintf(msgtxt, "Open of file, %s, failed!", distfile);
    Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDV, 1);
    return(-1);
}

for (curWH = strWH; curWH < endWH; ++curWH)
{
    for (iDist = 1; iDist <= 10; ++iDist)
    {
        DRec.DWID = curWH;
        DRec.DID = iDist;
        sprintf(curwhs, "%04d", curWH);
        sprintf(curdist, "%02d", iDist);

        /* build dname */
        strcpy(tempatr, "D");
        strcat(tempatr, curwhs);
        strcat(tempatr, curdist);
        CreateStr(1, 4, tstr);
        strcat(tempatr, tstr);
        cpyblap(DRec.DNAME, 10, tempatr, strlen(tempatr, ' '));

        /* build address1 */
        strcpy(daddr, curdist);
        sprintf(tstr, "%03d", (RANDOM(1,999)));
        strcat(daddr, tstr);
        strcat(daddr, " Ave.");
        CreateStr(1, 10, tempatr);
        strcat(daddr, tempatr);
        cpyblap(DRec.DADDR1, 20, daddr, strlen(daddr, ' '));

        /* build address2 */
        strcpy(daddr, "Bldg");
        strcat(daddr, " ");
        CreateStr(1, 12, tempatr);
        strcat(daddr, tempatr);
        cpyblap(DRec.DADDR2, 20, daddr, strlen(daddr, ' '));

        /* build city */
        iX = RANDOM(0,99);
        strncpy(DRec.DCITY, CITY_ARR[iX], CITY_LENGTH);
        strncpy(DRec.DSTATE, state_arr[iX], STATE_LENGTH);

        CREATE_ZIP(DRec.DZ1P);

        DRec.DTXA = (RANDOM(0,2000)) / 10000.0;
        DRec.DYTD = 30000.0;
        DRec.DNXTOR = 3001; /* Next order id */

        _Rwrite(Dist_File, (void *)&DRec, sizeof(DRec));
    }
}

_Rclose(Dist_File);
return(0);

} /* end of main() */

```

## C.28 FILLHIST.C:

```

/*-----*/
/* File name: FILLHIST */
/* Purpose: This program fills the history file. */
/*-----*/
/* Includes. */
/*-----*/

#include <stdlib.h>
#include <recio.h>
#include <string.h>
#include <xcfdbk.h>
#include <mlh/cpybytes.h>
#include <mlh/cpyblap.h>
#include <mlh/triml.h>
#include <mlh/mattod.h>
#include <errno.h>
#include <unistd.h> /* Has unlink() */
#include <sys/stat.h> /* Has lstat() */
#include <decimal.h> /* Used by hstry include file. */
#include <qcmdexc.h>
#include <commontpcc.h>
#include <tpccspace.h>
#include <os4msg.h>
#include <os4wait.h>

/* MAPINC generates structures for the database file. */
#pragma mapinc("hstry", "libl/hstry(*all)", "both", "d_p", "tpcc")
#include "hstry"

/*-----*/
/* Defines */
/*-----*/
#define DBLIB_PARM 1
#define DBFILE_PARM 2
#define STRWH_PARM 3
#define WH_PARM 4
#define LEVEL_PARM 5
#define INST_PARM 6
#define TOTAL_PARM 6

#define WH_PARM_LEN 6

/*-----*/
/* globals */
/*-----*/
char pgmlib_gl[11];
char pgmname_gl[11];
char msgq_gl[20];
char msgtxt[110];

```

```

/*-----*/
/* Function: multi_jobs */
/* Purpose: */
/* Return: -1 = error */
/* Any other value is number of jobs spawned. */
/*-----*/
int multi_jobs(char *dlib, char *dbfile, int str_wh, int total_whs,
               int jobs_alloc, int inst_num)
{
    int rc, times;
    int num_jobs;
    int num_wh;
    int wh_per_job;
    int have_extra; /* # of jobs with extra warehouse. */
    int lp; /* loop counter */
    unsigned int plen;

    char job_num[4] = " ";
    char dlib_pad[10];
    char dbfile_pad[10];
    char work_path[22];
    char hard_path[41];
    struct stat checker;

    /*-----*/
    /* Determine how many jobs to spawn. */
    /*-----*/
    if (jobs_alloc < total_whs)
    {
        num_jobs = jobs_alloc; /* Use all jobs available. */
    }
    else
    {
        num_jobs = total_whs;
    }

    wh_per_job = total_whs / num_jobs;
    have_extra = total_whs % num_jobs;

    if (0 == have_extra)
    {
        num_wh = wh_per_job;
    }
    else
    {
        /* Do an extra warehouse for this job. */
        num_wh = wh_per_job + 1;
    }

    /*-----*/
    /* Save job info so Master control can see and we can use later. */
    /*-----*/
    if (0 != set_plan_to_start(dlib, HISTORY_JOBS, inst_num, num_jobs))
    {
        printf(msgtxt,
              "FILLHIST can't set number of jobs (%d) it plans to spawn!", 1);
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
        return(-1);
    }

    printf(msgtxt, "FILLHIST plans to spawn %d jobs to fill the %s file.",
           num_jobs, dbfile);
    QsendMsg(MQ(IBM_MSG_FILE, "CPF9897", msgtxt, INFO_MSG, msgq_gl, 1, NULL);
    cpyblap(dlib_pad, 10, dlib, strlen(dlib), ' ');
    cpyblap(dbfile_pad, 10, dbfile, strlen(dbfile), ' ');

    printf(hard_path, "/QSYS.LIB/%s.LIB/%s.PGM", pgmlib_gl, pgmname_gl);
    printf(work_path, "%s/%0.7s", pgmlib_gl, pgmname_gl);
    plen = strlen(work_path);

    rc = 0;
    /*-----*/
    /* Set parms for the spawn. */
    /*-----*/
    for (lp=1; (lp <= num_jobs) && (-1 != rc); lp++)
    {
        /* Set job name number. */
        printf(work_path[plen], "%03d", lp); /* Add job number. */
        printf(job_num, "%03d", lp);

        if (lstat(work_path, checker) == 0) /* If the link exists */
        {
            /* Unlink the symbolic link. */
            /* (it may be to another library so it should be removed. */
            if (-1 == unlink(work_path))
            {
                char msgid[8];
                /* Send the errno message text message. */
                printf(msgid, "CPE%d", errno);
                QsendMsg(IBM_MSG_FILE, msgid, "", DIAG_MSG, CS_CUR_PGM, 1);

                printf(msgtxt, "Could NOT unlink path: '%s'", work_path);
                QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
            }
        } /* End: link exists. */

        if (0 == symlink(hard_path, work_path))
        {
            /*-----*/
            /* Spawn the job! */
            /*-----*/
            rc = spawn_FILL_job(dlib_pad, dbfile_pad, HISTORY_JOBS, inst_num,
                               str_wh, num_wh, NULL, work_path, job_num);

            str_wh = str_wh + num_wh; /* Next starting warehouse id. */
            if (lp == have_extra)
            {
                --num_wh; /* Start doing 1 less warehouse. */
            }
        }
        else
        {
            /*-----*/
            /* symlink not created. */
            /*-----*/
            char msgid[8];
            /* Send the errno message text message. */
            printf(msgid, "CPE%d", errno);
            QsendMsg(IBM_MSG_FILE, msgid, "", DIAG_MSG, CS_CTLBDY, 1);

            printf(msgtxt, "Could NOT create a symbolic link '%s' to %s",
                   work_path, hard_path);
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
            rc = -1; /* Exit from loop. */
        }
    } /* end: 'for' loop. */

    if (0 == rc)
    {
        rc = num_jobs;
        return(rc);
    }
} /* multi_jobs() */

/*-----*/
/* Function: main */
/*-----*/
int main(int argc, char **argv)
{
    int rc = 0;
    int times;
    int num_spawned;
    int inst_number;
    int max_allocated; /* over ride size */
    int hist_ovr_size;
    int Cid;
    short Dist;
    short wid;
    short StrWH, EndWH, numWH;

    unsigned int lib_len;
    char *slash_pos;

    _MI_Time CurrTime; /* Current time from MATDOD. */

    char call_level[3];
    char job_name[11] = " ";
    char dbfile[11];
    char dlib[11];
    char lib_file[17];
    char cmd[90];
    tpcsc HSRCD_both_t HstRec;
    _RFILE *Hstry_File;
    ExcData_t volatile exD = {0, "HANDLE "};

    /*-----*/
    if (argc < TOTAL_PARM)
    {
        printf(msgtxt, "%s program expected %s parms, received %d parm",
              pgmname_gl, (TOTAL_PARM - 1), (argc-1));
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
        exit;
    }

    /*-----*/
    /* Get the library name that we are running in now. */
    /*-----*/
    slash_pos = strchr(argv[0], '/');
    lib_len = slash_pos - argv[0];
    _CPYBYTES(pgmlib_gl, argv[0], lib_len);
    pgmlib_gl[lib_len] = '\0';

    ++slash_pos; /* Skip the '/' */
    _CPYBYTES(pgmname_gl, slash_pos, 10);
    pgmname_gl[triml(pgmname_gl, ' ')] = '\0';

    dlib[10] = '\0';
    _CPYBYTES(dlib, argv[DBLIB_PARM], 10);
    dlib[triml(dlib, ' ')] = '\0';

    dbfile[10] = '\0';
    _CPYBYTES(dbfile, argv[DBFILE_PARM], 10);
    dbfile[triml(dbfile, ' ')] = '\0';

    /* TPCSC message queue to send our build messages to. */
    _CPYBYTES(msgq_gl, "PFPCMSG ", 10);
    _CPYBYTES(&(msgq_gl[10]), argv[DBLIB_PARM], 10);

    /* Set library/file name to open and use. */
    printf(lib_file, "%s/%s", dlib, dbfile);
    StrWH = atoi(argv[STRWH_PARM]);
    numWH = atoi(argv[WH_PARM]);

    call_level[3] = '\0';
    _CPYBYTES(call_level, argv[LEVEL_PARM], 3);
    call_level[triml(call_level, ' ')] = '\0';

    printf(job_name, "%0.7s%03s", pgmname_gl, call_level);
    if (0 == strcmp(call_level, "000", 3))
    {
        if (1 == StrWH)
        {
            inst_number = 0;
        }
        else
        {
            inst_number = 1; /* split warehouse high group */
        }
    }

    /*-----*/
    /* Get max number of jobs we can use. */
    /*-----*/
    max_allocated = get_maxjobs_available(dlib, HISTORY_JOBS, inst_number);

    if ((1 < max_allocated) && (1 < numWH))
    {
        /*-----*/
        /* Spawn extra jobs to speed things up... */
        /*-----*/
        num_spawned = multi_jobs(dlib, dbfile, StrWH, numWH,
                                 max_allocated, inst_number);

        /*-----*/
        /* If 1 job failed to spawn, then don't wait on other spawned */
        /* jobs to complete. */
        /*-----*/
        if (-1 == num_spawned)
        {
            printf(msgtxt, "%s job failed!", job_name);
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
            return(-1);
        }

        /*-----*/
        /* All spawned successfully, wait on them to finish. */
        /*-----*/
        rc = wait_on_spawned_jobs(dlib, HISTORY_JOBS, inst_number, num_spawned);
    } /* end: allocated jobs > 1 and warehouses > 1 */
    else
    {
        /* Note: we don't have to query the available jobs, as we are */
        /* not starting another job. */
        /*-----*/
        /* Only up the count when we are called from CL program. */
        /* When spawned, our caller has already upped the count. */
        /*-----*/
        /* Save job info so Master control can see and use later. */
        /*-----*/
        if (0 != set_plan_to_start(dlib, HISTORY_JOBS, inst_number, 1))
        {
            printf(msgtxt,
                  "I can't set number of %s jobs (%d) I plan to start!",
                  get_db_filename(HISTORY_JOBS), 1);
            QsendMsg(MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
            return(-1);
        }

        times = 0;
        while ((0 != add_started_jobs(dlib, HISTORY_JOBS, inst_number))
              && (-1 != rc))
        {
            /*-----*/
            /* wait and try again... */
            /*-----*/
            tpcsc Wait(3, 0, NULL, NULL); /* Wait 3 seconds. */

            ++times; /* Total of 5 minutes wait time. */
            if (100 == times)
            {
                printf(msgtxt,

```

```

/* I can't add to the number of jobs I am starting! */
get_db_filename(HISTORY_JOBS));
Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_PGMBDY, 1);
return(-1);
}
} /* end of while(0!=add_started_jobs(dblib,HISTORY_JOBS,inst_number)*/
} /* end of else */
} /* end of if (0 == strcmp(call_level, "000", 3)) */
else
{
inst_number = atoi(argv(INST_PARAM)); /* get the parameter */
}
if (0 != strcmp(call_level, "000", 3)) /* if action required */
{
/*-----*/
/* Now that I have updated the # jobs started counter, use an */
/* exception handler to update space upon an error. */
/*-----*/
#pragma exception_handler \
(EXCP_HND, 0, 0, _C2_MH_ESCAPE | _C2_MH_FUNCTION_CHECK, _CTLA_HANDLE)
hist_ovr_size = IO_BLOCK_SIZE / sizeof(tpccc_HSRCD_Both_t);
sprintf(cmd,
"OVRDBF FILE(%s) TOFILE(%s/%s) SEQONLY(*YES sd) OVRSCOPE(*JOB)",
dbfile, dblib, dbfile, hist_ovr_size);
/* printf ("Override %s\n", cmd); */
#pragma exception_handler (QexcpHndr, exd, 0, _C2_MH_ESCAPE)
QCMDKX(cmd, strlen(cmd));
#pragma disable_handler
if (0 != exd.error)
{
sprintf(msgtxt, "%s program failed!", pgmname_gl);
Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
return(-1);
}
Hstry_File = _Ropen(dbfile, "ar,arseq=Y,bkrcd=Y,riofb=N");
if (NULL == Hstry_File)
{
sprintf(msgtxt, "Failed to open file: %s.", dbfile);
Qsendmsg_MQ(NULL, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
return(-1);
}
EndWH = StrWH + numWH;
sprintf(msgtxt,
"%s is starting to fill %s file from warehouse %d to %d.",
job_name, dbfile, StrWH, (EndWH - 1));
Qsendmsg_MQ(NULL, "CPF9897", msgtxt, INFO_MSG, msgq_gl, 1, NULL);
for (Cid = 1; Cid <= CUSTPERDIST; Cid++)
{
/*-----*/
/* if (0 == (Cid % 1000)) */
/* { */
/* printf(msgtxt, "%s starting on customer %d...", job_name, Cid); */
/* Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, COMP_MSG, msgq_gl, */
/* 1, NULL); */
/* } */
/*-----*/
for (wid = StrWH; wid < EndWH; wid++)
{
for (Dist = 1; Dist <= 10; Dist++)
{
HstRec.HCWID = wid;
HstRec.HWID = wid;
HstRec.HDID = wid;
HstRec.HCID = Dist;
HstRec.HCID = Cid;
HstRec.HAMT = 10.00;
CreateStrPad(12, 24, HstRec.HDATA);
/* Build Hist Date & Time. */
matmod(CurrTime);
CurrTime[7] = BASE_TIME; /* set the base time */
memcpy(HstRec.HTOD, CurrTime, sizeof(MI_Time));
Write(Hstry_File, (void *) &HstRec, sizeof(HstRec));
}
}
_Rclose(Hstry_File);
/* Disable the EXCP_HND handler. */
#pragma disable_handler
/* Send message now so it appears in outq before our caller's */
/* completion message. */
/*-----*/
sprintf(msgtxt,
"%s successfully completed filling warehouse %d to %d.",
job_name, StrWH, (StrWH+(numWH-1)));
Qsendmsg_MQ(NULL, "CPF9897", msgtxt, COMP_MSG, msgq_gl, 1, NULL);
times = 0;
while (0 != add_successul_jobs(dblib, HISTORY_JOBS, inst_number))
&& (-1 != rc))
{
/*-----*/
/* Wait and try again... */
/*-----*/
tpccc_Wait(3, 0, NULL, NULL); /* Wait 3 seconds. */
++times;
if (100 == times) /* Total of 5 minutes wait time. */
{
sprintf(msgtxt, "I can't set completed jobs value!");
Qsendmsg_MQ(NULL, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
return(-1);
}
}
} /* end of if (0 != strcmp(call_level, "000", 3)) */
if (0 == rc)
{
/* If we have NOT already sent a completion msg... */
if (0 == strcmp(call_level, "000", 3))
{
sprintf(msgtxt,
"%s program successfully completed filling warehouse %d to %d.",
pgmname_gl, StrWH, (StrWH+(numWH-1)));
Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, COMP_MSG, msgq_gl, 1, NULL);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, COMP_MSG, CS_CTLBDY, 1);
}
return(0);
}
else
{
sprintf(msgtxt, "%s program failed!", pgmname_gl);
Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
return(-1);
}
}
/*-----*/
/* Handle unexpected errors. */
}

```

```

/*-----*/
EXCP_HND;
rc = 0;
times = 0;
while (0 != add_error_jobs(dblib, HISTORY_JOBS, inst_number))
&& (-1 != rc))
{
/*-----*/
/* Wait and try again... */
/*-----*/
tpccc_Wait(3, 0, NULL, NULL); /* Wait 3 seconds. */
++times;
if (100 == times) /* Total of 5 minutes wait time. */
{
sprintf(msgtxt, "I can't set 'error' jobs value for %s!",
get_db_filename(HISTORY_JOBS));
Qsendmsg_MQ(NULL, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
return(-1);
}
}
sprintf(msgtxt, "%s program failed!", pgmname_gl);
Qsendmsg_MQ(NULL, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
return(-1);
}
} /* end of main */

```

## C.29 FILLITEM.C:

```

/*-----*/
/* File name: FILLITEM */
/* Purpose: This program fills the ITEM file. */
/* PARAMS: Database library name */
/*-----*/
/* Includes. */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <decimal.h>
#include <mh/cpybytes.h>
#include <mh/cpydiap.h>
#include <mh/trml.h>
#include <qmdexc.h>
#include <osams.h>
#include <commontpcc.h>
/* Defines. */
#define DBLIB_PARM 1
#define DBFILE_PARM 2
#define MAXITEMS 100000
/* MAPINC generates structures for the database files */
#pragma mapinc("item","LIBL/item(*all)","both","d_P","tpccc")
#include "item"
/* globals */
char msgtxt[100];
char msgq_gl[20];
_RFILE *Item_File;
/* There are 100 entries, each of length 24 */
char *ItemNameStr[] = {
"Ball_Point_Pen", "Sailboat_Fuel_Tank",
"Flowers_(Poppies)", "WCF_Road_Warrior_Game",
"Bird_House_Plans", "Pushbutton_TelePhone",
"Surgical_Gloves", "1_Gallon_Distilled_Water",
"Black_Labrador_Puppies", "Plain_Pockets_Pants",
"Electric_Plane", "Elephant_Ear_Shushroom",
"Gray_Hooded_Sweatshirts", "Lone_Ranger_Costume",
"Jiffy_Pain_Pills", "Devil's_Food_Cake",
"Full_Length_Fox_Fur", "AM_FM_Stereo_Car_Radio",
"Cheddar_Cheese_Ball", "Elephant_Ear_Shushroom",
"White_Traffic_Lane_Paint", "Plastic_Garbage_Pail",
"Four_Blade_Ceiling_Fans", "Doll_House_Furniture",
"Apple_Core_Remover", "Cola_Twelve_Pack",
"Italian_Food_Cook_Book", "Wool_Toe_Socks",
"Feel_Good_Vitamins", "100'_Rubber_Garden_Hose",
"Orange_Golf_Balls", "Snorkle_And_Fins_Set",
"IBM_Facsimile_Machine", "Sb_Ib_Bag_Flour",
"Peace_Lined_Roof_Gloves", "Bermuda_Grass_Seeds",
"Front_And_Rear_Floor_Mat", "Waiter_Costume",
"Magical_Mystery_Maze", "Red_Safety_Players",
"Silver_Push_Pins", "Quartz_Digital_Wristwatch",
"10_Lb_Test_Fishing_Line", "Blue_Pile_Carpeting",
"Red_Brick_House", "Bear_Claw_Doughnuts",
"AS/400_Advanced_Series", "12_Inch_Wooden_Ruler",
"Square_Plastic_Calendar", "Twelve_Num_Two_Pencils",
"Red_Yellow_Legal_Paper", "Change_Machine",
"Street_Hockey_Balls", "Half_Inch_3_Ring_Binders",
"Sporting_Good_Catalog", "Ten_Gallon_Hats",
"Promotion_Notification", "32_Gal_Keq_Beer",
"Business_Cards", "Bebble_Beach_Poster",
"Texas_Residence_Forms", "COBOL_Programmers_Guide",
"25_Inch_Color_TV", "Black_Magic_Marker",
"Telephone_Books", "Riding_Lawnmower",
"$7500_Gift_Certificate", "Compact_Disc_Player",
"Four_Drawer_Cabinets", "Dry-Erase_Markers",
"Multi-System_Paper", "Message_Waiting_Light",
"Automotive_Care_Manuals", "Carton_Ceiling_Tiles",
"12_Ounce_Styrofoam_Cup", "Phone_Calling_Card",
"Inflatable_Globe", "Walnut_Magazine_Rack",
"Over_Under_Shotgun", "While_You_Were_Gone_Memo",
"Three_Tea_Bags", "Bowling_Pin_Spotter",
"Golf_Club_Cleaner", "Staple_Removers",
"Downtown_Parking_Permit", "Sliding_Glass_Door_Locks",
"Floral_Pattern_Lampshade", "Junior_College_Books",
"Two_Wheel_Bike", "Zoo_Season_Pass",
"Baseball_Tickets", "Four_Person_Tent",
"Cross_Country_Ski_Set", "Motorcycle_Sidecars",
"Fireplace_Tools", "Radio_Controlled_Plane",
"Oak_Wall_Clock", "Rubber_Baby_Buggy_Wheel",
"Anti_Static_Strap", "Old_Wooden_Toothpicks"
};
/*-----*/
/* Function: main */
/*-----*/
int main(int argc, char* argv)
{
int i;
tpccc_ITRCD_both_t ItemRec;

```

```

char dbfile[11];
char dlib[11];
char cmd[100];

ExcData_t volatile exD = {0, "HANDLE "};

/* Parse the input parms */
dlib[10] = '\0';
_CPYBYTES(dlib, argv[DBLIB_PARM], 10);
dlib[triml(dlib, ' ')] = '\0';

dbfile[10] = '\0';
_CPYBYTES(dbfile, argv[DBFILE_PARM], 10);
dbfile[triml(dbfile, ' ')] = '\0';

/* TPC message queue to send our build messages to. */
_CPYBYTES(msgq_gl, "TPCCMSG ", 10);
_CPYBYTES(&msgq_gl[10], argv[DBLIB_PARM], 10);

sprintf(msgtxt, "FILLITEM program filling the %s db file...", dbfile);
Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, STATUS_MSG, CS_EXT_MSGQ, 1);

/* Do the overrides */
sprintf(cmd,
        "QVRDBF FILE(%s) TOPFILE(%s) SEQONLY(*YES 15000) OVRSCOPE(*JOB)",
        dbfile, dlib, dbfile);

#pragma exception_handler(0excepHdlr, exD, 0, _C2_MH_ESCAPE)
QCMDXCC(cmd, strlen(cmd));
#pragma disable_handler

if (0 != exD.error)
{
    sprintf(msgtxt, "QVRDBF command in FILLITEM program failed!");
    Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
    return(-1);
}

if (NULL == (Item_File = _Ropen(dbfile, "r+")))
{
    sprintf(msgtxt, "Open of file, %s, failed!", dbfile);
    Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
    return(-1);
}

for (i=1; i <= MAXITEMS; i++)
{
    ItemRec.IID = i;
    ItemRec.IMAGEID = RANDOM(1,10000);

    _CPYBYTES(ItemRec.INAME, ItemNameStr[RANDOM(0,99)], 24);
    ItemRec.IPRICE = (decimal(5,2)) ((RANDOM(100,10000)) / 100.00);

    /* Build IDATA random string 26 to 50 chars */
    CreateStrPad_extra(26, 50, ItemRec.IDATA, "ORIGINAL");

    _Rwrite(Item_File, (void *)&ItemRec, sizeof(ItemRec));
} /* end of for */

_Rclose(Item_File);

sprintf(msgtxt, "FILLITEM completed filling the %s file in library %s",
        dbfile, dlib);
Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, COMP_MSG, msgq_gl, 1, NULL);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, COMP_MSG, CS_CTLBDY, 1);
} /* end of main */

```

## C.30 FILLORD.C:

```

/*-----*/
/*
/* File name: FILLORD
/* Purpose: This program fills the neword, orders, and ordlin files.
/* It will create an order record, selecting the
/* customer for that order at random (each customer will
/* one and only one order but which order that will be is
/* determined with random numbers).
/* The next thing done is only done for orders between 2,101 and
/* 3000 for each district, this is the creation of a neworder
/* record. The orderline records for that order are created last,
/* the number of lines for the order is between 5 and 15, selected
/* at random.
/*-----*/

/*-----*/
/* Includes.
/*-----*/
#include <stdlib.h>
#include <errno.h>
#include <rcio.h>
#include <string.h>
#include <xfidbk.h>
#include <mlib.h>
#include <micomput.h>
#include <decimal.h>
#include <time.h>
#include <xcvct.h>
#include <except.h>
#include <QSYSINC/MIH/MATTOD> /* time of day ml time */
#include <QSYSINC/MIH/CMPSWF>
#include <QSYSINC/MIH/WAITIME>
#include <qcmdexc.h>
#include <nh/triml.h>
#include <filldata.h>
#include <tpccspace.h>
#include <commonptcc.h>
#include <os4msg.h>

/* MAPINC generates structures for the database files
#pragma mapinc("newordpf","lib1/newordpf(*all)","both key", "d_p","tpcc")
#include "newordpf"
#pragma mapinc("orderspf","lib1/orderspf(*all)","both key", "d_p","tpcc")
#include "orderspf"
#pragma mapinc("ordlinpf","lib1/ordlinpf(*all)","both key", "d_p","tpcc")
#include "ordlinpf"

/*-----*/
/* Defines
/*-----*/
#define ACTIVE_WAREHOUSES 100
#define MAX_WHRS_SIZE 15000

#define STRWH_PARM 1
#define NUMWH_PARM 2
#define INST_VALUE_PARM 3
#define JOB_INDEX_PARM 4
#define DBLIB_PARM 5
#define PARM_ORDERS_FILE 6
#define PARM_ORDLIN_FILE 7
#define PARM_NEWORD_FILE 8

```

```

/*-----*/
/* globals
/*-----*/
int strWH_gl; /* Starting warehouse from parameter input. */
short customer[ACTIVE_WAREHOUSES][10][3000]; /* array of customer numbers */
short offset_table[MAX_WHRS_SIZE]; /* table of offsets for warehouses. */
char offset_active = 'N';

_FILE *NewOrder_File, *Orders_File, *OrderLine_File;

tpcc_ORRCD_both_t OrderRec;
tpcc_OLRCD_both_t OrdlnRec;
tpcc_NORCD_both_t NewordRec;

static MI_Time CurrTime;
static Char CarrStr[3];
static unsigned long int rand_next = 1;

char msgtxt[133];
char msgq_gl[20];

/* There are 10 entries of length 24 each */
char "DIST INFO[]" =
{
    "Info for District Num 00",
    "Info for District Num 01",
    "Info for District Num 02",
    "Info for District Num 03",
    "Info for District Num 04",
    "Info for District Num 05",
    "Info for District Num 06",
    "Info for District Num 07",
    "Info for District Num 08",
    "Info for District Num 09",
    "Info for District Num 10"
};

char *CarId[] = {"00", "01", "02", "03", "04", "05",
                "06", "07", "08", "09", "10" };

/*-----*/
/* L o a d O r d e r s
/* Create & write orders and Orderline rec.
/*
/* P A R M S: curWH => current warehouse id
/* curDist => current district id
/* curOrd => current Order id
/*-----*/
void LoadOrders(int curWH, int curDist, int curOrd)
{
    decimal(7,2) d;
    short int oln,curOln;
    char StockDist[25];

    int warehouse_index; /* index into offset table.
    short order_pOint; /* index of the order number point.
    int warehouse_point; /* actual index into the table.

    OrderRec.OID = OrdlnRec.OLOID = curOrd;
    OrderRec.OUID = OrdlnRec.OLUID = curWH;
    OrderRec.ODID = OrdlnRec.OLDID = curDist;

    /* Generate a random customer id.
    warehouse_index = curWH + strWH_gl;
    order_pOint = (offset_table[warehouse_index] + curOrd - 1) % ORDERSPERDIST;
    /* Set index into table. Use table size.
    warehouse_point = warehouse_index + ACTIVE_WAREHOUSES;
    OrderRec.OCID = customer[warehouse_point][curDist - 1][order_pOint];

    MattOD(CurrTime);
    CurrTime[7] = BASE TIME;
    memcpy(OrderRec.OENTTOD, CurrTime, sizeof(MI_Time));

    if (curOrd < 2101)
    {
        strncpy(OrderRec.OCARID, CarId[RANDOM(1, 10)],strlen(CarId[0]));
    } else {
        memset(OrderRec.OCARID, '\0', strlen(OrderRec.OCARID));
    }
    OrderRec.OLINES = RANDOM(5, 15);
    curOln = OrderRec.OLINES;
    OrderRec.OLCALL = 1;
    _Rwrite(Orders_File, (void *)&OrderRec, sizeof(OrderRec));

    /* load order line records
    for (oln = 1; oln <= curOln; ++oln)
    {
        OrdlnRec.OLOID = curOrd;
        OrdlnRec.OLNBR = oln;
        while ((OrdlnRec.OLIID = MaxRand(&rand_next)) == 0);
        OrdlnRec.OLSPWH = curWH;
        OrdlnRec.OLQTY = 5;
        if (curOrd < 2101)
        {
            memcpy(OrdlnRec.OLDLVTOD, OrderRec.OENTTOD, sizeof(OrderRec.OENTTOD));
            OrdlnRec.OLAMNT = 0.00;
        } else {
            memcpy(OrdlnRec.OLDLVTOD, "00000000", 8);
            d = MaxRand(&rand_next) / 100;
            OrdlnRec.OLAMNT = d;
        }
        memcpy(StockDist, DistInfl[RANDOM(0, 9)], 11);
        memcpy((void *)&StockDist[11], DistInF2[curDist-1],2);
        memcpy((void *)&StockDist[13], DistInF3[RANDOM(0, 9)],11);
        strncpy(OrdlnRec.OLDSTL, StockDist, 24);

        _Rwrite(OrderLine_File, (void *)&OrdlnRec, sizeof(OrdlnRec));
    } /* End of order line loop */

    return;
} /* End of LoadOrders */

/*-----*/
/*
/* Function: load_customer
/*
/* Build the array for OCIDs.
/* Generates a unique random customer id and builds an array.
/*-----*/
void load_customer(int EndWH)
{
    int i,j,k,l, temp;

    if (EndWH > ACTIVE_WAREHOUSES)
    {
        for (i=0; i < EndWH; ++i)
        {
            /* Load the offset table with the starting point for customer.
            offset_table[i] = RANDOM(0, (ORDERSPERDIST-1));
        }
        EndWH = ACTIVE_WAREHOUSES;
        offset_active = 'Y'; /* Set the flag if required.
    } else {
        for (i=0; i < EndWH; ++i)
        {
            /* load the offset table */
            offset_table[i] = 0; /* set the starting point for customer */
        }
        offset_active = 'N';
    }

    for (i=0; i < EndWH; ++i) /* go through the warehouse numbers */

```

```

{
  /* set up the array for storing the order number */
  for (j=0; j < 10; ++j)
  {
    for (k=0; k < ORDERSPERDIST; ++k)
    {
      customer[i][j][k] = k+1; /* setup the values */
    } /* end of initialize customer id. */

    for (k=0; k < ORDERSPERDIST; ++k)
    {
      /* Get the swap point. */
      RANDOM2(k, (ORDERSPERDIST-1), 1)
      temp = customer[i][j][k]; /* get the value to be swapped */
      customer[i][j][k] = customer[i][j][temp]; /* swap part one */
      customer[i][j][temp] = temp; /* swap part two */
    }
  }
}
return;
} /* End of load_customer */

/*-----*/
/*
/* Function: load_files
/*
/*-----*/
int load_files(int EndWH, int CurOld)
{
  short int wid, dist;

  for (wid = strWH_gl; wid < EndWH; ++wid)
  {
    for (dist = 1; dist <= 10; ++dist)
    {
      LoadOrders(wid, dist, CurOld);

      if (CurOld >= 2101)
      {
        /* Load New Order record. */
        NewOrderRec.NOID = wid;
        NewOrderRec.NOID = dist;
        NewOrderRec.NOID = CurOld;
        _Rwrite(NewOrder_File, (void *) &NewOrderRec, sizeof(NewOrderRec));
      }
    }
  }
  return(0);
} /* end of load_files */

/*-----*/
/*
/* Function: main
/*
/*-----*/
int main(int argc, char ** argv)
{
  unsigned int lib_len;
  char *slash_pos;

  char pgmname[11];
  char pgmlib[11];
  char job_index[3];
  char job_name[11] = " ";
  char cmd[120]; /* override commands */
  char dblib_null[11];
  char dblib_space[11];
  char orders_file[11];
  char ordlin_file[11];
  char neword_file[11];
  int loopX;
  int inst_num = 0; /* the space instance number */
  int ordlin_ovr_size; /* override size for order line file. */
  int orders_ovr_size; /* override size for orders file. */
  int neword_ovr_size; /* override size for new orders file. */
  int numWH;
  unsigned long int this_seed;
  ExcData_t volatile exd = {0, "HANDLE "};

  /*-----*/
  /* Wait time variables (key think time waits)
  /*-----*/
  _MI_Time time_to_wait; /* The amount of time to wait */
  int hours = 0; /* Time components used to */
  minutes = 0; /* create an _MI_Time value */
  seconds = 0; /* that can be passed to the */
  hundredths = 0; /* 'waittime' function */
  short wait_option = _WAIT_NORMAL; /* wait time options */
  /* format an mtime value */
  /* representing the wait time */

  /* Parse the input parms */
  _CPYBYTES(dblib_space, argv[DBLIB_PARAM], 10);
  _CPYBYTES(dblib_null, argv[DBLIB_PARAM], 10);
  dblib_null[10] = '\0';
  dblib_null[triml(dblib_null, ' ')] = '\0';
  dblib_space[10] = '\0';

  /* TPC message queue to send our build messages to. */
  _CPYBYTES(msgq_gl, "TPCCMSG ", 10);
  _CPYBYTES(&msgq_gl[10], argv[DBLIB_PARAM], 10);

  strWH_gl = atoi(argv[STRWH_PARAM]);
  numWH = atoi(argv[NUMWH_PARAM]);
  numWH = strWH_gl + numWH;

  /*-----*/
  /* Get the library name that we are running in now.
  /*-----*/
  slash_pos = strchr(argv[0], '/');
  lib_len = slash_pos - argv[0];
  _CPYBYTES(pgmlib, argv[0], lib_len);
  pgmlib[lib_len] = '\0';

  ++slash_pos; /* Skip the '/' */
  _CPYBYTES(pgmname, slash_pos, 10);
  pgmname[triml(pgmname, ' ')] = '\0';

  job_index[3] = '\0';
  _CPYBYTES(job_index, argv[JOB_INDEX_PARAM], 3);
  job_index[triml(job_index, ' ')] = '\0';
  sprintf(job_name, "%0.7s%03s", pgmname, job_index);

  strncpy(orders_file, argv[PARAM_ORDERS_FILE], 11);
  strncpy(ordlin_file, argv[PARAM_ORDLIN_FILE], 11);
  strncpy(neword_file, argv[PARAM_NEWORD_FILE], 11);

  inst_num = atoi(argv[INST_VALUE_PARAM]);

  /*-----*/
  /* Send a STARTING message...
  /*-----*/
  while (0 != add_started_jobs(dblib_space, ORDER_JOBS, inst_num))
  {
    sprintf(msgtxt, "Error when calling add_started_jobs(0).");
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
    return(-1);
  }

  printf(msgtxt,
    "%s is starting to fill the ORDERS files from warehouse %d to %d.",
    job_name, strWH_gl, (numWH-1));
  QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
  QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, STATUS_MSG, CS_EXT_MSGQ, 1);

  ordlin_ovr_size = IO_BLOCK_SIZE / sizeof(tpcc_ORLDCD_both_t);
  neword_ovr_size = IO_BLOCK_SIZE / sizeof(tpcc_ORRCD_both_t);
  neword_ovr_size = IO_BLOCK_SIZE / sizeof(tpcc_NORCD_both_t);

  sprintf(cmd, "OVRDBF FILE(%s) TOFILE(%s/%s) SEQONLY(*YES %d) OVRSCOPE(*JOB)",
    neword_file, dblib_null, neword_file, neword_ovr_size);
  #pragma exception_handler(0excepHdlr, exd, 0, _C2_MH_ESCAPE)
  QCMDEXC(cmd, strlen(cmd));
  #pragma disable_handler
  if (0 != exd.error)
  {
    sprintf(msgtxt, "Override of Newordpf failed");
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
    return(-1);
  }

  sprintf(cmd, "OVRDBF FILE(%s) TOFILE(%s/%s) SEQONLY(*YES %d) OVRSCOPE(*JOB)",
    orders_file, dblib_null, orders_file, orders_ovr_size);
  #pragma exception_handler(0excepHdlr, exd, 0, _C2_MH_ESCAPE)
  QCMDEXC(cmd, strlen(cmd));
  #pragma disable_handler
  if (0 != exd.error)
  {
    sprintf(msgtxt, "Override of ORDERSPF failed.");
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
    return(-1);
  }

  sprintf(cmd, "OVRDBF FILE(%s) TOFILE(%s/%s) SEQONLY(*YES %d) OVRSCOPE(*JOB)",
    ordlin_file, dblib_null, ordlin_file, ordlin_ovr_size);
  #pragma exception_handler(0excepHdlr, exd, 0, _C2_MH_ESCAPE)
  QCMDEXC(cmd, strlen(cmd));
  #pragma disable_handler
  if (0 != exd.error)
  {
    sprintf(msgtxt, "Override of ORDLINPF failed.");
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
    return(-1);
  }

  if ((Orders_File = _Ropen(orders_file,
    "ar,arseq=Y,blkrcd=Y,riofb=N")) == NULL)
  {
    sprintf(msgtxt, "Open of ORDERSPF file failed.");
    add_error_jobs(dblib_space, ORDER_JOBS, inst_num);
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
    return(-1);
  }

  if ((OrderLine_File = _Ropen(ordlin_file,
    "ar,arseq=Y,blkrcd=Y,riofb=N")) == NULL)
  {
    sprintf(msgtxt, "Open of ORDLIN file %s failed.", ordlin_file);
    add_error_jobs(dblib_space, ORDER_JOBS, inst_num);
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
    return(-1);
  }

  if ((NewOrder_File = _Ropen(neword_file,
    "ar,arseq=Y,blkrcd=Y,riofb=N")) == NULL)
  {
    sprintf(msgtxt, "Open file %s failed\n", neword_file);
    add_error_jobs(dblib_space, ORDER_JOBS, inst_num);
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
    return(-1);
  }

  /* Set up random number generator */
  this_seed = get_seed(); /* get the seed value */
  srand(this_seed); /* set the seed */
  rand_next = (unsigned long int) this_seed;

  load_customer(numWH);

  for (loopX = 1; loopX <= ORDERSPERDIST; ++loopX)
  {
    load_files(numWH, loopX);
  }

  _Rclose(Orders_File);
  _Rclose(OrderLine_File);
  _Rclose(NewOrder_File);
  add_successful_jobs(dblib_space, ORDER_JOBS, inst_num);

  /*-----*/
  /* Send an ENDING message...
  /*-----*/
  sprintf(msgtxt, "Job has completed successfully.", job_name);
  QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, COMP_MSG, msgq_gl, 1, NULL);
  QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, STATUS_MSG, CS_EXT_MSGQ, 1);

  return(0);
} /* end of main */

```

## C.31 FILLPARTS.CL:

```

/*-----*/
/*
/* Program: FILLPARTS
/*
/* Purpose: Fill the files that make up a Split database.
/*
/* using any number of partitions 1 to 32
/*
/* Params: Where to put pgm objects
/*
/* Database library name
/*
/* Number of warehouses
/*
/* Build files and index mixed
/*
/*-----*/
FCM PARAM(dBLIB dBRASP nNUMWHs nNUMPARTS tTEMPSTG)

DCL dBLIB TYPE(*CHAR) LEN(10)
DCL dBRASPCN TYPE(*CHAR) LEN(3) /* DB ASP number w/null */
DCL dBRASPC TYPE(*CHAR) LEN(2) /* DB ASP number w/null */
DCL dBRASP TYPE(*DEC) LEN(2 0) /* DB ASP number */
DCL nNUMWHs TYPE(*DEC) LEN(6 0)
DCL nNUMWHSC TYPE(*CHAR) LEN(6)
DCL tTEMPSTG TYPE(*CHAR) LEN(7)

DCL nNUMPARTS TYPE(*DEC) LEN(2 0)
DCL nNUMPARTSC TYPE(*CHAR) LEN(2)

DCL sTRWH TYPE(*DEC) LEN(6 0) VALUE(1)
DCL nNEXTSTR TYPE(*DEC) LEN(6 0)
DCL nPARTSIZE TYPE(*DEC) LEN(6 0)
DCL sLASTSIZE TYPE(*DEC) LEN(6 0)

DCL sTRWHC TYPE(*CHAR) LEN(6)
DCL nNUMWHSCN TYPE(*CHAR) LEN(6)
DCL nNUMWHSCN TYPE(*CHAR) LEN(7)

DCL dLVVRSYZ TYPE(*DEC) LEN(10 0)
DCL nPMLIB TYPE(*CHAR) LEN(10)
DCL sMSGQ TYPE(*CHAR) LEN(10) VALUE(INDEXMSGQ)

```

```

DCL &GENMSGQ TYPE(*CHAR) LEN(10) VALUE(TPCMSG)
DCL &RTNTYPE TYPE(*CHAR) LEN(2)
DCL &FUNCTION TYPE(*CHAR) LEN(3)
DCL &NULL TYPE(*CHAR) LEN(1) VALUE(X'00')
DCL &RECMFFLAG TYPE(*CHAR) LEN(1) VALUE(Y)
DCL &TAG TYPE(*CHAR) LEN(2)
DCL &CURRPART TYPE(*DEC) LEN(2)
/*-----*/
/* File name variables one per file */
/*-----*/
DCL &CSTMNAN TYPE(*CHAR) LEN(10)
DCL &DSTRCTNAM TYPE(*CHAR) LEN(10)
DCL &HSTRYNAM TYPE(*CHAR) LEN(10)
DCL &NEWORINAM TYPE(*CHAR) LEN(10)
DCL &ORDLINNAM TYPE(*CHAR) LEN(10)
DCL &ORDNAM TYPE(*CHAR) LEN(10)
DCL &STOCKNAM TYPE(*CHAR) LEN(10)
DCL &WRHSNAM TYPE(*CHAR) LEN(10)

MONMSG MSGID(CPF0000 MCH0000 CZM0000) EXEC(GOTO ERREXIT)

/* Other pgms will be called from library we are running in. */
IF COND(&NUMPARTS <= &NUMWHS) THEN(+
DO)
  SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*ESCAPE) +
  MSGDTA('Number of warehouses must be greater or equal +
  to the number of partitions.')
  GOTO ERREXIT
ENDDO
RTVJOBID OBJ(FILLPARTS) OBJTYPE(*PGM) RTNLIB(&PGLIB)

CD DIR(/) /* go to the base directory */
MKDIR DIR(&PGLIB) /* Create the directory needed for spawning */
MONMSG CPF0A0 EXEC(RCVMSG RMV(*YES))

ADDLBLE &DLBL
MONMSG CPF2103 EXEC(RCVMSG RMV(*YES))

CALL TPCTOOLS/SETTOOLBL
CALL &PGLIB/SETBLDLBL

/*-----*/
SNDPGMMSG MSGID(CPF9897) +
  MSGDTA('FILLPARTS: creating initial DB files...') +
  MSGF(QSYS/QCPFMMSG) TOPGMQ(*EXT) MSGTYPE(*STATUS)

CRTFF FILE(&DLBL/AREFFIL)
MONMSG MSGID(CPF2103 CPF7302) EXEC(RCVMSG RMV(*YES))

CHGVAR &STRWHC VALUE(&STRWH)
CHGVAR &NUMWHS VALUE(&NUMWHS)
CHGVAR &NUMWHSCN VALUE(&NUMWHSCN *CAT &NULL)

CRTBLDSPCE DLBL(&DLBL) STRWH(&STRWH) WHS(&NUMWHSC) SPLITDB(Y)

CHGVAR &PARTSIZE VALUE(&NUMWHS / &NUMPARTS)
CHGVAR &LASTSTR VALUE(&NUMWHS - ((&NUMPARTS - 1) * &PARTSIZE))
CHGVAR &NEXTSTR VALUE(&STRWH + &PARTSIZE)

/* MSGQ for Send and Receive messages for index builds */
CRTMSGQ MSGQ(&DLBL/4MSGQ)
MONMSG CPF2112 EXEC(CLRMSGQ MSGQ(&DLBL/4MSGQ))
/*-----*/
/* Fill the ORDERS, NEWORDS, ORDLIN files. */
/*-----*/

IF COND(&NUMPARTS = 1) THEN(+
CRTORDFILE DLBL(&DLBL) ORDNAM(ORDERSPF) +
  ORDLINNAM(ORDLINPF) NEWORINNAM(NEWORDPF) +
  STRWH(&STRWH) WHS(&NUMWHS) RECOMPFILE(Y))
ELSE
  CMD(+
DO)
  CHGVAR VAR(&CURRPART) VALUE(1)
  ORDLLOOP: CHGVAR VAR(&TAG) VALUE(&CURRPART)
  CHGVAR VAR(&ORDNAM) VALUE('ORDERSPF' *TCAT &TAG)
  CHGVAR VAR(&ORDLINNAM) VALUE('ORDLINPF' *TCAT &TAG)
  CHGVAR VAR(&NEWORINNAM) VALUE('NEWORDPF' *TCAT &TAG)
  IF (&CURRPART <= &NUMPARTS) THEN(+
  CRTORDFILE DLBL(&DLBL) ORDNAM(&ORDNAM) +
  ORDLINNAM(&ORDLINNAM) NEWORINNAM(&NEWORINNAM) +
  STRWH(&STRWH) WHS(&PARTSIZE) RECOMPFILE(&RECMFFLAG))
  ELSE CMD(+
  CRTORDFILE DLBL(&DLBL) ORDNAM(&ORDNAM) +
  ORDLINNAM(&ORDLINNAM) NEWORINNAM(&NEWORINNAM) +
  STRWH(&STRWH) WHS(&LASTSIZE) RECOMPFILE(&RECMFFLAG))
  CHGVAR VAR(&RECMFFLAG) VALUE('N')
  CHGVAR VAR(&STRWH) VALUE(&STRWH + &PARTSIZE)
  CHGVAR VAR(&CURRPART) VALUE(&CURRPART + 1)
  IF COND(&CURRPART <= &NUMPARTS) THEN(GOTO +
  CMDLBL(ORDLOOP))
ENDDO

CHGVAR VAR(&NUMPARTS) VALUE(&NUMPARTS)
SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
  MSGDTA('INDEXBUILD: submitting jobs to build indexes...') +
  TOPGMQ(*EXT) MSGTYPE(*STATUS)

CHGVAR VAR(&DBASPC) VALUE(&DBASPC)
CHGVAR VAR(&NUMWHSC) VALUE(&NUMWHSC)
SBMJOB CMD(CRTDBIX IX(*ORDLIN) DLBL(&DLBL) DBASPC(&DBASPC) +
  TEMPSTG(&TEMPSTG) NUMP(&NUMPARTS) WHS(&NUMWHSC)) +
  JOB(ORDLININX) JOBQ(&PGLIB/TPCBLDJOBQ) MSGQ(&DLBL/4MSGQ)

/*-----*/
/* Fill the STOCK and customer files. */
/*-----*/
CHGVAR VAR(&STRWH) VALUE(1)
CHGVAR VAR(&RECMFFLAG) VALUE('Y')
CRTFF FILE(&DLBL/STOCKPF) SRCMBR(STOCKPF) +
  SIZE(*NOMAX) WAITRCD(2) SHARE(*NO) +
  REUSEDLT(*NO) LVLCHK(*NO)

CRTFF &DLBL/CSTMTRPF SRCMBR(CSTMTRPF) SIZE(*NOMAX) +
  WAITRCD(2) SHARE(*NO) REUSEDLT(*NO) LVLCHK(*NO)

CRTCMD &PGLIB/MASTCUSSTK
CRTPGM PGM(&PGLIB/MASTCUSSTK) +
  MODULE(&PGLIB/MASTCUSSTK +
  &PGLIB/COMMONTPCC &PGLIB/TPCCSPACE) +
  ENTMOD(&PGLIB/MASTCUSSTK) +
  BNDSRVPGM(TPCTOOLS/TPCCTOOLS QSYS/QDBRUNHA)

IF COND(&NUMPARTS <= 1) THEN(+
DO)
  DLTF FILE(STOCKPF)
  DLTF FILE(CSTMTRPF)
  /* CREATE THE PHYSICAL FILES */
  CHGVAR VAR(&CURRPART) VALUE(1)
  STOCKLOOP: CHGVAR VAR(&TAG) VALUE(&CURRPART)
  CHGVAR VAR(&STOCKNAM) VALUE('STOCKPF' *TCAT &TAG)
  CHGVAR VAR(&CSTMTRNAM) VALUE('CSTMTRPF' *TCAT &TAG)
  CRTFF FILE(&DLBL/STOCKNAM) SRCMBR(STOCKPF) +
  SIZE(*NOMAX) WAITRCD(2) SHARE(*NO) +
  REUSEDLT(*NO) LVLCHK(*NO)

  CRTFF &DLBL/CSTMTRNAM SRCMBR(CSTMTRPF) SIZE(*NOMAX) +
  WAITRCD(2) SHARE(*NO) REUSEDLT(*NO) LVLCHK(*NO)
  CHGVAR VAR(&CURRPART) VALUE(&CURRPART + 1)
ENDDO

IF COND(&CURRPART <= &NUMPARTS) THEN(GOTO +
  CMDLBL(STOCKLOOP))

CHGVAR VAR(&STRWH) VALUE(1)
CHGVAR VAR(&RECMFFLAG) VALUE('Y')
IF COND(&NUMPARTS = 1) THEN(+
  CRTWHFILE DLBL(&DLBL) FILE(DSTRCT) STRWH(&STRWH) +
  WHS(&NUMWHS) RECOMPFILE(Y))
ELSE
  CMD(+
DO)
  CHGVAR VAR(&CURRPART) VALUE(1)
  WHRHSLOOP: CHGVAR VAR(&TAG) VALUE(&CURRPART)
  CHGVAR VAR(&WRHSNAM) VALUE('WRHS' *TCAT &TAG)
  IF (&CURRPART <= &NUMPARTS) THEN(+
  CRTWHFILE DLBL(&DLBL) FILE(&WRHSNAM) STRWH(&STRWH) +
  WHS(&PARTSIZE) RECOMPFILE(&RECMFFLAG))
  ELSE CMD(+
  CRTWHFILE DLBL(&DLBL) FILE(&WRHSNAM) STRWH(&STRWH) +
  WHS(&LASTSIZE) RECOMPFILE(&RECMFFLAG))
  CHGVAR VAR(&RECMFFLAG) VALUE('N')
  CHGVAR VAR(&STRWH) VALUE(&STRWH + &PARTSIZE)
  CHGVAR VAR(&CURRPART) VALUE(&CURRPART + 1)
  IF COND(&CURRPART <= &NUMPARTS) THEN(GOTO +
  CMDLBL(WRHSLOOP))
ENDDO

/*-----*/
/* Create the DISTRICT part 1 file and programs. */
/*-----*/
CHGVAR VAR(&STRWH) VALUE(1)
CHGVAR VAR(&RECMFFLAG) VALUE('Y')
IF COND(&NUMPARTS = 1) THEN(+
  CRTDIST DLBL(&DLBL) FILE(DSTRCT) STRWH(&STRWH) +
  WHS(&NUMWHS) RECOMPFILE(Y))
ELSE
  CMD(+
DO)
  CHGVAR VAR(&CURRPART) VALUE(1)
  DISTLOOP: CHGVAR VAR(&TAG) VALUE(&CURRPART)
  CHGVAR VAR(&DSTRCTNAM) VALUE('DSTRCT' *TCAT &TAG)
  IF (&CURRPART <= &NUMPARTS) THEN(+
  CRTDIST DLBL(&DLBL) FILE(&DSTRCTNAM) STRWH(&STRWH) +
  WHS(&PARTSIZE) RECOMPFILE(&RECMFFLAG))
  ELSE CMD(+
  CRTDIST DLBL(&DLBL) FILE(&DSTRCTNAM) STRWH(&STRWH) +
  WHS(&LASTSIZE) RECOMPFILE(&RECMFFLAG))
  CHGVAR VAR(&RECMFFLAG) VALUE('N')
  CHGVAR VAR(&STRWH) VALUE(&STRWH + &PARTSIZE)
ENDDO

```

```

CHGVAR VAR (&CURRPART) VALUE (&CURRPART + 1)
IF COND (&CURRPART = 1E &NUMPARTS) THEN (GOTO +
CMDLBL (DISTLOOP))
ENDDO
/*-----*/
/* Create delivery log and delivery log a */
/*-----*/
CHGVAR VAR (&DLVRSZ) VALUE (&NUMHWS + 1000)
CRTPF FILE (&DLIB/OLVRYLOG) SIZE (&DLVRSZ 10000 10000) +
ALLOCATE (*YES) WAITRCD (*NOMAX) SHARR (*YES) LVLCHK (*NO)
CRTPF FILE (&DLIB/OLVRYLOGA) SIZE (*NOMAX) WAITRCD (*NOMAX) +
SHARR (*YES) LVLCHK (*NO)
/*-----*/
CRTPTF FILE (&DLIB/DBGUPRT) LVLCHK (*NO)
CRTPTF FILE (&DLIB/DBGUPRT2) LVLCHK (*NO)
/*-----*/
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) +
MSGDTA ('INDEXBUILD: waiting for ordline index to complete...') +
TOPGMQ (*EXT) MSGTYPE (*STATUS)
RCVMMSG MSGQ (&DLIB/AMSGQ) MSCSTYPE (*ANY) WAIT (*MAX) RTNTYPE (&RTNTYPE)
IF COND ((&RTNTYPE = '15') | (&RTNTYPE = '17')) THEN (DO)
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*DIAG) +
MSGDTA ('Error with submitted INDEXBUILD job. See appropriate joblog!')
GOTO ERREXIT
ENDDO
/*-----*/
/* Build ORDERS and NEWORD indexes */
/*-----*/
SBMJOB CMD (CRTDBIX IX (*ORDERS) DBLIB (&DLIB) DBASP (&DBASPC) +
TEMPSTG (&TEMPSTG) NUMP (&NUMPARTS) WHS (&NUMHSC) +
JOB (ORDERSINX) JOBO (&PGLIB/TPCBLDJOBQ) MSQ (&DLIB/AMSGQ)
SBMJOB CMD (CRTDBIX IX (*NEWORD) DBLIB (&DLIB) DBASP (&DBASPC) +
TEMPSTG (&TEMPSTG) NUMP (&NUMPARTS) WHS (&NUMHSC) +
JOB (NEWORDINX) JOBO (&PGLIB/TPCBLDJOBQ) MSQ (&DLIB/AMSGQ)
/*-----*/
/* WAIT FOR COMPLETION OF SUBMITTED JOBS. */
/* Send a status msg to the screen and also one to the joblog... */
/*-----*/
IF COND (&NUMPARTS *GT 1) THEN (DO)
CALL CRTINDEX PARM ('HSTRY ' &DLIB &DBASPC +
&NUMHSCN &NUMPARTS &TEMPSTG)
CALL CRTINDEX PARM ('WRHS ' &DLIB &DBASPC +
&NUMHSCN &NUMPARTS &TEMPSTG)
CALL CRTINDEX PARM ('DSTRCT ' &DLIB &DBASPC +
&NUMHSCN &NUMPARTS &TEMPSTG)
ENDDO
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) +
MSGDTA ('INDEXBUILD: waiting for 5 submitted jobs to complete...') +
TOPGMQ (*EXT) MSGTYPE (*STATUS)
RCVMMSG MSGQ (&DLIB/AMSGQ) MSCSTYPE (*ANY) WAIT (*MAX) RTNTYPE (&RTNTYPE)
IF COND ((&RTNTYPE = '15') | (&RTNTYPE = '17')) THEN (DO)
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*DIAG) +
MSGDTA ('Error with submitted INDEXBUILD job. See appropriate joblog!')
GOTO ERREXIT
ENDDO
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) +
MSGDTA ('INDEXBUILD: waiting for 4 submitted jobs to complete...') +
TOPGMQ (*EXT) MSGTYPE (*STATUS)
RCVMMSG MSGQ (&DLIB/AMSGQ) MSCSTYPE (*ANY) WAIT (*MAX) RTNTYPE (&RTNTYPE)
IF COND ((&RTNTYPE = '15') | (&RTNTYPE = '17')) THEN (DO)
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*DIAG) +
MSGDTA ('Error with submitted INDEXBUILD job. See appropriate joblog!')
GOTO ERREXIT
ENDDO
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) +
MSGDTA ('INDEXBUILD: waiting for 3 submitted jobs to complete...') +
TOPGMQ (*EXT) MSGTYPE (*STATUS)
RCVMMSG MSGQ (&DLIB/AMSGQ) MSCSTYPE (*ANY) WAIT (*MAX) RTNTYPE (&RTNTYPE)
IF COND ((&RTNTYPE = '15') | (&RTNTYPE = '17')) THEN (DO)
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*DIAG) +
MSGDTA ('Error with submitted INDEXBUILD job. See appropriate joblog!')
GOTO ERREXIT
ENDDO
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) +
MSGDTA ('INDEXBUILD: waiting for 2 submitted jobs to complete...') +
TOPGMQ (*EXT) MSGTYPE (*STATUS)
RCVMMSG MSGQ (&DLIB/AMSGQ) MSCSTYPE (*ANY) WAIT (*MAX) RTNTYPE (&RTNTYPE)
IF COND ((&RTNTYPE = '15') | (&RTNTYPE = '17')) THEN (DO)
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*DIAG) +
MSGDTA ('Error with submitted INDEXBUILD job. See appropriate joblog!')
GOTO ERREXIT
ENDDO
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) +
MSGDTA ('INDEXBUILD: waiting for 1 submitted jobs to complete...') +
TOPGMQ (*EXT) MSGTYPE (*STATUS)
RCVMMSG MSGQ (&DLIB/AMSGQ) MSCSTYPE (*ANY) WAIT (*MAX) RTNTYPE (&RTNTYPE)
IF COND ((&RTNTYPE = '15') | (&RTNTYPE = '17')) THEN (DO)
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*DIAG) +
MSGDTA ('Error with submitted INDEXBUILD job. See appropriate joblog!')
GOTO ERREXIT
ENDDO
DLTMSGQ MSGQ (&DLIB/AMSGQ)
MONMSG CPF0000
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*COMP) +
MSGDTA ('FILLPARTS program completed successfully!')
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*COMP) +
MSGDTA ('FILLPARTS program completed successfully!') +
TOMSGQ (&DLIB/AMSGQ)
GOTO ENDPGM
ERREXIT: SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) MSGTYPE (*INFO) +
MSGDTA ('FILLPARTS failed!') TOMSGQ (&DLIB/AMSGQ)
SNDFGMMSG MSGID (CPF9897) MSGF (QSYS/QCPFMMSG) +
MSGDTA ('FILLPARTS failed!') MSGTYPE (*ESCAPE)
ENDPGM: ENDPGM
/*-----*/
/* File name: FILLSTOCK */
/* Purpose: Fill the STOCK file. */
/*-----*/
/* NOTE: The logic is the same as in FILLCUS c file. */
/* PARMs: database library name */
/* fill_Q_name */
/* write_Q_name */
/* fill_space_name */
/* file_id */
/* instance number */
/* starting warehouse number */
/* number of warehouses */
/* our job number in the scheme of things */
/* total number of jobs to fill data */
/*-----*/
/* Includes. */
#include <dtlib.h>
#include <errno.h>
#include <recio.h>
#include <string.h>
#include <sysdb.h>
#include <mlib.h>
#include <micomput.h>
#include <decimal.h>
#include <qsyntax.h>
#include <qtchgb.h>
#include <mlh/cpybytes.h>
#include <mlh/cpyblap.h>
#include <mlh/trml.h>
#include <mlh/mattod.h>
#include <qsysinc/mlh/enq>
#include <qsysinc/mlh/deq>
#include <qsysinc/mlh/rsrvsp>
#include <cmmsg.h>
#include <filldata.h>
#include <commpntpc.h>
#include <tpccspace.h>
/*-----*/
/* MAPINC generates structures for the database file. */
#pragma mapinc ("stockpf", "LIBL/stockpf(*all)", "both", "d_P", "tpcc")
#include "stockpf"
/*-----*/
/* Defines */
/*-----*/
#define DBLIB_PARAM 1
#define FILL_Q_PARAM 2
#define WRITE_Q_PARAM 3
#define FILL_SPACE_PARAM 4
#define FILE_ID_PARAM 5
#define INST_NUMBER_PARAM 6
#define STRWH_PARAM 7
#define NUMWH_PARAM 8
#define JOB_NUMBER_PARAM 9
#define NUMBER_JOBS_PARAM 10
/*-----*/
/* Global declares */
/*-----*/
int file_id;
int inst_number;
char dlib[11];
char msgtxt[100];
char msgq_gl[20];
char pgname_gl[11];
char pgmlib_gl[11];
__SYSPTR write_Q_ptr = NULL;
__ENQ_Msg_Prefix_T ENQ_msg_prefix;
/*-----*/
/* Function: error_cleanup */
/*-----*/
void error_cleanup(void)
{
char Q_msg_text[MSG_LENGTH];
add_error_jobs(dlib, file_id, inst_number);
if (NULL != write_Q_ptr)
{
sprintf(Q_msg_text, "Q"); /* Quit. */
enq(write_Q_ptr, &ENQ_msg_prefix, Q_msg_text);
}
printf(msgtxt, "%s program failed!", pgname_gl);
QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBdy, 1);
return;
} /* End: error_cleanup() */
/*-----*/
/* Function: fill_stock */
/* Note: DistInf fields are declared in the FILLSTOCK H file. */
/*-----*/
void fill_stock(unsigned long record_number,
unsigned long num_record_per_IO_block,
unsigned long max_record,
char *data_space, int StartWH, int NumWH)
{
unsigned long record_count; /* Records filled in block. */
unsigned long current_record;
tpcc_STRCD_both_t *StkRec;
record_count = 0;
current_record = record_number;
while ((record_count < num_record_per_IO_block) &&
(current_record <= max_record))
{
StkRec = (tpcc_STRCD_both_t *)
&data_space[(record_count * sizeof(tpcc_STRCD_both_t))];
/*-----*/
/* Create STIDnn (district info) */
/*-----*/
memccpy(StkRec->STID01, DistInf1[RANDOM(0, 9)], 11);
memccpy(StkRec->STID01+11, DistInf2[0], 2);
memccpy(StkRec->STID01+13, DistInf3[RANDOM(0, 9)], 11);
/*-----*/
/* Fill stock record. */
/* Changed for uses with database version 3.4 */
/* The order is STIDD as the first key and STWID as the */
/* second key. */
/*-----*/
StkRec->STWID = (current_record - 1) % NumWH + 1 + StartWH;
StkRec->STIID = (current_record - 1) / NumWH + 1;
StkRec->STQTY = RANDOM(10,100);
}
}

```

## C.32 FILLSTOCK.C:



```

memcpy(StkRec->STD102, DistInfl[RANDOM(0, 9)], 11);
memcpy(StkRec->STD102+11, DistInfl2[1], 2);
memcpy(StkRec->STD102+13, DistInfl3[RANDOM(0, 9)], 11);

memcpy(StkRec->STD103, DistInfl[RANDOM(0, 9)], 11);
memcpy(StkRec->STD103+11, DistInfl2[2], 2);
memcpy(StkRec->STD103+13, DistInfl3[RANDOM(0, 9)], 11);

memcpy(StkRec->STD104, DistInfl[RANDOM(0, 9)], 11);
memcpy(StkRec->STD104+11, DistInfl2[3], 2);
memcpy(StkRec->STD104+13, DistInfl3[RANDOM(0, 9)], 11);

memcpy(StkRec->STD105, DistInfl[RANDOM(0, 9)], 11);
memcpy(StkRec->STD105+11, DistInfl2[4], 2);
memcpy(StkRec->STD105+13, DistInfl3[RANDOM(0, 9)], 11);

memcpy(StkRec->STD106, DistInfl[RANDOM(0, 9)], 11);
memcpy(StkRec->STD106+11, DistInfl2[5], 2);
memcpy(StkRec->STD106+13, DistInfl3[RANDOM(0, 9)], 11);

memcpy(StkRec->STD107, DistInfl[RANDOM(0, 9)], 11);
memcpy(StkRec->STD107+11, DistInfl2[6], 2);
memcpy(StkRec->STD107+13, DistInfl3[RANDOM(0, 9)], 11);

memcpy(StkRec->STD108, DistInfl[RANDOM(0, 9)], 11);
memcpy(StkRec->STD108+11, DistInfl2[7], 2);
memcpy(StkRec->STD108+13, DistInfl3[RANDOM(0, 9)], 11);

memcpy(StkRec->STD109, DistInfl[RANDOM(0, 9)], 11);
memcpy(StkRec->STD109+11, DistInfl2[8], 2);
memcpy(StkRec->STD109+13, DistInfl3[RANDOM(0, 9)], 11);

memcpy(StkRec->STD110, DistInfl[RANDOM(0, 9)], 11);
memcpy(StkRec->STD110+11, DistInfl2[9], 2);
memcpy(StkRec->STD110+13, DistInfl3[RANDOM(0, 9)], 11);

/*-----*/
/* Create STDATA random string 26 to 50 chars */
/*-----*/
CreateStrPad_extra(26, 50, StkRec->STDATA, "ORIGINAL");

StkRec->STYTD = 0;
StkRec->STORDRS = 0;
StkRec->STRECORD = 0;

/*-----*/
++record_count;
++current_record;
}
return;
) /* end of fill_stock */

/*-----*/
/*
/* Function: main
/*
/*-----*/
int main(int argc, char **argv)
{
short block, job_number;
int StartWH;
int NumWH;
int number_jobs, number_blocks_per_job;
error structure errCode;
unsigned int lib_len;

char slash_pos;

char fill_space[21];
char fill_space_ptr; /* Always points to start of fill space. */
char fill_space_ptr2; /* Points to different blocks with space. */

char fill_Q[11];
char write_Q[11];
SYSPTR fill_Q_ptr;

unsigned long rec_number; /* Relative record number. */
unsigned long num_record_per_IO_block;
unsigned long record_skip_size;
unsigned long last_record = 0;

char Q_msg_text[MSG_LENGTH];
DEQ_Msg_Prefix_T DEQ_Msg_Prefix;

struct {
Qus_Job_Change_Information_t jci;
Qus_JOBC0100_t x;
int priority;
} chgI;

ExcData_t volatile exD = {0, "HANDLE "};

/*-----*/
/* Start executable code.
/*-----*/

/* Set up the exception handler. Handle all escapes & FC excps. */
/* Branch to: ERR when an error occurs. */
#pragma exception_handler \
(ERR, 0, 0, C2_MH_ESCAPE | C2_MH_FUNCTION_CHECK, _CTLA_HANDLE)

errCode.bytes_prov = 0;

/*-----*/
/* Get the library name that we are running in.
/*-----*/
slash_pos = strchr(argv[0], '/');
lib_len = slash_pos - argv[0];
CPYBYTES(pgmlib_gl, argv[0], lib_len);
pgmlib_gl[lib_len] = '\0';

++slash_pos; /* Skip the '/' that follows the pgmlib field. */
CPYBYTES(pgmname_gl, slash_pos, 10);
pgmname_gl[triml(pgmname_gl, ' ')] = '\0';

/*-----*/
/* TPC message queue to send our build messages to.
/*-----*/
CPYBYTES(msqgl, BLD_MSQQ_NAME, 10);
CPYBYTES(msqgl, argv[DBLIB_PARM], 10);

dblib[10] = '\0';
CPYBYTES(dblib, argv[DBLIB_PARM], 10);
dblib[triml(dblib, ' ')] = '\0';

CPYBYTES(fill_space, argv[FILL_SPACE_PARM], 10);
CPYBYTES(fill_space+10, argv[DBLIB_PARM], 10);

job_number = atoi(argv[JOB_NUMBER_PARM]);

/*-----*/
sprintf(msgtxt, "FILLSTOCK: filling user space %10s in library %s...",
fill_space, dblib);
Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, INFO_MSG, msqgl, 1, NULL);

StartWH = atoi(argv[STRWH_PARM]) = 1;
inst_number = atoi(argv[INST_NUMBER_PARM]);
/*-----*/
/* Calculate the number of records and the starting record.
/*-----*/
}

/* Number of records is the (io block size) / record size.
/*-----*/
NumWH = atoi(argv[NUMWH_PARM]);
file_id = atoi(argv[FILE_ID_PARM]); /* Customer or Stock file? */

num_record_per_IO_block = IO_BLOCK_SIZE / sizeof(tpcc_STRCD_both_t);
last_record = NumWH * MAXITEMS;

/*-----*/
/* Set Queuing values.
/*-----*/
DEQ_msg_prefix.Wait_Forever = 1; /* Wait forever.
ENQ_msg_prefix.Msg_Len = MSG_LENGTH;

fill_Q[10] = '\0';
CPYBYTES(fill_Q, argv[FILL_Q_PARM], 10);
fill_Q[triml(fill_Q, ' ')] = '\0';
write_Q[10] = '\0';
CPYBYTES(write_Q, argv[WRITE_Q_PARM], 10);
write_Q[triml(write_Q, ' ')] = '\0';

if (NULL == (fill_Q_ptr = rslvsp(Usrq, fill_Q, dblib, _AUTH_ALL)))
{
sprintf(msgtxt, "Could not resolve to user queue, %s, in library %s.",
fill_Q, dblib);
Qsendmsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
error_cleanup();
return(-1);
}
if (NULL == (write_Q_ptr = rslvsp(Usrq, write_Q, dblib, _AUTH_ALL)))
{
sprintf(msgtxt, "Could not resolve to user queue, %s, in library %s.",
write_Q, dblib);
Qsendmsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
error_cleanup();
return(-1);
}

QUSPTRS(fill_space, &fill_space_ptr, &errCode); /* Get the ptr */
fill_space_ptr2 = fill_space_ptr;

/* Set up random number generator */
srand(get_aseed());

/*-----*/
/* Find the starting record number.
/*-----*/
number_jobs = atoi(argv[NUMBER_JOBS_PARM]);
rec_number = (job_number * num_record_per_IO_block) + 1;
record_skip_size = num_record_per_IO_block * number_jobs;

block = 1;
IO_BLOCKS_PER_JOB(number_blocks_per_job, num_record_per_IO_block,
last_record, number_jobs)

/* Change our job priority so we run a little slower than the
/* master job that called us.
/*-----*/
chgI.jci.Number_Fields_Entered = 1;
chgI.x.Length_Field_Info = sizeof(Qus_JOBC0100_t) + sizeof(int);
chgI.x.Key_Field = 1802;
chgI.x.Type_of_Data = 'B';
CPYBYTES(chgI.x.Reserved, " ", 3);
chgI.x.Length_Data = 4;
chgI.priority = 50;

QWTCGJB("JOBC0100", &chgI, &errCode);

if (errCode.bytes_avail > 0)
{
sprintf(msgtxt, "Unable to change FILLSTOCK job's priority.");
Qsendmsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
}

while (rec_number <= last_record)
{
/*-----*/
/* Wait until the storage area is available for us to fill.
/*-----*/
deq(&DEQ_msg_prefix, Q_msg_text, fill_Q_ptr);

if ('Q' == Q_msg_text[0])
{
add_successful_jobs(dblib, file_id, inst_number);
sprintf(msgtxt,
"Ending Fill job. 'Quit' message received on user queue.");
Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, INFO_MSG, msqgl, 1, NULL);
return;
}

/*-----*/
/* Start filling...
/*-----*/
fill_stock(rec_number, num_record_per_IO_block, last_record,
fill_space_ptr2, StartWH, NumWH);

sprintf(Q_msg_text, "%02d", block); /* Convert to char. */
enq(write_Q_ptr, &ENQ_msg_prefix, Q_msg_text);

if (block == number_blocks_per_job)
{
block = 1;
fill_space_ptr2 = fill_space_ptr; /* Reset to beginning of space. */
} else {
++block;
fill_space_ptr2 = fill_space_ptr2 + IO_BLOCK_SIZE;
}
rec_number += record_skip_size;
} /* End: while < last record. */

add_successful_jobs(dblib, file_id, inst_number);

sprintf(msgtxt,
"FILLSTOCK: job number %d successfully filled user space %10s.",
job_number, fill_space);
Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, COMP_MSG, msqgl, 1, NULL);
Qsendmsg(IBM_MSG_FILE, "CPF9897", msgtxt, COMP_MSG, CS_CTLBID, 1);
return;

#pragma disable_handler

/*-----*/
/* Any exceptions causes us to branch here.
/*-----*/
ERR:
error_cleanup();
return(-1);
} /* end of main */

/*-----*/
/*
/* Name: FILLWH
/*-----*/

```

## C.33 FILLWH.C:

```

/*
/* Purpose: fill the warehouse file.
/*
/*-----*/
#include <stdlib.h>
#include <string.h>
#include <decimal.h> /* Used by the wrhs include. */
#include <mlib.h>
#include <micomput.h>
#include <recio.h>
#include <time.h>
#include <mh/cpyblap.h>
#include <os4msg.h>
#include <commentpcc.h>
#include <filldata.h>

/*-----*/
/* Defines
/*-----*/
#define DBLIB_PARM 1
#define WHFILE_PARM 2
#define STRWH_PARM 3
#define NUMWH_PARM 4

/* MAFINC generates structures for the database files
#pragma mafinc("wrhs","LIBL/wrhs(*all)","both","d_p","tpcc")
#include "wrhs"

/*-----*/
/*
/* Name: main
/*
/*-----*/
int main(int argc, char **argv)
{
short int strWH, endWH, curWH;
int iX;

char dblib[11];
char whfile[11];
char msgtxt[50];
char msgq_gi[20];
char tempstr[40];

char curwhs[5];
char waddr1[21];
char waddr2[21];

time_t ltime;
FILE *WH_File;

tpcc_WRRCD_both_t WRec;

/* Set up random number generator */
time(&ltime);
srand(ltime); /* seed the random num generator */

dblib[10] = '\0';
strcpy(dblib, argv[DBLIB_PARM], 10);
dblib[triml(dblib, ' ')] = '\0';

whfile[10] = '\0';
strcpy(whfile, argv[WHFILE_PARM], 10);
whfile[triml(whfile, ' ')] = '\0';

strWH = atoi(argv[STRWH_PARM]);
endWH = (atoi(argv[NUMWH_PARM])) + strWH;

/* TPC message queue to send our build messages to */
strcpy(msgq_gi, "TPCCMSG ", 10);
strcpy(msgq_gi[10], argv[DBLIB_PARM], 10);

if (NULL == (WH_File = fopen(whfile, "r+")))
{
sprintf(msgtxt, "Open of file, %s, failed!", whfile);
Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gi, 1, NULL);
Qsendmsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDF, 1);
return(-1);
}

for (curWH = strWH; curWH < endWH; ++curWH)
{
WRec.WID = curWH;
sprintf(curwhs, "%04d", curWH);

/* build wname */
CreateStr(1, 5, tempstr);
strcat(tempstr, curwhs);
strcat(tempstr, "W");
cpyblap(WRec.WNAME, 10, tempstr, strlen(tempstr), ' ');

/* build address1 */
CreateStr(1, 10, waddr1);
sprintf(tempstr, "%s Ave.%s", curwhs, waddr1);
cpyblap(WRec.WADDR1, 20, tempstr, strlen(tempstr), ' ');

/* build address2 */
strcpy(tempstr, "Bldg 1");
strcat(tempstr, curwhs);
CreateStr(1, 10, waddr2);
strcat(tempstr, waddr2);
cpyblap(WRec.WADDR2, 20, tempstr, strlen(tempstr), ' ');

/* build city */
iX = RANDOM(0,99);
strcpy(WRec.WCITY, CITY_ARR[iX], CITY_LENGTH);
strcpy(WRec.WSTATE, state_arr[iX], STATE_LENGTH);

CREATE_ZIP(WRec.WZIP);

WRec.WTAX = (RANDOM(0,2000)) / 10000.0;
WRec.WYTD = 300000.00;

fwrite(WH_File, (void *)&WRec, sizeof(WRec));

fclose(WH_File);
return(0);
} /* end of main() */

DCL &NUMWHS TYPE(*DEC) LEN(6 0)
DCL &RECOMPFILE TYPE(*CHAR) LEN(1)

DCL &NUM1 TYPE(*DEC) LEN(6 0)
DCL &NUM2 TYPE(*DEC) LEN(6 0)

DCL &STRWHC TYPE(*CHAR) LEN(6)
DCL &NUMWHSC TYPE(*CHAR) LEN(6)
DCL &STRWHNULL TYPE(*CHAR) LEN(7)
DCL &NUMWHNULL TYPE(*CHAR) LEN(7)
DCL &NULLCH TYPE(*CHAR) LEN(1) VALUE('X'00')

DCL &PGMLIB TYPE(*CHAR) LEN(10)
DCL &MSGQ TYPE(*CHAR) LEN(10) VALUE(TPCCMSG)
DCL &STOCKSRC TYPE(*CHAR) LEN(10) VALUE(STOCKPF)

DCL &MDATA TYPE(*CHAR) LEN(42) +
VALUE('CRTSTOCK: filling the db file...')

MONMSG MSGID(CPF0000 MCH0000) EXEC(GOTO ERREXIT)

RTVJOB JOB(FILSTKFILE) OBJTYPE(*PGM) RTNLIB(&PGMLIB)
MONMSG CPF9811 EXEC(DO)
SNDPQMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*DIAG) +
MSGDTA('Add the library with CRTSTOCK pgm to lib list!')
GOTO ERREXIT
ENDDO

SNDPQMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) TOPQM(*EXT) +
MSGDTA('CRTSTOCK program starting...') MSGTYPE(*STATUS)

CRTMSGQ MSGQ(&DBLIB/&MSGQ)
MONMSG CPF2112 EXEC(RCVMSG RMV(*YES))

CALL TPCCTOOLS/SETTOOLIB

ADDLIB &DBLIB
MONMSG CPF2103 EXEC(RCVMSG RMV(*YES))

/*-----*/
IF (&RECOMPFILE *EQ 'Y') THEN(DO)

OVRDBF &STOCKSRC TOFILE(&DBLIB/&DBFILE) OVRSCOPE(*JOB)
MONMSG CPF2103 EXEC(RCVMSG RMV(*YES))

CRTCMOD &PGMLIB/FILLSTOCK

CRTPGM &PGMLIB/FILLSTOCK +
MODULE(&PGMLIB/FILLSTOCK &PGMLIB/COMMONTPCC +
&PGMLIB/TPCCSPACE) +
ENTMOD(&PGMLIB/FILLSTOCK) BNDSRVPGM(TPCCTOOLS/TPCCTOOLS)

DLTOVR FILE(&STOCKSRC) IVL(*JOB)
ENDDO

CHGVAR VAR(&ST MDATA 23 9) VALUE(&DBFILE)
SNDPQMMSG MSGID(CPF9897) MSGDTA(&MDATA) +
MSGF(QSYS/QCPFMMSG) TOPQM(*EXT) MSGTYPE(*STATUS)
SNDPQMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +
MSGDTA(&MDATA) TOMSGQ(&DBLIB/&MSGQ)

CHGVAR &NUMWHSC VALUE(&NUMWHS)
CHGVAR &NUMWHNULL VALUE(&NUMWHSC *CAT &NULLCH)
CHGVAR &STRWHC VALUE(&STRWH)
CHGVAR &STRWHNULL VALUE(&STRWHC *CAT &NULLCH)

/* WRKSYSSTS OUTPUT(*PRINT) RESET(*YES) */

CALL &PGMLIB/MASTCUSSTK PARM(&DBLIB &STRWHNULL &NUMWHNULL &DBFILE)

/* WRKSYSSTS OUTPUT(*PRINT) RESET(*NO) */

SNDPQMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*COMP) +
MSGDTA('CRTSTOCK program completed successfully!')
SNDPQMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*COMP) +
MSGDTA('CRTSTOCK program completed successfully!') +
TOMSGQ(&DBLIB/&MSGQ)
GOTO ENDPGM

ERREXIT: SNDPQMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) +
MSGDTA('CRTSTOCK failed!') MSGTYPE(*ESCAPE)
SNDPQMMSG MSGID(CPF9897) MSGF(QSYS/QCPFMMSG) MSGTYPE(*INFO) +
MSGDTA('CRTSTOCK failed!') TOMSGQ(&DBLIB/&MSGQ)

ENDPGM: ENDPGM

```

## C.35 MASTCUSSTK.C:

```

/*-----*/
/*
/* File name: MASTCUSSTK
/*
/* Purpose: This program spawns jobs to fill the following files:
/* - customer
/* - stock
/*
/* PARMS: database library name
/* starting warehouse number
/* number of warehouses
/* number of jobs we can start
/* name of file to fill
/*
/*-----*/
/* Includes.
/*-----*/
#include <stdlib.h> /* Malloc declares. */
#include <errno.h>
#include <string.h>
#include <recio.h>
#include <cxfsbk.h>
#include <unlstd.h>
#include <mlib.h>
#include <decimal.h>
#include <qmdesc.h>
#include <sys/stat.h>
#include <qscurts.h>
#include <qsdlts.h>
#include <qsqrts.h>
#include <qsqrctq.h>
#include <qsdltsq.h>
#include <qsqrjobi.h>
#include <qtcchj.h>
#include <mh/cpybytes.h>
#include <mh/triml.h>
#include <qsysinc/mih/rlvsp>
#include <qsysinc/mih/deq>
#include <qsysinc/mih/enq>
#include <spaw.h>
#include <qgwpid.h>
#include <commentpcc.h>
#include <tpccspace.h>
#include <filldata.h>
#include <os4msg.h>
#include <os4wait.h>

/*-----*/

```

## C.34 FILSTKFILE.CL:

```

/*-----*/
/*
/* NAME: FILSTKFILE
/*
/* PURPOSE: fill the following Database files:
/* - STOCKPF or for split DB: STOCKPF1 to STOCKPF32
/*-----*/
PGM PARM(&DBLIB &DBFILE &STRWH &NUMWHS &RECOMPFILE)

DCL &DBLIB TYPE(*CHAR) LEN(10)
DCL &DBFILE TYPE(*CHAR) LEN(10)
DCL &STRWH TYPE(*DEC) LEN(6 0)

/*-----*/

```

```

/* MAPINC generates structures for the database file. */
/*-----*/
#pragma mapinc("cstrmpf","*libl/cstrmpf(*all)","both","d_P","tpccc")
#include "cstrmpf"
#pragma mapinc("stockpf","*libl/stockpf(*all)","both","d_P","tpccc")
#include "stockpf"

/*-----*/
/* Defines */
/*-----*/
#define DELIB_PARM 1
#define START_WH_PARM 2
#define PARM_NUM_WHS 3
#define PARM_FILE_NAME 4

#define CUSTOMER_SPWN_PGM "FILLCUS"
#define CUSTOMER_NAME_PART_STRING "FILLCUS"
#define STOCK_SPWN_PGM "FILLSTCK"
#define STOCK_NAME_PART_STRING "FILLSTCK"

#define SET_FILL_QUEUE_NAME(q_name, job_num) \
{ \
    sprintf(name_temp, "FIL%01d%03d", name_tag, inst_number, job_num); \
    CPYBYTES(q_name, name_temp, 10); \
}

#define SET_WRITE_QUEUE_NAME(q_name, job_num) \
{ \
    sprintf(name_temp, "WRI%01d%03d", name_tag, inst_number, job_num); \
    CPYBYTES(q_name, name_temp, 10); \
}

#define SET_SPACE_NAME(spc_name, job_num) \
{ \
    sprintf(name_temp, "%03d", job_num); \
    CPYBYTES(spc_name+7, name_temp, 3); \
}

typedef _Packed struct key_struct {
    char name[10];
    long value;
} key_values_structure[5];

/*-----*/
/* Global declares */
/*-----*/
int number_jobs = MAX_JOBS;
int inst_number; /* instance number for the jobs
                  when the database is split. */

char ovr_flag = 'N';
char pgmlib_g1[11];
char pgmname_g1[11];
char dblib[11];
char queue_name[20];
char name_tag[4]; /* Part of queue or space name. */
char fill_space_name[20];
char name_temp[11];
char msgtxt[130];
char msgq_g1[20];

char *IO_space[MAX_JOBS];
char insert = 'S' /* value used for an insert using qdbrunha */
char hash_name[10];
long key_count; /* the key count used for call to qdbrunha */
long ret_data; /* return data from call to qdbrunha */
tpccc_CSTRCD_both_t * cstrcd_data_ptr; /* pointer to the customer file data, used for reference by qdbrunha */
tpccc_STRCD_both_t * stock_data_ptr; /* pointer to the stock file data */

key_values_structure key_values; /* keys for the hash function */
SYSPTR fill_Q_ptr[MAX_JOBS];
SYSPTR write_Q_ptr[MAX_JOBS];

char Q_msg_text[MSG_LENGTH];
_ENQ_Msg_Prefix_T ENQ_msg_prefix;

_RFILE *DB_File = NULL;

struct {
    Qws_Job_Change_Information_t jci;
    Qws_JOB0100_t x;
    int priority;
} chgl;

Qws_JOB0100_t jobI;
error_structure errCode;

void qdbrunha(char * hash_name, char function, long key_count,
              key_values_structure key_values, char * Returned_data,
              long * ret_data);

/*-----*/
/* Function: dlt_usrQ */
/*-----*/
void dlt_usrQ(void)
{
    int i, objname_number;

    errCode.bytes_prov = 16;

    for (i = 0; i < number_jobs; ++i)
    {
        objname_number = i + 1;

        if (NULL != fill_Q_ptr[i])
        {
            SET_FILL_QUEUE_NAME(queue_name, objname_number)
            QUSDLTUQ(queue_name, &errCode);
            if (errCode.bytes_avail > 0)
            {
                sprintf(msgtxt, "Error deleting user queue %10s. Error id = %07s",
                        queue_name, errCode.except_id);
                QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
            }
            fill_Q_ptr[i] = NULL;
        }

        if (NULL != write_Q_ptr[i])
        {
            SET_WRITE_QUEUE_NAME(queue_name, objname_number)
            QUSDLTUQ(queue_name, &errCode);
            if (errCode.bytes_avail > 0)
            {
                sprintf(msgtxt, "Error deleting user queue %10s. Error id = %07s",
                        queue_name, errCode.except_id);
                QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
            }
            write_Q_ptr[i] = NULL;
        }
    }
    return;
} /* end: dlt_usrQ() */

/*-----*/
/* Function: free_spaces */
/*-----*/
void free_spaces(void)
{
    int i;

    errCode.bytes_prov = 16;

    for (i = 0; i < number_jobs; ++i)
    {
        if (NULL != IO_space[i])
        {
            SET_SPACE_NAME(fill_space_name, {i+1})
            QUSDLTUQ(fill_space_name, &errCode);
            if (errCode.bytes_avail > 0)
            {
                sprintf(msgtxt,
                        "Error deleting temporary fill space '%.20s'. Error id = %07s",
                        fill_space_name, errCode.except_id);
                QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
            }
            IO_space[i] = NULL;
        }
    }
    return;
} /* end: free_spaces() */

/*-----*/
/* Function: send_FINAL_ERROR_MSG */
/*-----*/
/* Purpose: send the correct final escape message. */
/*-----*/
void send_FINAL_ERROR_MSG(char *msg_txt)
{
    int i;

    /* Send the error message in the msgtxt variable as a DIAG. */
    QsendMsg(IBM_MSG_FILE, "CPF9897", msg_txt, DIAG_MSG, CS_CUR_PGM, 1);

    /* Now cleanup - delete objects. */
    sprintf(Q_msg_text, "Q"); /* Quit. */
    for (i = 0; i < number_jobs; ++i)
    {
        if (NULL != fill_Q_ptr[i])
        {
            sprintf(msgtxt, "%s: (ENQ) ERROR! Sending Quit to fill job %d...",
                    pgmname_g1, i);
            Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, INFO_MSG, msgq_g1,
                    1, NULL);
            enq(fill_Q_ptr[i], &ENQ_msg_prefix, Q_msg_text);
        }
    }

    if (NULL != DB_File)
    {
        _Rclose(DB_File);
    }
    dlt_usrQ();
    free_spaces();

    /* Change our job's priority back to the original value. */
    if (jobI.Run_Priority > 0)
    {
        chgl.priority = jobI.Run_Priority;
        QWTCGJJB("JOB0C100", &chgl, &errCode);

        if (errCode.bytes_avail > 0)
        {
            sprintf(msgtxt,
                    "Unable to change our job's priority back to original value.");
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
        }
    }

    sprintf(msgtxt, "%s program failed!", pgmname_g1);
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDDY, 1);
    return;
} /* send_FINAL_ERROR_MSG() */

/*-----*/
/* Function: spawn_CS_fill_job */
/*-----*/
/* Purpose: spawn the customer and stock jobs */
/*-----*/
int spawn_CS_fill_job(char *dblib, int file_id, char *strWH, char *numWH,
                     int job_number, char *work_path)
{
    int rc, times, objname_number;
    int id count = 0;
    pid_t pid;
    inheritance_t inherit;
    char *env_arg[1];
    char *wp_arg[1];

    char ch_job_num[8];
    char ch_max_jobs[8];
    char ch_file_id[4];
    char ch_inst_num[3];
    char write_queue[10]; /* No need to send the dblib with q name. */

    objname_number = job_number + 1;
    SET_SPACE_NAME(fill_space_name, objname_number)
    SET_FILL_QUEUE_NAME(queue_name, objname_number)
    SET_WRITE_QUEUE_NAME(write_queue, objname_number)

    /*-----*/
    /* See if there is an available job for us to use. */
    /*-----*/
    times = 0;
    while ((0 == get_available_jobs(dblib)) && (-1 != rc))
    {
        /*-----*/
        /* Wait and try again... */
        /*-----*/
        tpccc_Wait(5, 0, NULL, NULL); /* Wait 5 seconds. */

        ++times;
        if (2880 == times) /* wait for 4 hours for an available job. */
        {
            sprintf(msgtxt,
                    "No available %s jobs. Tired of waiting for one. Quitting!",
                    get_DB_filename(file_id));
            send_FINAL_ERROR_MSG(msgtxt);
        }
    }

    /*-----*/
    sprintf(ch_job_num, "%03d", job_number); /* convert to char. */
    sprintf(ch_max_jobs, "%03d", number_jobs); /* convert to char. */
    sprintf(ch_file_id, "%03d", file_id); /* convert to char. */
    sprintf(ch_inst_num, "%03d", inst_number); /* convert to char. */

    /* We want names to start with 1 and not 0 for ascetic reasons. */
    objname_number = job_number + 1;
    SET_SPACE_NAME(fill_space_name, objname_number)
    SET_FILL_QUEUE_NAME(queue_name, objname_number)
    SET_WRITE_QUEUE_NAME(write_queue, objname_number)

    /*-----*/
    /* Set parms for the spawn. */
    /*-----*/
    wp_arg[0] = "dummy";
    wp_arg[1] = dblib;
    wp_arg[2] = queue_name;

```

```

wp_arg[3] = write_queue;
wp_arg[4] = fill_space_name;
wp_arg[5] = ch_file_id;
wp_arg[6] = ch_inst_num;
wp_arg[7] = strWH;
wp_arg[8] = numWH;
wp_arg[9] = ch_job_num;
wp_arg[10] = ch_max_jobs;
wp_arg[11] = NULL; /* terminate the array. */

env_arg[0] = NULL;
inherit.flags = 0;
inherit.pgroup = SPAWN_NEWPGROUP;

times = 0;
while ((0 != add_started_jobs(dblib, file_id, inst_number)) && (-1 != rc))
{
    /*-----*/
    /* wait and try again... */
    /*-----*/
    tpcc_wait(3, 0, NULL, NULL); /* Wait 3 seconds. */

    ++times;
    if (100 == times) /* Total of 5 minutes wait time. */
    {
        sprintf(msgtxt, "I can't add to the number of %s jobs I am starting!",
            get_DB_filename(file_id));
        send_FINAL_ERROR_MSG(msgtxt);
    }
}

/*-----*/
pid = spawn(work_path, fd_count, NULL, &inherit,
    (char **)(&wp_arg), (char **)(&env_arg));

if (pid < 0)
{
    rc = 0;
    times = 0;
    while ((0 != add_error_jobs(dblib, file_id, inst_number)) && (-1 != rc))
    {
        /*-----*/
        /* Wait and try again... */
        /*-----*/
        tpcc_wait(3, 0, NULL, NULL); /* Wait 3 seconds. */

        ++times;
        if (100 == times) /* Total of 5 minutes wait time. */
        {
            sprintf(msgtxt, "I can't set 'error' jobs value for %s files!",
                get_DB_filename(file_id));
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
        }

        sprintf(msgtxt, "Link '%s' not spawned. Process id = %d", work_path, pid);
        send_FINAL_ERROR_MSG(msgtxt);
    }
    return(0);
} /* End of: spawn_CS_fill_job() */

/*-----*/
/* Function: write_IO */
/*-----*/
int write_IO(size_t record_size, unsigned long last_record,
    unsigned long number_records_per_IO_block,
    int num_blocks_per_job)
{
    short block, job;
    int block_size;
    unsigned long rec; /* Record loop index. */
    unsigned long current_record = 1;
    char *out_rec;

    _DEQ_Msg_Prefix_T DEQ_msg_prefix;

    /*-----*/
    DEQ_msg_prefix.Wait_Forever = 1; /* Wait forever. */
    /*-----*/

    while (current_record <= last_record)
    {
        for (block = 0;
            (block < num_blocks_per_job) && (current_record <= last_record);
            ++block)
        {
            block_size = IO_BLOCK_SIZE * block;
            for (job = 0;
                (job < number_jobs) && (current_record <= last_record);
                ++job)
            {
                deq(&DEQ_msg_prefix, Q_msg_text, write_Q_ptr[job]);
                if ('Q' == Q_msg_text[0])
                {
                    sprintf(msgtxt,
                        "Ending Master job. 'Quit' message received on user queue.");
                    send_FINAL_ERROR_MSG(msgtxt);
                }
            }
            /*-----*/
            /*
            *s: writing - file ' ', job %d, block %d, record %d... */
            /*-----*/
            pgmname_gl, job, block+1, current_record;
            /*
            *sendsmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, INFO_MSG, msgq_gl,
            * l, NULL);
            /*-----*/

            for (rec = 0;
                (rec < number_records_per_IO_block) &&
                (current_record <= last_record);
                ++rec)
            {
                /* Get the record position in fill array. */
                /* User the qdbrunha to do the writing */
                out_rec = IO_space[job] + block_size + (rec * record_size);
                if (hash_name[0] == 'C')
                {
                    cstmr_data_ptr = (tpcc_CSRCID both t *)out_rec;
                    key_values[0].value = cstmr_data_ptr->CID;
                    /* get cid */
                    key_values[1].value = cstmr_data_ptr->CDID;
                    /* get cdid */
                    key_values[2].value = cstmr_data_ptr->CWID;
                    /* get cwid */
                }
                else
                {
                    /* the file is a stock file process as a stock record */
                    stock_data_ptr = (tpcc_SSRCID both t *)out_rec;
                    key_values[0].value = stock_data_ptr->STIID;
                    /* get the item id */
                    key_values[1].value = stock_data_ptr->STWID;
                    /* get the stock warehouse id */
                }
                qdbrunha ( hash_name, insert, key_count, key_values,
                    (char*) out_rec, &ret_data );
                /*
                *writes(DB_File, (void *)out_rec, record_size); */
                ++current_record;
            }
            /*-----*/
            /* Let Fill job know it can fill the next block of data. */
            /*-----*/
            if (current_record <= last_record)
                sprintf(Q_msg_text, "%02d", block); /* Convert to char. */
            else
                sprintf(Q_msg_text, "Q"); /* Quit. */

            enq(fill_Q_ptr[job], &ENQ_msg_prefix, Q_msg_text);
        } /* End: job loop. */
    } /* End: block loop. */
} /* end: while loop. */
return(0);
} /* End of: write_IO() */

/*-----*/
/* Function: un_link */
/*-----*/
void un_link(char *name_part)
{
    int jj;
    char work_path[22];
    char *WorkPathEnd; /* pointer to end of the working path. */

    sprintf(work_path, "%s/%s", pgmlib_gl, name_part);
    WorkPathEnd = &work_path[strlen(work_path)];

    for (jj = 0; jj < number_jobs; ++jj)
    {
        sprintf(WorkPathEnd, "%03d", (jj+1));
        if (-1 == unlink(work_path))
        {
            sprintf(msgtxt, "FYI: could NOT unlink the symbolic link %s\n", work_path);
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
        }
    }
} /* End of: un_link() */

/*-----*/
/* Function: build_link */
/*-----*/
int build_link(char *hard_path, char *work_path)
{
    struct stat checker;

    if (lstat(work_path, &checker) == 0) /* If the link exists */
    {
        /* Unlink the sybolic link. */
        /* (it may be to another program so it should be removed. */
        if (-1 == unlink(work_path))
        {
            sprintf(msgtxt, "Could NOT unlink the symbolic link %s\n", work_path);
            send_FINAL_ERROR_MSG(msgtxt);
        }
    } /* End: link exists. */

    if (0 != symlink(hard_path, work_path))
    {
        /* symlink not created. */
        sprintf(msgtxt, "Could NOT create a symbolic link '%s' to '%s'",
            work_path, hard_path);
        send_FINAL_ERROR_MSG(msgtxt);
    }
    return(0);
} /* End of: build_link() */

/*-----*/
/* Function: crt_queues */
/*-----*/
void crt_queues(void)
{
    /* User queue values. */
    /*-----*/
    int job, objname_number;
    int initial_msgs; /* Number of user queue msgs. */
    int addl_msgs; /* additional number of msgs. */

    char just_name[11];

    char authority[10] = "ALL ";
    char replace[10] = "YES ";
    char description[50] = "TPCC user queue for job ";

    char type[] = "F"; /* First in first out. */

    /*-----*/
    errCode.bytes_avail = 16;
    initial_msgs = (number_jobs * NUM_IO_BLOCKS); /* All we should need. */
    addl_msgs = (MAX_JOBS * NUM_IO_BLOCKS) - initial_msgs;

    copyblap(queue_name+10, 10, dblib, strlen(dblib), ' ');

    /*-----*/
    /* Create the message queues for the files. */
    /*
    * Mag queue names will be FILCUS0001 and FILSTRK0001 and so on.
    * Mag queue names will be WRICUS0001 and WRISTRK0001 and so on.
    *-----*/
    for (job = 0; job < number_jobs; ++job)
    {
        objname_number = job + 1; /* So names start with 1 and not 0. */
        SET_FILL_QUEUE_NAME(queue_name, objname_number)

        QUSCRTUQ(queue_name, "TPCCMSG ", type, 0, MSG_LENGTH, initial_msgs,
            addl_msgs, authority, description, replace, &errCode);
        if (errCode.bytes_avail > 0)
        {
            if (0 != memcmp(errCode.except_id, "CPF9870", 7))
            {
                sprintf(msgtxt, "Error creating fill queue '%.20s'. Error id = %.07s",
                    queue_name, errCode.except_id);
                send_FINAL_ERROR_MSG(msgtxt);
            }
        }
        /*-----*/
        /* Get pointer to the user queue. */
        /*-----*/
        _CPYBYTES(just_name, queue_name, 10);
        just_name[10] = '\0';
        if (NULL == (fill_Q_ptr[job] = rslvsp(Usrq, just_name, dblib, AUTH_ALL)))
        {
            sprintf(msgtxt, "Could not resolve to user queue: %s.", queue_name);
            send_FINAL_ERROR_MSG(msgtxt);
        }
    }
    /*-----*/
    SET_WRITE_QUEUE_NAME(queue_name, objname_number)

    QUSCRTUQ(queue_name, "TPCCMSG ", type, 0, MSG_LENGTH, initial_msgs,
        addl_msgs, authority, description, replace, &errCode);
    if (errCode.bytes_avail > 0)
    {
        if (0 != memcmp(errCode.except_id, "CPF9870", 7))
        {

```

```

        sprintf(msgtxt, "Error creating write queue '%.20s'. Error id = %.07s",
                queue_name, errCode.except_id);
        send_FINAL_ERROR_MSG(msgtxt);
    }
}
/*-----*/
/* Get pointer to the user queue. */
/*-----*/
_CPYBYTES(just_name, queue_name, 10);
Just_name[10] = '\0';
if (NULL == (write_Q_ptr[job] = rslvsp(Usrq, just_name, dlib, AUTH_ALL)))
{
    sprintf(msgtxt, "Could not resolve to user queue: %s.", queue_name);
    send_FINAL_ERROR_MSG(msgtxt);
}
}
return;
} /* End of: crt_queues() */

/*-----*/
/* Function: crt_spaces */
/*-----*/
void crt_spaces(void)
{
    int job;
    char init_char;
    char replace[10] = "YES ";
    char public_auth[10] = "ALL ";
    char text_descript[50];

    cpyblap(text_descript, 50, "Space to temporarily hold fill data.",
            strlen("Space to temporarily hold fill data."), ' ');
    errCode.bytes_prov = 16;
    init_char = 0;

    sprintf(fill_space_name, "FIL%04d", name_tag, inst_number);
    cpyblap(fill_space_name+10, 10, dlib, strlib(dlib), ' ');

    for (job = 0; job < number_jobs; ++job)
    {
        /*-----*/
        /* Space names will be FILC0001 and FILSTK0001, etc. */
        /*-----*/
        SET_SPACE_NAME(fill_space_name, (job+1))

        QUSCRTUS(fill_space_name, "PF ", (IO_BLOCK_SIZE * NUM_IO_BLOCKS),
                &init_char, public_auth, text_descript, &errCode);
        if (errCode.bytes_avail > 0)
        {
            sprintf(msgtxt, "Error creating data space '%.20s'. Error id = %.07s",
                    fill_space_name, errCode.except_id);
            send_FINAL_ERROR_MSG(msgtxt);
        }

        QUSPTRUS(fill_space_name, &IO_space[job], &errCode); /* Get the ptr */
        if (errCode.bytes_avail > 0)
        {
            sprintf(msgtxt,
                    "Error getting data space '%.20s' pointer. Error id = %.07s",
                    fill_space_name, errCode.except_id);
            send_FINAL_ERROR_MSG(msgtxt);
        }
    }
    return;
} /* End of: crt_spaces() */

/*-----*/
/* Function: main */
/*-----*/
int main(int argc, char **argv)
{
    int rc;
    unsigned int lib_len;
    short i; /* Loop control. */
    int job;
    int error_count;
    int length; /* string length of work_path. */
    int file_id;
    int times;
    int number_blocks_per_job;
    unsigned long number_records_per_IO_block;

    char *slash_pos;

    char *WorkPathEnd; /* pointer to the end of the working path. */
    char work_path[22]; /* path name for the link used as job name. */
    char name_part[8]; /* job name string */
    char file_name[11];
    char hard_path[41];
    char pgm_to_spawn[11];
    char cmd[100];

    size_t record_size;
    unsigned long last_record;

    ExcData_t volatile exD = {0, "HANDLE "};

    /*-----*/
    /* Start of code. */
    /*-----*/
    jobI.Run_Priority = 0; /* Initialize in case we have an error. */

    /*-----*/
    /* Get the library name that we are running in. */
    /*-----*/
    slash_pos = strchr(argv[0], '/');
    lib_len = slash_pos - argv[0];
    _CPYBYTES(pgmlib_g1, argv[0], lib_len);
    pgmlib_g1[lib_len] = '\0';

    ++slash_pos; /* Skip the '/' that follows the pgmlib field. */
    _CPYBYTES(pgmname_g1, slash_pos, 10);
    pgmname_g1[triml(pgmname_g1, ' ')] = '\0';

    /* TPC message queue to send our build messages to. */
    _CPYBYTES(msgq_g1, BLD_MSGQ_NAME, 10, 10);
    _CPYBYTES(&(msgq_g1[10]), argv[DBLIB_PARAM], 10);

    /*-----*/
    if (1 == atoi(argv[START_WH_PARAM]))
        inst_number = 0; /* normal or low split. */
    else
        inst_number = 1; /* high split */

    /*-----*/
    /* Initialize in case we have an error. That way we won't try to */
    /* destroy non-existent spaces and user queues. */
    /*-----*/
    for (job = 0; job < number_jobs; ++job)
    {
        IO_space[job] = NULL;
        fill_Q_ptr[job] = NULL;
        write_Q_ptr[job] = NULL;
    }
    /*-----*/

    /* Get the file name and related parameters. */
    /*-----*/
    dlib[10] = '\0';
    _CPYBYTES(dlib, argv[DBLIB_PARAM], 10);
    dlib[triml(dlib, ' ')] = '\0';

    file_name[10] = '\0';
    _CPYBYTES(file_name, argv[PARAM_FILE_NAME], 10);
    file_name[triml(file_name, ' ')] = '\0';

    switch(file_name[0])
    {
        case 'C':
            file_id = CUSTOMER_JOBS;
            sprintf(name_tag, "CUS");
            sprintf(pgm_to_spawn, CUSTOMER_SPWN_PGM);
            sprintf(name_part, CUSTOMER_NAME_PART_STRING);
            record_size = CUSTOMER_RECORD_SIZE;
            last_record =
                MAX_DISTRICTS * CUSTPERDIST * atoi(argv[PARAM_NUM_WHS]);
            number_jobs = get_maxjobs_available(dlib, CUSTOMER_JOBS, inst_number);
            /* insert information for hash */
            memcpy(hash_name, "CSTM", 10); /* insert the hash name */
            key_count = 3; /* customer has three keys */
            break;
        case 'S':
            file_id = STOCK_JOBS;
            sprintf(name_tag, "STK");
            sprintf(pgm_to_spawn, STOCK_SPWN_PGM);
            sprintf(name_part, STOCK_NAME_PART_STRING);
            record_size = STOCK_RECORD_SIZE;
            last_record = MAXITEMS * atoi(argv[PARAM_NUM_WHS]);
            number_jobs = get_maxjobs_available(dlib, STOCK_JOBS, inst_number);
            /* insert information for hash */
            memcpy(hash_name, "STOCK", 10); /* insert the hash name */
            key_count = 2; /* stock has two keys */
            break;
        default:
            sprintf(msgtxt, "Unknown file name: '%s'", file_name);
            send_FINAL_ERROR_MSG(msgtxt);
    }
    break;
} /* end of switch */

/*-----*/
/* Error getting job number from space? */
/*-----*/
if (-1 == number_jobs)
{
    sprintf(msgtxt, "ERROR retrieving the 'max jobs available' value for %s.",
            get_DB_filename(file_id));
    send_FINAL_ERROR_MSG(msgtxt);
}

/*-----*/
/* Save job info so Master control can see and we can use later. */
/*-----*/
if (0 != set_plan_to_start(dlib, file_id, inst_number, number_jobs))
{
    sprintf(msgtxt, "I can't set number of jobs (%d) I plan to spawn!",
            number_jobs);
    send_FINAL_ERROR_MSG(msgtxt);
}

sprintf(msgtxt, "I plan to spawn %d jobs to fill the %s file.",
        number_jobs, get_DB_filename(file_id));
Qsendmsg_MQ(IMM_MSG_FILE, "CPF9897", msgtxt, INFO_MSG, msgq_g1, 1, NULL);

/*-----*/
crt_spaces();
crt_queues();

/*-----*/
/* One time setup for job spawn. */
/*-----*/
sprintf(hard_path, "%SVC.LIB/%s.LIB/%s.PGM", pgmlib_g1, pgm_to_spawn);
sprintf(work_path, "%S/%s", pgmlib_g1, name_part);
length = strlen(work_path);
WorkPathEnd = &work_path[length];

/*-----*/
/* One time setup for job queuing. */
/*-----*/
ENQ_msg_prefix.Msg_Len = MSG_LENGTH;

number_records_per_IO_block = IO_BLOCK_SIZE / record_size;

IO_BLOCKS_PER_JOB(number_blocks_per_job, number_records_per_IO_block,
        last_record, number_jobs);

/*-----*/
/* Start the LOOP to fill jobs.. */
/*-----*/
for (job = 0; job < number_jobs; ++job)
{
    times = 0;
    while (0 == (rc = get_available_jobs(dlib)))
    {
        sprintf(msgtxt, "%s: no jobs available. Waiting...", pgmname_g1);
        Qsendmsg_MQ(IMM_MSG_FILE, "CPF9897", msgtxt, INFO_MSG, msgq_g1, 1, NULL);
        /* Wait and try again.. */
        /*-----*/
        tpcc_Wait(5, 0, NULL, NULL); /* Wait 5 seconds. */
        ++times;
        if (2880 == times) /* wait for 4 hours for an available job. */
        {
            sprintf(msgtxt,
                    "%s: no available %s jobs. Tired of waiting for one. Quitting!",
                    pgmname_g1, get_DB_filename(file_id));
            send_FINAL_ERROR_MSG(msgtxt);
        }
    }
    if (-1 == rc)
    {
        sprintf(msgtxt,
                "Error getting an available job for filling the %s file.",
                get_DB_filename(file_id));
        send_FINAL_ERROR_MSG(msgtxt);
    }

    /*-----*/
    /* Tell spawned job he can starting immediately filling data. */
    /* The mesch is to enqueue the data blk number. */
    /* We don't need a number as all we need are a number of msgs, */
    /* but having a number may make debugging easier.. */
    /*-----*/
    for (i = 1; i <= number_blocks_per_job; ++i)
    {
        sprintf(Q_msg_text, "%02d", i); /* Convert from short to char. */
        eng(fill_Q_ptr[job], &ENQ_msg_prefix, Q_msg_text);
    }

    /*-----*/
    /* Spawn the jobs to fill the file. */
    /* (Add 1 to job number so the numbers range from 1 to n and */
    /* 0 to n). */
    /*-----*/
    sprintf(WorkPathEnd, "%03d", (job+1));
    build_link(hard_path, work_path);
    spawn_CS_fill_job(argv[DBLIB_PARAM], file_id, argv[START_WH_PARAM],

```

```

        argv[PARAM_NUM_WHS], job, work_path);
} /* End: loop to start fill jobs. */

/*-----*/
/* Now that the jobs have been spawned, change our priority to a
/* value that ensures we will have a higher priority than our
/* spawned jobs. Note: we wait until now as the spawned jobs are
/* spawned with the same priority as this job when they are
/* spawned.
/*-----*/

/*-----*/
/* Get current job's run priority.
/*-----*/
QUSRJOB1(job1, sizeof(job1), "JOB0100",
        "",
        "", &errCode);

if (errCode.bytes_avail > 0)
{
    printf(msgtxt, "Unable to retrieve our job's priority.");
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
} else {
    /*-----*/
    /* Change our job priority so we run a higher priority (faster).
    /*-----*/
    chgl.jci.Number_Fields_Entered = 1;
    chgl.x.Length_Field_Info = sizeof(Qus_JOB0100_t) + sizeof(int);
    chgl.x.Key_Field = 1802;
    chgl.x.Type_Of_Data = 'B';
    _CPYBYTES(chgl.x.Reserved, " ", 3);
    chgl.x.Length_Data = 4;
    chgl.priority = 10;

    QWTCGJB("JOB0100", &chgl, &errCode);

    if (errCode.bytes_avail > 0)
    {
        printf(msgtxt, "Unable to change our job's priority.");
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
    }
} /* We were able to retrieve our job's priority. */

/*-----*/
/* Setup Override of DB file while we wait for all jobs to start.
/*-----*/
sprintf(cmd,
        "QVRDBF FILE(%s) TOFILE(%s) SEQONLY(*YES %lu) OVRSCOPE(*JOB)",
        file_name, dblib, file_name, number_records_per_IO_block);

#pragma exception_handler(QexcepHdir, exD, 0, _C2_MH_ESCAPE)
QCMDEXC(cmd, strlen(cmd));
#pragma disable_handler

if (0 != exD.error)
{
    printf(msgtxt, "Override of file '%s' failed.", file_name);
    send_FINAL_ERROR_MSG(msgtxt);
}

ovr_flag = 'Y';

/*-----*/
/* Now OPEN the DB file.
/*-----*/
if (NULL == (DB_file = _Ropen(file_name, "ar,arrrseq=Y,blkrcd=Y,riofb=N")))
{
    printf(msgtxt, "Open of %s with override %s/%s failed.",
            file_name, dblib, file_name);
    send_FINAL_ERROR_MSG(msgtxt);
}

/*-----*/
/* Check that no jobs failed.
/*-----*/
if (0 < get_error_jobs(dblib, file_id, inst_number))
{
    printf(msgtxt, "The number of fill jobs in error > 0. Check joblogs.");
    send_FINAL_ERROR_MSG(msgtxt);
}

/*-----*/
/* Now that all the jobs have been spawned, start writing data
/* that they have filled in the temporary space, to the permanent
/* data base file.
/*-----*/
write_IO(record_size, last_record, number_records_per_IO_block,
        number_blocks_per_job);

/*-----*/
/* Cleanup!
/*-----*/
_Rclose(DB_file);
dlr_usrQ();
free_spaces();
un_link(name_part);

sprintf(cmd, "DLTOVR FILE(%s) LVL(*JOB)", file_name);

#pragma exception_handler(QexcepHdir, exD, 0, _C2_MH_ESCAPE)
QCMDEXC(cmd, strlen(cmd));
#pragma disable_handler

if (job1.Run_Priority > 0)
{
    chgl.priority = job1.Run_Priority;

    QWTCGJB("JOB0100", &chgl, &errCode);

    if (errCode.bytes_avail > 0)
    {
        printf(msgtxt,
                "Unable to change our job's priority back to original value.");
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
    }
}

return;
} /* End: main() */

```

## C.36 MASTORD.C:

```

/*-----*/
/* FILE: MASTORD
/*
/* PURPOSE: control program to spawns jobs to fill the neword,
/* orders, and ordlin files.
/* It fills them all simultaneously.
/*-----*/

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <qustvus.h>
#include <qustrus.h>

```

```

#include <qustvus.h>
#include <quopen.h>
#include <xcvt.h>
#include <time.h>

/*-----*/
/* Spawn declares
/*-----*/

#include <spawn.h>
#include <sys/stat.h>
#include <qspwid.h>
#include <unistd.h>
#include <mlh/cpybytes.h>
#include <mlh/trml.h>
#include <QSYSINC/MIH/CMPSWF>
#include <QSYSINC/MIH/WAITTIME>
#include <tpcspc.h>
#include <osmsg.h>
#include <os4wait.h>

/*-----*/
#define DBLIB_PGM 1
#define PARM_ORDERS_NAME 2
#define PARM_ORDLIN_NAME 3
#define PARM_NEWORD_NAME 4
#define STRWH_PARM 5
#define NUMWH_PARM 6

#define SPAWN_PGM "FILLORD"

/*-----*/
/* Globals
/*-----*/
char pgmlib_gl[11];
char pgmname_gl[11];
char msgtxt[133];
char msgq_gl[20];
char dblib[11];

/*-----*/
int start_tasks(int *num_jobs, int jobs_alloc, int str_wh, int total_whs,
               int inst_num, char *dblib,
               char *orders_file, char *ordlin_file, char *neword_file)
{
    int wh_per_job, have_extra, num_wh;
    int lp;
    unsigned int plen;
    char job_num[4] = " ";

    inheritance_t inherit;
    int fd_count = 0;
    char *env_arg[1];
    char *wp_arg[10];
    char work_path[22];
    char hard_path[41];
    char str_string[8];
    char num_string[8];
    char number_string[8];
    char instance_string[3];
    pid_t pid;
    struct stat checker; /* used to unlink path */

    /*-----*/
    /* Start executable code.
    /*-----*/
    if (jobs_alloc < total_whs)
    {
        *num_jobs = jobs_alloc; /* Use all jobs available. */
    } else {
        *num_jobs = total_whs;
    }
    wh_per_job = total_whs / (*num_jobs);
    have_extra = total_whs % (*num_jobs);

    if (0 == have_extra)
    {
        num_wh = wh_per_job;
    } else {
        /* Do an extra warehouse for this job. */
        num_wh = wh_per_job + 1;
    }

    /*-----*/
    /* Save job info so Master control can see and we can use later.
    /*-----*/
    if (0 != set_plan_to_start(dblib, ORDER_JOBS, inst_num, *num_jobs))
    {
        printf(msgtxt, "I can't set number of jobs (%d) I plan to spawn!");
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
        return(-1);
    }

    printf(msgtxt, "MASTORD: I plan to spawn %d jobs to fill the %s files.",
            *num_jobs, get_DB_filename(ORDER_JOBS));
    QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);

    sprintf(hard_path, "QSYS.LIB/%s.LIB/%s.PGM", pgmlib_gl, SPAWN_PGM);
    sprintf(work_path, "%s/%s", pgmlib_gl, SPAWN_PGM);
    plen = strlen(work_path);

    /*-----*/
    /* Set parms for the spawn.
    /*-----*/
    for (lp = 1; lp <= *num_jobs; ++lp)
    {
        /* Set job name number. */
        sprintf(work_path, "%03d", lp); /* Add the job number. */
        sprintf(job_num, "%03d", lp);

        if (lstat(work_path, &checker) == 0) /* If the link exists */
        {
            /* Unlink the symbolic link.
            /* (it may be to another program so it should be removed.
            /* if (-1 == unlink(work_path))
            {
                char msgid[8];
                /* Send the errno message text message.
                printf(msgid, "CPE%d", errno);
                QsendMsg(IBM_MSG_FILE, msgid, "", DIAG_MSG, CS_CTLBDY, 1);

                printf(msgtxt, "Could NOT unlink path: '%s'", work_path);
                QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
                return (-1); /* abort the operation */
            }
        } /* End: link exists. */

        if (0 != symlink(hard_path, work_path))
        {
            char msgid[8];
            /* Send the errno message text message.
            printf(msgid, "CPE%d", errno);
            QsendMsg(IBM_MSG_FILE, msgid, "", DIAG_MSG, CS_CTLBDY, 1);

            printf(msgtxt, "Could NOT create a symbolic link '%s' to '%s'",
                    work_path, hard_path);
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDY, 1);
            return(-1);
        }

        /*-----*/
        /* Spawn the job.
        /*-----*/
    }
}

```

```

/*-----*/
wp_arg[0] = "dummy";
sprintf(str_string, "%05d", str_wh);
sprintf(num_string, "%05d", num_wh);
sprintf(instance_string, "%d", inst_num);

wp_arg[1] = str_string;
wp_arg[2] = num_string;
wp_arg[3] = instance_string;
wp_arg[4] = job_num;
wp_arg[5] = dblib;
wp_arg[6] = orders_file;
wp_arg[7] = ordlin_file;
wp_arg[8] = neword_file;
wp_arg[9] = NULL;

env_arg[0] = NULL;
inherit.flags = 0;
inherit.pgroup = SPAWN_NEWPGROUP;

pid = spawn(work_path, fd_count, NULL, &inherit,
            (char **)(&wp_arg), (char **)(&env_arg));
if (pid < 0)
{
    sprintf(msgtxt, "Spawn %d failed!\n", lp);
    Qsendmsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDV, 1);
    return(-1);
}

/*-----*/
str_wh = str_wh + num_wh; /* Next starting warehouse id.*/
if (lp == have_extra)
{
    --num_wh; /* Start doing 1 less warehouse. */
}
} /* end: for loop */

return(0);
} /* end: start_tasks() */

/*-----*/
/*
 * Function: main
 */
/*-----*/
int main(int argc, char **argv)
{
    int strWH, numWhs;
    int number_jobs;
    int max_allocated;
    int InstNum; /* Instance number used in split data base */
    int time_passed; /* time out counter for wait loops */

    unsigned int lib_len;
    char *slash_pos;

    /*-----*/
    /* Wait time variables (key think time waits) */
    /*-----*/
    _MI_time time_to_wait; /* The amount of time to wait */
    int hours = 0; /* Time components used to */
    minutes = 0, /* create an MI_time value */
    seconds = 1, /* that can be passed to the */
    hundredths = 0; /* waittime function */
    short wait_option = _WAIT_NORMAL; /* wait time options */
    /* format an mitime value */
    /* representing the wait time */

    /*-----*/
    /* Get the library name that we are running in now. */
    /*-----*/
    slash_pos = strchr(argv[0], '/');
    lib_len = slash_pos - argv[0];
    _CPYBYTES(pgmlib_gl, argv[0], lib_len);
    pgmlib_gl[lib_len] = '\0';

    ++slash_pos; /* Skip the '/' */
    _CPYBYTES(pgmname_gl, slash_pos, 10);
    pgmname_gl[triml(pgmname_gl, ' ')] = '\0';

    dblib[10] = '\0';
    _CPYBYTES(dblib, argv[DBLIB_PARAM], 10);
    dblib[triml(dblib, ' ')] = '\0';

    /* TPCC message queue to send our build messages to. */
    _CPYBYTES(msgq_gl, "TPCCMSG ", 10);
    _CPYBYTES(&msgq_gl[10], argv[DBLIB_PARAM], 10);

    strWH = atoi(argv[STRWH_PARAM]);
    numWhs = atoi(argv[NUMWH_PARAM]);
    if (1 == strWH)
    else
        InstNum = 1; /* high split section */

    /*-----*/
    /* Get max number of jobs we can use. */
    /*-----*/
    max_allocated = get_maxjobs_available(dblib, ORDER_JOBS, InstNum);

    if (max_allocated < 1)
    {
        sprintf(msgtxt, "We were allocated only %d jobs. This is not enough!",
                max_allocated);
        Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDV, 1);
        return(-1);
    }

    if (start_tasks(number_jobs, max_allocated, strWH, numWhs,
                    InstNum, argv[DBLIB_PARAM], argv[PARAM_ORDERS_NAME],
                    argv[PARAM_ORDLIN_NAME], argv[PARAM_NEWORD_NAME]) != 0)
    {
        sprintf(msgtxt, "Error was detected while trying to start tasks!\n");
        Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDV, 1);
        return(-1);
    }

    mitime(time_to_wait, hours, minutes, seconds, hundredths);
    time_passed = 0;
    while ((get_started_jobs(dblib, ORDER_JOBS, InstNum) +
            get_successful_jobs(dblib, ORDER_JOBS, InstNum)) < number_jobs)
    {
        waittime(time_to_wait, wait_option);
        if (time_passed > 70)
        {
            sprintf(msgtxt, "Error only %d of %d tasks started!\n",
                    get_started_jobs(dblib, ORDER_JOBS, InstNum) +
                    get_successful_jobs(dblib, ORDER_JOBS, InstNum),
                    number_jobs);
            Qsendmsg_MQ(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, msgq_gl, 1, NULL);
            QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, ESCAPE_MSG, CS_CTLBDV, 1);
            return(-1);
        }
        ++time_passed;
    }
    time_passed = 0;
    while ((get_started_jobs(dblib, ORDER_JOBS, InstNum)) > 0)
    {
        waittime(time_to_wait, wait_option);
        ++time_passed;
    }
}

```

```

if ((time_passed % 5) == 0)
{
    get_successful_jobs(dblib, ORDER_JOBS, InstNum);
}
} /* wait for the files to be built. */

return(0);
} /* end of main */

/*-----*/
/*
 * NAME: CRTSTOCKLF
 */
/*
 * PURPOSE: create the stock logical view.
 */
/*
 * Note that Split DB file names have source members with
 * the number number of partitions in the last two
 * characters of the name.
 */
/*-----*/
PGM PARM(&DBLIB &NUMPARTS)

DCL &FILE TYPE(*CHAR) LEN(10) VALUE(STOCK)
DCL &DBLIB TYPE(*CHAR) LEN(10)
DCL &NUMPARTS TYPE(*CHAR) LEN(2)
DCL &FILESRCFL TYPE(*CHAR) LEN(10) VALUE(STOCK)

/*-----*/
MOMMSG(CPF7302 CPF001) EXEC(GOTO ERREXIT)
IF COND(&NUMPARTS *GT '01') THEN( +
DO)
    CHGVAR &SST(&FILESRCFL 6 3) VALUE('000')
    CHGVAR &SST(&FILESRCFL 9 2) VALUE(&NUMPARTS)
ENDDO
/* Create PARTIAL key view. */
CHROBJ OBJ(&DBLIB/&FILE) OBJTYPE(*FILE)
MOMMSG CPF9801 EXEC(CRTLIF FILE(&DBLIB/&FILE) SRCMBR(&FILESRCFL) +
WAITRCD(2) LVLCHK(*NO))

SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFM5G) +
MSGDTA('CRTVIEW completed successfully!') MSGTYPE(*COMP)
GOTO ENDPGM

ERREXIT: SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFM5G) +
MSGDTA('CRTVIEW failed!') MSGTYPE(*ESCAPE)
ENDPGM: ENDPGM

```

## C.38 CRTSTOCKLF.CL:

## C.37 STRTPCCJRN.CL:

```

/*-----*/
/*
 * Name: STRTPCCJRN
 */
/*
 * FUNCTION: start journaling for the TPCC workload.
 */
/*
 * INPUT: DBLIB = where files are located.
 * Creating a Split Database?
 */
/*
 * OUTPUT: JOURNALING STARTED FOR ALL TPCC FILES.
 */
/*-----*/
PGM PARM(&DBLIB)

DCL &DBLIB TYPE(*CHAR) LEN(10)
DCL &NUMPARTS TYPE(*DEC) LEN(2 0)
DCL &JRNNAME TYPE(*CHAR) LEN(10)
DCL &CURRPART TYPE(*DEC) LEN(2 0)
DCL &CURRPARTC TYPE(*CHAR) LEN(2)
DCL &FILENAME TYPE(*CHAR) LEN(10)

MOMMSG MSGID(CPF7030 CPF9812)

SNDPGMMSG MSGID(CPF9897) MSGF(QSYS/QCPFM5G) MSGTYPE(*INFO) +
MSGDTA('STRTPCCJRN: starting journaling of DB files...')

RTVMBRD FILE(&DBLIB/WRHS) NBRDTAMBR(&NUMPARTS)
IF COND(&NUMPARTS *EQ 1) THEN(DO)
    STRJRNPF FILE(&DBLIB/CSMRPF) +
    JRN(&DBLIB/TPCCJRN) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    STRJRNPF FILE(&DBLIB/DSTRCT) +
    JRN(&DBLIB/TPCCJRN) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    STRJRNPF FILE(&DBLIB/HSTRY1) +
    JRN(&DBLIB/TPCCJRN) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    STRJRNPF FILE(&DBLIB/ITEM) +
    JRN(&DBLIB/TPCCJRN) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    STRJRNPF FILE(&DBLIB/NEWORDF) +
    JRN(&DBLIB/TPCCJRN) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    STRJRNPF FILE(&DBLIB/ORDERSPF) +
    JRN(&DBLIB/TPCCJRN) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    STRJRNPF FILE(&DBLIB/ORDLNPF) +
    JRN(&DBLIB/TPCCJRN) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    STRJRNPF FILE(&DBLIB/STOCKPF) +
    JRN(&DBLIB/TPCCJRN) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    STRJRNPF FILE(&DBLIB/WRHS) +
    JRN(&DBLIB/TPCCJRN) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
ENDDO
ELSE DO
    /*-----*/
    /* Split Database */
    /*-----*/
    CHGVAR &CURRPARTC VALUE(1)
    STRLOOP: CHGVAR &CURRPARTC VALUE(&CURRPART)
    CHGVAR &JRNNAME VALUE(STRTPCCJRN *TCAT &CURRPARTC)
    CHGVAR &FILENAME VALUE(CSTMRF *TCAT &CURRPARTC)
    STRJRNPF FILE(&DBLIB/&FILENAME) +
    JRN(&DBLIB/&JRNNAME) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    CHGVAR &FILENAME VALUE(DSTRCT *TCAT &CURRPARTC)
    STRJRNPF FILE(&DBLIB/&FILENAME) +
    JRN(&DBLIB/&JRNNAME) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    CHGVAR &FILENAME VALUE(HSTRY1 *TCAT &CURRPARTC)
    STRJRNPF FILE(&DBLIB/&FILENAME) +
    JRN(&DBLIB/&JRNNAME) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    CHGVAR &FILENAME VALUE(ITEM *TCAT &CURRPARTC)
    STRJRNPF FILE(&DBLIB/&FILENAME) +
    JRN(&DBLIB/&JRNNAME) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    CHGVAR &FILENAME VALUE(NEWORDF *TCAT &CURRPARTC)
    STRJRNPF FILE(&DBLIB/&FILENAME) +
    JRN(&DBLIB/&JRNNAME) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    CHGVAR &FILENAME VALUE(ORDERSPF *TCAT &CURRPARTC)
    STRJRNPF FILE(&DBLIB/&FILENAME) +
    JRN(&DBLIB/&JRNNAME) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    CHGVAR &FILENAME VALUE(ORDLNPF *TCAT &CURRPARTC)
    STRJRNPF FILE(&DBLIB/&FILENAME) +
    JRN(&DBLIB/&JRNNAME) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
    CHGVAR &FILENAME VALUE(STOCKPF *TCAT &CURRPARTC)
    STRJRNPF FILE(&DBLIB/&FILENAME) +
    JRN(&DBLIB/&JRNNAME) IMAGES(*BOTH) OMTJRNE(*OPNCLO)

```

```

/* STRJRNPF FILE(&DBLIB/WRHS1) +
   CHGVAR VAR(&FILENAME) VALUE(WRHS *TCAT &CURRPARTC)
   STRJRNPF FILE(&DBLIB/&FILENAME) +
   JRN(&DBLIB/&JRNNAME) IMAGES(*BOTH) OMTJRNE(*OPNCLO)
   CHGVAR VAR(&CURRPART) VALUE(&CURRPART + 1)
   IF COND(&CURRPART *LE &NUMPARTS) THEN(GOTO STRLOOP)

ENDDD /* end of the else section */

/*-----*/
SNDFPMMSG MSGID(CPF9897) MSGF(QSYS/QCFPMMSG) +
MSGDTA('STRTPCCJRN completed successfully!') MSGTYPE(*COMP)
ENDPGM

```

## C.39 TPCCSPACE.C:

```

/*-----*/
/*
/* File name: TPCCSPACE
/*
/* Purpose: Functions for working with TPCC Build space.
/*
/*-----*/

/*-----*/
/* Start include files.
/*-----*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <quartzus.h>
#include <quaptus.h>
#include <quagen.h>
#include <QSYSINC/MIH/CMPSWP>
#include <mlh/cpybytes.h>
#include <mlh/cpyblap.h>
#include <os4msg.h>
#include <commontpcc.h>
#include <tpccspace.h>
#include <tpccspace1.h>

/* Global values used for all calls */
control_space *control_ptr;
char space_active = 'N'; /* space access flag. */

int setup_space(char *dblib);
job_data *get_job_ptr(int TypeID, int InstNum);
int *get_item_ptr(int InfoType, job_data *job_ptr);

/*-----*/
int dec_space_item(int *item)
{
    int test_value; /* used in compare and swap */
    int new_value; /* used in compare and swap */
    int check; /* used in compare and swap */
    do
    {
        test_value = *item; /* get old value */
        new_value = test_value - 1; /* decrement counter must be positive */
        if (new_value < 0)
        {
            check = -1; /* error flag path */
            return(-1); /* the value is too low */
        }
        check = _CMPSWP(&test_value, item, new_value);
    } while (check == 0); /*update task count */
    return 0;
}

/*-----*/
int inc_space_item(int *item)
{
    int test_value; /* used in compare and swap. */
    int new_value; /* used in compare and swap. */
    int check; /* used in compare and swap. */
    do
    {
        test_value = *item; /* get old task count value */
        new_value = test_value + 1; /* increment task counter */
        check = _CMPSWP(&test_value, item, new_value); /*update task count. */
    } while (check == 0);
    return(0);
}

/*-----*/
int chg_space_item(int *item, int input_value)
{
    int test_value; /* used in compare and swap. */
    int new_value; /* used in compare and swap. */
    int check; /* used in compare and swap. */
    if (*item != input_value)
    {
        do
        {
            test_value = *item; /* get old task count value */
            check = _CMPSWP(&test_value, item, input_value);
        } while (check == 0);
        return(0);
    } else {
        return(1); /* the value was not changed */
    }
} /* end of procedure */

/*-----*/
int dec_available_count ( void )
{
    return(dec_space_item(&control_ptr->available_jobs));
}

/*-----*/
int inc_available_count(void)
{
    if (control_ptr->max_jobs > control_ptr->available_jobs)
    {
        return(inc_space_item(&control_ptr->available_jobs));
    } else {
        char msgtxt[100];
        sprintf(msgtxt,
            "Error: attempted to increment the number of jobs larger than max of %d",
            control_ptr->max_jobs);
        QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
        return(-1);
    }
}

/*-----*/
int get_available_jobs(char *dblib)
{
    if (space_active == 'N')
    {
        if (setup_space(dblib) != 0)
        {
            return (-1);
        }
    }
}

}
return control_ptr->available_jobs;
}

/*-----*/
int get_number_instances(char *dblib)
{
    if (space_active == 'N') /* check space flag. */
    {
        if (setup_space(dblib) != 0)
        {
            return (-1);
        }
    }
    return control_ptr->max_instance;
}

/*-----*/
char get_split(char *dblib)
{
    if (space_active == 'N')
    {
        if (setup_space(dblib) != 0)
        {
            return('U');
        }
    }
    if (control_ptr->max_instance > 1)
    {
        return 'V';
    } else {
        return 'N';
    }
}

/*-----*/
int get_starting_wh(char *dblib, int InstNum)
{
    if (space_active == 'N') /* check space flag. */
    {
        if (setup_space(dblib) != 0)
        {
            return(-1);
        }
    }
    if ((InstNum < 0) || (InstNum > control_ptr->max_instance))
    {
        char msg_txt[120];
        sprintf(msg_txt, "An instance number of %d is out of range 0 - %d",
            InstNum, control_ptr->max_instance - 1);
        QsendMsg(IBM_MSG_FILE, "CPF9897", msg_txt, DIAG_MSG, CS_CUR_PGM, 1);
        return -1; /* if the instance is out of range */
    }
    return control_ptr->working[InstNum].start_whs;
}

/*-----*/
int get_number_whs(char *dblib, int InstNum)
{
    if (space_active == 'N') /* check space flag. */
    {
        if (setup_space(dblib) != 0)
        {
            return(-1);
        }
    }
    if ((InstNum < 0) || (InstNum > control_ptr->max_instance))
    {
        char msg_txt[120]; /* diagnostic message text */
        sprintf ( msg_txt, "An instance number of %d is out of range 0 - %d",
            InstNum, control_ptr->max_instance - 1 );
        QsendMsg(IBM_MSG_FILE, "CPF9897", msg_txt, DIAG_MSG, CS_CUR_PGM, 1);
        return -1; /* if the instance is out of range */
    }
    return control_ptr->working[InstNum].number_whs;
}

/*-----*/
int get_number_of_cpus(char *dblib)
{
    if (space_active == 'N') /* check space flag */
    {
        if (setup_space(dblib) != 0)
        {
            return (-1);
        }
    }
    return control_ptr->actual_cpu_cnt;
}

/*-----*/
int *get_item_ptr(int InfoType, job_data *jobs_ptr)
{
    switch(InfoType)
    {
        case MAX_JOBS_ALLOWED:
            return ( &jobs_ptr->max_allowed );
            break;
        case JOBS_PLANNED:
            return ( &jobs_ptr->planned );
            break;
        case JOBS_RUNNING:
            return ( &jobs_ptr->running );
            break;
        case JOBS_COMPLETED:
            return ( &jobs_ptr->completed );
            break;
        case JOBS_ERROR:
            return ( &jobs_ptr->error );
            break;
        default:
            return (NULL);
            break;
    } /* end of the switch */
}

/* Get a pointer to the job data. */
job_data *get_job_ptr(int TypeID, int InstNum)
{
    if ((InstNum < 0) || (InstNum > control_ptr->max_instance))
    {
        char msg_txt[120]; /* diagnostic message text */
        sprintf ( msg_txt, "An instance number of %d is out of range 0 - %d",
            InstNum, control_ptr->max_instance - 1 );
        QsendMsg(IBM_MSG_FILE, "CPF9897", msg_txt, DIAG_MSG, CS_CUR_PGM, 1);
        return NULL; /* if the instance is out of range */
    }
    switch (TypeID)
    {
        case ORDER_JOBS:
            return( &control_ptr->working[InstNum].orders );
            break;
        case STOCK_JOBS:
            return( &control_ptr->working[InstNum].stock );
            break;
        case HISTORY_JOBS:
            return( &control_ptr->working[InstNum].history );
            break;
        case CUSTOMER_JOBS:

```



```

        return( &control_ptr->working[InstNum].customer );
    break;
    default:
        return (NULL);
    break;
} /* end of the switch */

/* Return the value of the task count */
int GetJobInfo(char *dblib, int TypeID, int InstNum, int InfoType)
{
    int rv; /* return value */
    int * item_ptr; /* pointer to the requested item */
    job_data * task_ptr; /* pointer to the data block */
    if ( space_active == 'N') /* check space flag */
    {
        if ( setup_space( dblib ) != 0 )
        {
            return (-1);
        }
    }
    if (NULL==( jobs_ptr = get_job_ptr ( TypeID, InstNum)))
        return (-1);
    else
    {
        if ( NULL == (item_ptr = get_item_ptr ( InfoType, jobs_ptr )))
        {
            return (-1); /* flag an error */
        }
        else
        {
            return (*item_ptr); /* return the value */
        }
    }
}

/* Return the value of the task count */
int IncJobInfo(char *dblib, int TypeID, int InstNum, int InfoType)
{
    int rv; /* return value */
    int * item_ptr; /* pointer to the item in the space */
    job_data * task_ptr; /* pointer to the data block */
    if ( space_active == 'N') /* check space flag */
    {
        if ( setup_space( dblib ) != 0 )
        {
            return (-1);
        }
    }
    if (NULL==( task_ptr = get_job_ptr ( TypeID, InstNum)))
        return (-1);
    else
    {
        if ( NULL == (item_ptr = get_item_ptr ( InfoType, task_ptr )))
        {
            return (-1); /* flag an error */
        }
        else
        {
            return (inc_space_item(item_ptr)); /* return the value */
        }
    }
}

int DecJobInfo(char *dblib, int TypeID, int InstNum, int InfoType)
{
    int rv; /* return value */
    int * item_ptr; /* pointer to the data item */
    job_data * task_ptr; /* pointer to the data block */
    if ( space_active == 'N') /* check space flag */
    {
        if ( setup_space( dblib ) != 0 )
        {
            return (-1);
        }
    }
    if (NULL==( task_ptr = get_job_ptr ( TypeID, InstNum)))
        return (-1);
    else
    {
        if ( NULL == (item_ptr = get_item_ptr ( InfoType, task_ptr )))
        {
            return (-1); /* flag an error */
        }
        else
        {
            return (dec_space_item(item_ptr)); /* return the value */
        }
    }
}

int setup_space(char *dbl)
{
    error_structure err;
    char space_name[21]; /* user space name */
    int length; /* string length */
    Qus Generic Header 0100 t * space; /* user space header
    CPYBYTES( test_lib, dbl, 10 );
    test_lib[10] = '\0'; /* terminate the string */
    CPYBYTES( space_name, USERSPACE, 10 );
    cpylapl( &space_name[10], 10, test_lib, strlen(test_lib), ' ' );
    /* get the user space name */
    space_name[20] = '\0'; /* terminate the space name
    err.bytes_prov = 16; /* set the number of bytes and error reporting mode
    QUSPTBUS(space_name, &space, &err); /* get the pointer
    if (err.bytes_avail != 0 )
    {
        return(-1);
    }
    control_ptr = (control_space * )space->User_Area; /* get the address
    space_active = 'Y'; /* set the flag to the active state
    return(0);
}

int get_maxjobs_available(char *dbl, int fileID, int InstNum)
{
    return( GetJobInfo( dblib, fileID, InstNum, MAX_JOBS_ALLOWED ));
}

int get_started_jobs ( char *dbl, int fileID, int InstNum )
{
    return( GetJobInfo( dblib, fileID, InstNum, JOBS_RUNNING ));
}

int add_started_jobs ( char *dbl, int fileID, int InstNum )
{
    int test_value; /* used to check setup space */
    test_value = get_available_jobs( dblib );
    if ( test_value == -1) /* error detected */
    {
        return (-1);
    }
    if ( dec_available_count() != 0 )
    {
        return (-1);
    }
    if ( IncJobInfo( dblib, fileID, InstNum, JOBS_RUNNING ) == -1 )
    {
        inc_available_count (); /* update available */
        return (-1);
    }
    else
    {
        return (0);
    }
}

int get_successful_jobs(char *dbl, int fileID, int InstNum)
{
    return( GetJobInfo(dblib, fileID, InstNum, JOBS_COMPLETED));
}

int add_successful_jobs(char *dbl, int fileID, int InstNum)
{
    if ((IncJobInfo(dblib, fileID, InstNum, JOBS_COMPLETED)) == -1)
    {
        return -1;
    }
    DecJobInfo(dblib, fileID, InstNum, JOBS_RUNNING);
    return(inc_available_count());
}

int get_error_jobs(char *dbl, int fileID, int InstNum)
{
    return(GetJobInfo(dblib, fileID, InstNum, JOBS_ERROR));
}

int add_error_jobs(char *dbl, int fileID, int InstNum)
{
    if(( IncJobInfo(dblib, fileID, InstNum, JOBS_ERROR)) == -1)
    {
        return -1;
    }
    return ( inc_available_count ());
}

int get_plan_to_start(char *dbl, int fileID, int InstNum)
{
    return(GetJobInfo(dblib, fileID, InstNum, JOBS_PLANNED));
}

int set_plan_to_start(char *dbl, int fileID, int InstNum, int set_value)
{
    job_data *jobs_ptr; /* pointer to the job data block. */
    if (space_active == 'N')
    {
        if (setup_space(dbl) != 0)
        {
            return(-1);
        }
    }
    jobs_ptr = get_job_ptr( fileID, InstNum);
    if ((jobs_ptr != NULL) && (jobs_ptr->max_allowed >= set_value))
    {
        chg_space_item(&jobs_ptr->planned, set_value);
        return(0);
    }
    else {
        char msg_txt[120]; /* output message for diagnostic */
        char *file_name;
        if (jobs_ptr != NULL)
        {
            if ((file_name = get_DB_filename(fileID)) != NULL)
                sprintf(msg_txt,
                    "max_allowed (%d) for %s jobs is too low for 'planned to start' field of %d",
                    jobs_ptr->max_allowed, file_name, set_value);
            else
                sprintf(msg_txt, "FileID %d is invalid", fileID);
        }
        else {
            sprintf(msg_txt, "Job pointer not found", fileID);
        }
        QsendMsg(IBM_MSG_FILE, "CPF9897", msg_txt, DIAG_MSG, CS_CUR_PGM, 1);
        return(-1);
    }
}

int add_plan_to_start(char *dbl, int fileID, int InstNum)
{
    int current_value; /* test value for the number planned */
    int max_value; /* max for this item */
    current_value = GetJobInfo( dblib, fileID, InstNum, JOBS_PLANNED );
    max_value = GetJobInfo( dblib, fileID, InstNum, MAX_JOBS_ALLOWED );
    if (current_value < max_value)
    {
        return(IncJobInfo(dblib, fileID, InstNum, JOBS_PLANNED));
    }
    else {
        return(-1);
    }
}

int remove_plan_to_start(char *dbl, int fileID, int InstNum)
{
    return(DecJobInfo(dblib, fileID, InstNum, JOBS_PLANNED));
}

/*-----*/
/*
/* Function: get_DB_filename
/* Purpose: convert file id value used to access space into a file
/* name.
/*-----*/
char *get_DB_filename(int file_id)
{
    switch (file_id)
    {
        case ORDER_JOBS:
            return("ORDER");
            break;
        case STOCK_JOBS:
            return("STOCK");
            break;
        case HISTORY_JOBS:
            return("HISTORY");
            break;
        case CUSTOMER_JOBS:
            return("CUSTOMER");
            break;
        default:
            {
                char msgtxt[30];
                sprintf(msgtxt, "Unknown file id of %d", file_id);
                QsendMsg(IBM_MSG_FILE, "CPF9897", msgtxt, DIAG_MSG, CS_CUR_PGM, 1);
                return(NULL);
            }
    } /* end: switch */
    return(NULL);
} /* end: get_DB_filename() */

#ifdef TPCCSPACE_H
#define TPCCSPACE_H
/*-----*/

```

## C.40 TPCCSPACE.H:

```

/*
/* Header file name: TPCCSPACE
/*
/* Purpose: common functions
/*
/*-----*/
/*-----*/
/* DEFINES
/*-----*/

/* Define the job info types uses as the fileID */
/* Note: new file types need to be updated in get_DB_filename() */
#define ORDER_JOBS 0
#define STOCK_JOBS 1
#define HISTORY_JOBS 2
#define CUSTOMER_JOBS 3

/*-----*/
/* Prototypes
/*-----*/
int get_available_jobs(char *dblib);
int get_number_of_cpus(char *dblib);
int get_number_instances(char *dblib);

int get_starting_wh(char *dblib, int InstNo);
int get_number_whs(char *dblib, int InstNo);

int get_maxjobs_available(char *dblib, int fileID, int InstNo);

int get_started_jobs(char *dblib, int fileID, int InstNo);
int add_started_jobs(char *dblib, int fileID, int InstNo);

int get_successful_jobs(char *dblib, int fileID, int InstNo);
int add_successful_jobs(char *dblib, int fileID, int InstNo);

int get_error_jobs(char *dblib, int fileID, int InstNo);
int add_error_jobs(char *dblib, int fileID, int InstNo);

int get_plan_to_start(char *dblib, int fileID, int InstNo);
int set_plan_to_start(char *dblib, int fileID, int InstNo, int set_value);
int add_plan_to_start(char *dblib, int fileID, int InstNo);
int remove_plan_to_start(char *dblib, int fileID, int InstNo);

char get_split(char *dblib);

/*-----*/
/*
/* Function: get_DB_filename
/*
/* Purpose: convert file id used to access space into a file name.
/*
/* Return: file name
/*
/*-----*/
char *get_DB_filename(int file_id);

/* ----->>> End of TPCCSPACE.H <<<----- */
#endif

```

## C.41 TPCCSPACEI.H:

```

#ifndef TPCCSPACEI_H
#define TPCCSPACEI_H
/*-----*/
/*
/* Header file name: TPCCSPACEI
/*
/* Purpose: common space functions
/*
/*-----*/

/*-----*/
/* DEFINES
/*-----*/
#define USERSPACE "TPCCDBJOBS"

#define MAX_JOBS_ALLOWED 10
#define JOBS_PLANNED 11
#define JOBS_RUNNING 12
#define JOBS_COMPLETED 13
#define JOBS_ERROR 14
#define SPLIT_SIZE 2

/* space data definition */
typedef struct
{
    int max_allowed; /* maximum number of jobs allowed limit value. */
    int planned; /* number of jobs planned. */
    int running; /* number of task currently running */
    int completed; /* number of tasks that have completed successfully. */
    int error; /* count of jobs that ended with an error */
} job_data;

typedef struct
{
    int start_whs; /* starting warehouse for this instance */
    int number_whs; /* number of warehouses in this instance. */
    job_data orders; /* data for the orders tasks */
    job_data stock; /* number of stock task running */
    job_data customer; /* number of customer task running */
    job_data history; /* number of history tasks running */
} instance_data;

typedef struct
{
    int max_jobs; /* limit value set at the start */
    int available_jobs; /* number of available processors */
    int actual_cpu_cnt; /* number of cpus available */
    int max_instance; /* maximum number of instances */
    instance_data working[SPLIT_SIZE];
} control_space;

/* ----->>> End of TPCCSPACEI.H <<<----- */
#endif

```

# Appendix D. Application Source Code

## **Program Flow.**



```

)new_order_input; /* output to database server */
char time[8]; /* data is not send to delivery */
)div_input; /* to the server */

/* delivery does not have return from the database server */

typedef _Packed struct {
    short output_qty;
    char output_borg;
    char output_item_name[24];
    decimal (5,2) output_item_price;
    decimal (7,2) output_olamnt;
} output_line_item;

typedef _Packed struct {
    char neword result;
    char oentm[6]; /* zoned amount */
    char clast[16]; /* customer last name */
    char ccredit[2];
    decimal (5,2) cdct;
    int oid; /* order id */
    decimal (5,2) wtax;
    decimal (5,2) dtax;
    decimal (11,2) totamt;
    output_line_item output_line[16];
} new_order_output; /* output from database server */

typedef _Packed struct {
    new_order_input *new_inp;
    new_order_output *new_out;
    char execution_status[24];
} new_order_display; /* output from database server */

typedef _Packed struct {
    char txn_type;
    char jobname[10];
    char payment_type;
    short did;
    short wid;
    int cid;
    short cdid;
    short cwid;
    decimal ( 7, 2) amount;
    char clast[16]; /* customer last name */
    char filler[15];
} payment_input; /* data sent to the database server */

typedef _Packed struct {
    char output_fmt_num[2];
    char time_of_day[8]; /* time of day data */
    char time_filler[6]; /* time filler */
    char waddr1[20];
    char waddr2[20];
    char wcity[20];
    char wstate[2];
    char wzip[9];
    char daddr1[20];
    char daddr2[20];
    char dcity[20];
    char dstate[2];
    char dzip[9];
    int cid;
    char cfirst[16];
    char cinit[2];
    char clast[16];
    char cldate[8]; /* customer last data date */
    char caddr1[20];
    char ccredit[2];
    char caddr2[20];
    decimal(5,2) cdct;
    char ccity[20];
    char cstata[2];
    char czip[9];
    char cphone[16];
    decimal (13,2) cbal;
    decimal (13,2) ccrdlm;
    char cdatt1[50];
    char cdatt2[50];
    char cda3[50];
    char cdatt4[50];
} payment_output; /* data returned from the database server */

typedef _Packed struct {
    payment_input *pay_inp;
    payment_output *pay_out;
} payment_display; /* data returned from the database server */

typedef _Packed struct {
    char txn_type;
    char jobname[10];
    char ordsts_type;
    short did;
    short wid;
    char clast[16];
    char filler[16];
} order_status_input; /* data sent to database server */

typedef _Packed struct {
    short do_olapwh;
    int do_oliid;
    short do_olqty; /* is this correct ?? */
    decimal(7,2) do_olamnt;
    char do_oldiv1[8]; /* zoned data from database server */
} order_status_line_item; /* subarray of order status */

typedef _Packed struct {
    char ordsts_fmt_num[2];
    int oid;
    char cfirst[16];
    char cinit[2];
    char clast[16];
    decimal(13,2) cbal;
    int oid;
    char oettod[8]; /* time of day must be converted */
    char oettm[6]; /* blank field */
    char ocarid[2];
    short olenbr;
    order_status_line_item olineinfo[15];
} order_status_output; /* data returned from the database server */

typedef _Packed struct {
    order_status_input *ordsts_inp;
    order_status_output *ordsts_out;
} order_status_display; /* data displayed */

typedef _Packed struct {
    char txn_type;
    char jobname[10];
    short wid;
    char carrier[2];
}

```

## D.3 DELVRYSRV.C:

```

#include "common.h"

/* tpsvrinit is executed when a server is booted, before it begins
processing requests. It is not necessary to have this function.
Also available is tpsvrdone (not used in this example), which is
called at server shutdown time. */

static int sd, sd2, addr=0, length, port=7000;
int rc=670, remain=0;
static pid_t pid;
error_code err_code;
data_area_data data_area_input;
int mod number=0;
char clnt_ip1[13], clnt_ip2[13];
char svr_ip1[13], svr_ip2[13];

tpsvrinit(int argc, char *argv[])
{
    /* Some compilers warn if argc and argv aren't used. */
    argc = argc;
    argv = argv;

    /* userlog writes to the central TUXEDO message log */
    userlog("Welcome to the tpcsc server");
    err_code.bytes_prov = 0;
    QWCRDTAA ( &data_area_input, 46, modn_dtaara, 1, 1, &err_code );
    mod number = atoi(data_area_input.data_block);
    QWCRDTAA ( &data_area_input, 48, clnt_dtaara, 1, 11, &err_code );
    memcpy( clnt_ip1, &data_area_input.data_block, 11 );
    clnt_ip1[12] = '\0';
    QWCRDTAA ( &data_area_input, 48, clnt_dtaara, 13, 11, &err_code );
    memcpy( clnt_ip2, &data_area_input.data_block, 11 );
    clnt_ip2[12] = '\0';
    QWCRDTAA ( &data_area_input, 48, svr_dtaara, 1, 11, &err_code );
    memcpy( svr_ip1, &data_area_input.data_block, 11 );
    svr_ip1[12] = '\0';
    QWCRDTAA ( &data_area_input, 48, svr_dtaara, 13, 11, &err_code );
    memcpy( svr_ip2, &data_area_input.data_block, 11 );
    svr_ip2[12] = '\0';

    pid = getpid();
    remain = pid % mod number;
    if(remain == 0) {
        addr=inet_addr(clnt_ip1);
        sd=bind(addr, pid);
        addr=inet_addr(svr_ip1);
        sd2=bconnect(sd,addr,port);
    }
    else {
        addr=inet_addr(clnt_ip2);
        sd=bind(addr, pid);
        addr=inet_addr(svr_ip2);
        sd2=bconnect(sd,addr,port);
    }
    return(0);
}

/* This function performs the actual service requested by the client.
Its argument is a structure containing among other things a pointer
to the data buffer, and the length of the data buffer. */

/* DELIVERY */
DELVRY(TPSVCINFO *rqst)
{
    if(send(sd, rqst->data, 230, 0)<0) {

```



```

<td>
<input name="Q11" size=2 MAXLENGTH=2>
</td>
<td colspan=5>
</td>
<tr>
<tr>
<td>
<input name="S12" SIZE=5 MAXLENGTH=5>
</td>
<td>
<input name="I12" size=6 MAXLENGTH=6>
</td>
<td>
<input name="Q12" size=2 MAXLENGTH=2>
</td>
<td colspan=5>
</td>
<tr>
<tr>
<td>
<input name="S13" SIZE=5 MAXLENGTH=5>
</td>
<td>
<input name="I13" size=6 MAXLENGTH=6>
</td>
<td>
<input name="Q13" size=2 MAXLENGTH=2>
</td>
<td colspan=5>
</td>
<tr>
<tr>
<td>
<input name="S14" SIZE=5 MAXLENGTH=5>
</td>
<td>
<input name="I14" size=6 MAXLENGTH=6>
</td>
<td>
<input name="Q14" size=2 MAXLENGTH=2>
</td>
<td colspan=5>
</td>
<tr>
</tr>
</table>
<input TYPE="submit" NAME="CMD" VALUE="Process">
</form>
</body>
</HTML>,
</HTML><HEAD><TITLE>TPC-C Payment</TITLE></HEAD>
<body>
<center><h1>TPC-C Payment</h1></center>
<FORM ACTION="tpcc.dll" METHOD="GET">
<INPUT TYPE="hidden" NAME="PI" VALUE="">
<INPUT TYPE="hidden" NAME="STATUSID" VALUE="0">
<INPUT TYPE="hidden" NAME="ERROR" VALUE="0">
<INPUT TYPE="hidden" NAME="FORMID" VALUE="4">
<INPUT TYPE="hidden" NAME="TERMID" VALUE="">
<INPUT TYPE="hidden" NAME="SYCNID" VALUE="">
<pre>
Warehouse: ?
District: <input name="D_ID" size=2 MAXLENGTH=2><br>
Customer: <input name="C_ID" size=4 MAXLENGTH=4>
Cust-Warehouse: <input name="C_W_ID" SIZE=5 MAXLENGTH=5>
Cust-District: <input name="C_D_ID" size=2 MAXLENGTH=2><br>
Name: <input name="LNAME" size=16 MAXLENGTH=16> Since:<br>
Disc:<br>
Phone:<br>
Amount Paid: $<input name="PAID" size=7 MAXLENGTH=7>
New Cust Balance:<br>
Credit Limit:<br>
Cust-Data:<br>
</pre>
<input TYPE="submit" NAME="CMD" VALUE="Process">
</form>
</body>
</HTML>,
</HTML><HEAD><TITLE>TPC-C Delivery</TITLE></HEAD>
<body>
<center><h1>TPC-C Delivery</h1></center>
<FORM ACTION="tpcc.dll" METHOD="GET">
<INPUT TYPE="hidden" NAME="PI" VALUE="">
<INPUT TYPE="hidden" NAME="STATUSID" VALUE="0">
<INPUT TYPE="hidden" NAME="ERROR" VALUE="0">
<INPUT TYPE="hidden" NAME="FORMID" VALUE="5">
<INPUT TYPE="hidden" NAME="TERMID" VALUE="">
<INPUT TYPE="hidden" NAME="SYCNID" VALUE="">
<pre>
Warehouse: ?
Carrier Number: <input TYPE=TEXT SIZE=2 MAXLENGTH=2 NAME="CARRIER">
Execution Status:
</pre>
<input TYPE="submit" NAME="CMD" VALUE="Process">
</form>
</body>
</HTML>,
</HTML><HEAD><TITLE>TPC-C Order-Status</TITLE></HEAD>
<body>
<center><h1>TPC-C Order-Status</h1></center>
<FORM ACTION="tpcc.dll" METHOD="GET">
<INPUT TYPE="hidden" NAME="PI" VALUE="">
<INPUT TYPE="hidden" NAME="STATUSID" VALUE="0">
<INPUT TYPE="hidden" NAME="ERROR" VALUE="0">
<INPUT TYPE="hidden" NAME="FORMID" VALUE="6">
<INPUT TYPE="hidden" NAME="TERMID" VALUE="">
<INPUT TYPE="hidden" NAME="SYCNID" VALUE="">
<pre>
Warehouse: ?
District: <input name="D_ID" size=2 MAXLENGTH=2><br>
Customer: <input name="C_ID" size=4 MAXLENGTH=4>
Name: <input name="LNAME" size =16 MAXLENGTH=16><br>
Cust-Balance: <br>
Order-Number:
Entry-Date:
Carrier-Number: <br>
</pre>
<table>
<tr>
<th>Supply W</th>
<th>Item Id</th>
<th>Qty</th>
<th>Amount</th>
<th>Delivery-Date</th>
</tr>
</table>
<input TYPE="submit" NAME="CMD" VALUE="Process">
</form>
</body>
</HTML>,
</HTML><HEAD><TITLE>TPC-C Stock Level</TITLE></HEAD>
<body>
<center><h1>TPC-C Stock Level</h1></center>
<FORM ACTION="tpcc.dll" METHOD="GET">
<INPUT TYPE="hidden" NAME="PI" VALUE="">
<INPUT TYPE="hidden" NAME="STATUSID" VALUE="0">
<INPUT TYPE="hidden" NAME="ERROR" VALUE="0">
<INPUT TYPE="hidden" NAME="FORMID" VALUE="7">
<INPUT TYPE="hidden" NAME="TERMID" VALUE="">
<INPUT TYPE="hidden" NAME="SYCNID" VALUE="">
<pre>
Warehouse: ?
Stock Level Threshold: \
Low Stock:
</pre>
<input TYPE="submit" NAME="CMD" VALUE="Process">
</form>
</body>
</HTML>,
</HTML><HEAD><TITLE>Welcome To TPC-C</TITLE></HEAD><BODY><center><h2>
The Warehouse and District you entered are not valid. Please try again.<br>
</center><FORM ACTION="tpcc.dll" METHOD="GET">
<INPUT TYPE="hidden" NAME="FORMID" VALUE="1">
<INPUT TYPE="hidden" NAME="TERMID" VALUE="2">
<INPUT TYPE="hidden" NAME="SYCNID" VALUE="0">
Warehouse ID <input name="w_id" SIZE=5 MAXLENGTH=5><br>
District ID <input name="d_id" SIZE=2 MAXLENGTH=2><br>
</pre>
<input TYPE="submit" NAME="CMD" VALUE="Submit">
</FORM></BODY>
</HTML>,
</HTML><HEAD><TITLE>TPC-C data not found error</TITLE></HEAD><BODY>
<center><h2>TPC-C Benchmark application data not found.</h2><br>
<h3>ERROR 404</h3></center><br>
</BODY>
</HTML>,
</HTML><HEAD><TITLE>TPC-C application exit</TITLE></HEAD><BODY>
<center><h2>TPC-C Benchmark application has ended.</h2><br>
<h3>GOOD-BYE!</h3></center><br>
</BODY>
</HTML>
);
#pragma convert(0)
#define NUM_SVRS 5
void main(int argc, char *argv[]) {
struct inheritance spw_inherit;
char *childArgv[4], shmid_s[20];
char *dummy=NULL, *base;
char buffer[BufferSize];
char warehouse[WarehouseFieldSize], district[DistrictFieldSize];
char *hiddenWid, *hiddenDid, *contLen;
int status, sd, shmid;
int formPort=0;
int header_size; /* header size */
int content_size; /* size of the contents */
int i, j, length;
Config_t *config;
pid_t child;
struct sockaddr_in sin;
int numusers=0, numchildjobs=0;
char numchildjobs_s[5];
int addr=0;
char ip_addr[12];
memcopy(ip_addr, argv[1], 1);
ip_addr[1] = '0';
numusers = atoi(argv[2]);
Qp02InitEnv();
/* Determine amount of shared memory required */
length=sizeof(Config_t)
+((NUM_TRANS*2)*strlen(HEADER))
+16; /* Slop */
for(i=0;i<NUM_FORMS;++i) length+=strlen(FORM[i]);
if((shmid=shmget(
IPC_PRIVATE,
length,
S_IRUSR|S_IWUSR
))<0) {
perror("nshmgmt() failed");
exit(1);
}
if((config=shmat(
shmid,
NULL,
0
))!=NULL) {
perror("nshmat() failed");
exit(1);
}
sprintf(shmid_s,"%i",shmid);
childArgv[0]=shmid_s;
childArgv[1]=shmid_s;
childArgv[3]=NULL;
base=(char *)config+sizeof(Config_t);
/* Create operation forms and spawn the jobs */
for(i=0;i<NUM_FORMS;++i) {
if( i != NOT_FOUND_ERROR )
{
if( i != EXIT )
{
sprintf(base,"%s%s",HEADER,FORM[i]);
header_size = strlen(HEADER);
}
else
{
sprintf(base,"%s%s",EXITHEADER,FORM[i]);
header_size = strlen(EXITHEADER);
}
}
else
{
sprintf(base,"%s%s",ERRORHEADER,FORM[i]);
header_size = strlen(ERRORHEADER);
}
}
/* Locate areas containing content length, and hidden wid/d_id */
contLen=strchr(base,delimiter);
if (( i != WELCOME ) && ( i != NOT_FOUND_ERROR ) && ( i != INPUT_ERROR ) &&
( i != MENU ) && ( i != EXIT ))

```





```

    }
    else
    {
        display_error ();
    }
}
if (writeln==0) printf("writeln is zero\n");
if (write(sd2,writePtr,writeln)<0)
{
    perror("write() failed");
    break; /* error management for write error */
}
if ((form_number == EXIT) || (form_number == NOT_FOUND_ERROR))
    break; /* exit the receive loop */
} /* end of the recv else statement */
} /* end of for loop */
close(sd2);

/* Free Buffers & Detach from System/T */
tpterm();
shmdt (config);
iconv_close(toIAS);
iconv_close(toEBCDIC);
close(sd);
exit(0);
}

void process_start ( void )
{
    writePtr=config->form[WELCOME].addr;
    writeLen=config->form[WELCOME].size;
}

void display_error ( void )
{
    form_number = NOT_FOUND_ERROR;
    writePtr=config->form[NOT_FOUND_ERROR].addr;
    writeLen=config->form[NOT_FOUND_ERROR].size;
}

void process_normal ( char * temp )
{
    char form_selector;
    char * marker; /* Location of the formid string */
    char input_error='Y'; /* input error flag */
    char input_error_1='Y'; /* scan warehouse input error flag */
    char input_error_2='Y'; /* scan district input error flag */
    int i;

    /* Find the text string FORMID= and then get the form number */
    if (marker=strstr(temp,FORMID)!=NULL)
    {
        /* advance pointer to the end of the formid string */
        temp = marker + strlen(FORMID);
        form_number = *temp - ONE; /* get the form number then */
    }
    else /* FORMID not found */
    {
        form_number = NOT_FOUND_ERROR; /* set form number to error form */
        i = form_number; /* set form index */
        /* setup error form and return */
        writePtr=config->form[i].addr;
        writeLen=config->form[i].size;
        hiddenWid=config->form[i].hiddenWid; /* set the warehouse field */
        hiddenDid=config->form[i].hiddenDid; /* set the district field */
        wid=config->form[i].wid; /* set the warehouse field */
        return;
    }
    /* check the form number */
    if (form_number == WELCOME) /* welcome form */
    {
        /* check for the warehouse id text */
        if ((marker=strstr(temp,WAREHOUSEID))!=NULL)
        {
            temp=marker+strlen(WAREHOUSEID)-1; /* go just to the = sign */
            input_error = 'Y';
            /* scan for warehouse ID */
            input_error_1 = scan_strings (temp,warehouse,5,ampersand);
            {
                if ((temp=strchr(temp,parameter))!=NULL)
                /* advance pointer to start of district data */
                {
                    /* scan for district ID */
                    input_error_2=scan_strings (temp,district,3,ampersand);
                }
            }
            if ((input_error_1 == 'N') && (input_error_2 == 'N'))
                input_error = 'N'; /* set the input error flag */
            if (input_error == 'N') /* district and warehouse are correct */
            {
                form_number = MENU; /* set up the menu display */
                i = form_number; /* set form index */
                /* setup menu form and return */
                writePtr=config->form[i].addr;
                writeLen=config->form[i].size;
                hiddenWid=config->form[i].hiddenWid; /* set the warehouse field */
                hiddenDid=config->form[i].hiddenDid; /* set the district field */
            }
            else
            {
                form_number = INPUT_ERROR;
                /* setup input error screen form and return */
                i = form_number;
            } /* end of inner else */
        }
        else /* warehouse id not found */
        {
            form_number = NOT_FOUND_ERROR; /* set form number to error screen */
            i = form_number;
        }
        writePtr=config->form[i].addr;
        writeLen=config->form[i].size;
        hiddenWid=config->form[i].hiddenWid;
        hiddenDid=config->form[i].hiddenDid;
        wid=config->form[i].wid; /* set the warehouse field */
    } /* end of check for Welcome FORM */
}
else /* The form is a menu selection */
{
    /* check for the TERMID text */
    if (marker=strstr(temp,TERMID)!=NULL)
    {
        temp = marker + strlen(TERMID); /* advance the pointer */
        for (i=0;
            temp[i]&&temp[i]!=INPUT_BLANK&&temp[i]!=ampersand&&
            (i<WarehouseFieldSize);
            ++i)
            warehouse[i]=temp[i];
        /* check SYNCID text */
        if (marker=strstr(temp,SYNCID)!=NULL)
        {
            temp = marker + strlen(SYNCID); /* go to the syncid string */
            /* advance pointer to syncid */
            for (i=0;
                temp[i]&&temp[i]!=INPUT_BLANK&&temp[i]!=ampersand&&
                (i<DistrictFieldSize);
                ++i)
                district[i]=temp[i];
            /* check for the &CMD... text */
            if (marker=strstr(temp,CMDSTRING)!=NULL)
            {
                /* advance the pointer to the end of the command string */
                temp = marker + strlen(CMDSTRING);
            }
        }
    }
}
}

form_selector = *temp;
/* form_selector is the first character of the command */
}
else
{
    form_selector = ASCII_N; /* default to new order */
}
/* check the form number */
if (form_selector == ASCII_N)
    i = NEW_ORDER; /* set up the new order form */
else
{
    if (form_selector == ASCII_P)
        i = PAYMENT;
        date_loc=config->form[i].date_location;
        copy_date();
    }
    else
    {
        if (form_selector == ASCII_O)
            i = ORDER_STATUS;
        else
        {
            if (form_selector == ASCII_D)
                i = DELIVER;
            else
            {
                if (form_selector == ASCII_S)
                    i = STOCK_LEVEL;
                else
                {
                    if (form_selector == ASCII_E)
                        i = EXIT;
                    else
                        i = NEW_ORDER; /* send new order */
                }
            }
        }
        /* end else if (form_selector == ASCII_S) */
        /* end else if (form_selector == ASCII_D) */
        /* end else if (form_selector == ASCII_O) */
        /* end else if (form_selector == ASCII_P) */
        /* end else if (form_selector == ASCII_N) */
    }
    form_number = i; /* set the form number */
    writePtr=config->form[i].addr;
    writeLen=config->form[i].size;
    /* Set local ptrs to hidden vars */
    hiddenWid=config->form[i].hiddenWid;
    hiddenDid=config->form[i].hiddenDid;
    wid=config->form[i].wid; /* set the warehouse field */
} /* end of if SYNCID found */
else
{
    form_number = NOT_FOUND_ERROR; /* set the form number */
    writePtr=config->form[NOT_FOUND_ERROR].addr;
    writeLen=config->form[NOT_FOUND_ERROR].size;
} /* end of if TERMID found */
else
{
    form_number = NOT_FOUND_ERROR; /* set the form number */
    writePtr=config->form[NOT_FOUND_ERROR].addr;
    writeLen=config->form[NOT_FOUND_ERROR].size;
} /* end of else form id not equal WELCOME form */
}

void copy_date(void)
/* Poke date into form when required */
{
    int i; /* for loop counter */
    _MI_Time time_of_day; /* time of day */
    matloc(time_of_day); /* get the time of day */
    time_of_day[7]='B';
    getdftime (time_of_day, cl_time, 'm'); /* get time string */
    cl_time[19]='\0'; /* terminate the string */
    for (i=0;i<19; ++i)
    {
        switch (cl_time[i])
        {
            case '-':
                date_loc[i] = DASH;
                break;
            case ':':
                date_loc[i] = COLON;
                break;
            case ' ':
                date_loc[i] = BLANK;
                break;
            default:
                date_loc[i]=(cl_time[i]-'0')*ZERO;
                break;
        }
    } /* end of switch */
} /* end of for loop */
} /* end of copy date */

/* checks if the input data is valid */
/* ***** */
char parse_input ( char * input, char ** new_ptr )
{
    int index;
    int test_string_size; /* set longer than max string */
    char * test_string; /* string to test */
    char * data_string; /* pointer to new test string */
    char mode = 'N';
    int normal_len;
    int welcome_len;
    normal_len = strlen (TEST_DATA) -1; /* index of length */
    welcome_len = strlen (WELCOME_DATA) -1; /* index of length */
    for (index=0; index<5; index++)
    {
        if (input[index] != GET_STRING[index])
        {
            *new_ptr = NULL; /* set a null pointer for return */
            return ('Y');
        }
    }
    if (input[5] == TEST_DATA[0])
    {
        test_string = (unsigned char *) &TEST_DATA[0];
        test_string_size = strlen (TEST_DATA);
        mode = 'N'; /* set the processing mode */
    }
    else
    {
        if (input[5] == WELCOME_DATA[0])
        {
            test_string = (unsigned char *) &WELCOME_DATA[0];
            test_string_size = strlen (WELCOME_DATA);
            mode = 'W'; /* set the processing mode */
        }
        else
        {
            *new_ptr = NULL; /* set a null pointer for return */
            return ('Y');
        }
    }
    data_string = (unsigned char *) &input[5]; /* get starting point */
    for (index=1; index<test_string_size; index++) /* index 0 has been tested */
    {
        if (data_string[index] != test_string[index])
        {
            *new_ptr = NULL; /* set a null pointer for return */
            return ('Y'); /* error detected */
        }
    }
}

```

```

        }
    } /* end of the for loop */
    *new_ptr = data_string + test_string_size;
    return (mode);
}

/*****
 * checks if the input data is numeric */
char scan_strings ( char ** current_ptr, char * output_loc, int size,
char terminator )
{
    int blank_count=0; /* blank character counter */
    char * start_ptr; /* pointer to the active string start */
    char * output_ptr; /* pointer to current string location */
    int scan_count; /* for loop counter */
    char scan_error='N'; /* scan error flag */
    start_ptr = *current_ptr; /* set the starting position pointer */
    output_ptr = output_loc; /* saved location */
    for( scan_count = 0; scan_count<size; scan_count++)
    {
        *current_ptr=*current_ptr+1; /* next character point */
        if (((**current_ptr) >= ZERO) && (**current_ptr)<= NINE)
        {
            if ( blank_count == 0 ) /* not after a trailing blank */
            {
                *output_ptr = **current_ptr; /* save the data */
                output_ptr = output_ptr+1; /* go to the next location */
            }
            else
            {
                scan_error = 'Y'; /* this is an input error */
                scan_count = size; /* you are done scanning */
            }
        }
        else /* end of if numeric */
        {
            if ( (**current_ptr) == terminator )
            {
                scan_count = size; /* you are done scanning */
                if ((**current_ptr)==start_ptr) /* no character were processed */
                {
                    scan_error = 'Y';
                }
            }
            else /* character is not a terminator */
            {
                if ( (**current_ptr) == INPUT_BLANK ) /* check for blanks */
                {
                    if ((**current_ptr)==start_ptr) /* leading blank */
                    {
                        start_ptr = start_ptr+1; /* skip this character */
                    }
                    else
                    {
                        blank_count=blank_count+1; /* count the trailing input blank */
                    }
                }
                else
                {
                    scan_error = 'Y'; /* this is an input error */
                    scan_count = size; /* you are done scanning */
                }
            }
        }
    } /* end of else not numeric */
} /* end of the for loop */
if (**current_ptr)==start_ptr /* blank input */
{
    scan_error = 'Y';
}
return scan_error;
}

/*****
 * Initialize the environment including tuxedo */
void initTux ( void )
{
    toIAS=iconv_open(ToIAS,FromEBCDIC);
    if(toIAS.return_value<0){
        perror("\niconv_open() failed");
        exit(1);
    }
    toEBCDIC=iconv_open(ToEBCDIC,FromIAS);
    if(toEBCDIC.return_value<0){
        perror("\niconv_open() failed");
        exit(1);
    }
    /* set up the tuxedo environment */
    putenv("TUXDIR=/qopenysa/tuxsdsk");
    putenv("APPDIR=/home/tpccvclnt");
    putenv("TUXCONFIG=/home/tpccvclnt/tuxconfig");

    /* Attach to System/T as a Client Process */
    if (tpinitt((TPINIT *) NULL) == -1) {
        (void) fprintf(stderr, "Tpinitt failed\n");
        exit(1);
    }

    /* allocate space for buffers */
    if((nd.new_inp=(new_order_input *) tmalloc("CARRAY", NULL, 670))!=NULL) {
        (void) fprintf(stderr, "Can't allocate buffer for neword\n");
        exit(1);
    }
    if((pd.pay_inp=(payment_input *) tmalloc("CARRAY", NULL, 670))!=NULL) {
        (void) fprintf(stderr, "Can't allocate buffer for paymnt\n");
        exit(1);
    }
    if((di=(div_input *) tmalloc("CARRAY", NULL, 670))!=NULL) {
        (void) fprintf(stderr, "Can't allocate buffer for delvry\n");
        exit(1);
    }
    if((od.ordsts_inp=(order_status_input *) tmalloc("CARRAY", NULL, 670))!=NULL) {
        (void) fprintf(stderr, "Can't allocate buffer for ordsts\n");
        exit(1);
    }
    if((s_d_stk_inp=(stock_input *) tmalloc("CARRAY", NULL, 670))!=NULL) {
        (void) fprintf(stderr, "Can't allocate buffer for stklvl\n");
        exit(1);
    }

    if((nd.new_out=(new_order_output *) tmalloc("CARRAY", NULL, 670))!=NULL) {
        (void) fprintf(stderr, "Can't allocate buffer for neword\n");
        exit(1);
    }
    if((pd.pay_out=(payment_output *) tmalloc("CARRAY", NULL, 670))!=NULL) {
        (void) fprintf(stderr, "Can't allocate buffer for paymnt\n");
        exit(1);
    }
    if((od.ordsts_out=(order_status_output *) tmalloc("CARRAY", NULL, 670))!=NULL) {
        (void) fprintf(stderr, "Can't allocate buffer for ordsts\n");
        exit(1);
    }
    if((s_d_stk_out=(stock_output *) tmalloc("CARRAY", NULL, 670))!=NULL) {
        (void) fprintf(stderr, "Can't allocate buffer for stklvl\n");
        exit(1);
    }
}

/*****
 * Processes the transaction input data */
void process_input ( char * input_buffer )
{
    int formId; /* form ID from the input */
    int formIdPost; /* modified form input */
    int i;
    int contentSize;
    unsigned char *fromPtr, *toPtr;
    Config_t *config;
    int count;
    int counter; /* for loop counter */
    char *work; /* buffer to get the warehouse number */
    unsigned int length, inSize, outSize;
    char argument[BufferSize]; /* location after converting */
    char *temp; /* inStr; */
    long send_len = 670, *recv_len;
    int ret;
    /* MI Time time of day; /* time of day */
    char cmd[20];

    recv_len = &send_len;
    /* Find the form id number and save as the FormId */
    if ((work=strstr(input_buffer, FORMID))==NULL)
    {
        printf("\nMissing FORMID");
        formId = NUM_FORMS;
    }
    else
    {
        work = work + sizeof(FORMID) - 1;
        formId = *work - ONE; /* get the form id number */
    }
    /* Find start of argument area */
    if ((inStr=strstr(input_buffer, TERMD))!=NULL)
        printf("\nMissing TERMD");
    else if ((temp= strchr(inStr, BLANK))!=NULL) /* locate next blank */
        printf("\nMissing Blank");
    else {
        inSize=outSize=(temp-inStr);
        temp=argument;
        temp[inSize]=0; /* Insert string terminator */

        /* Convert arguments to EBDCIC */
        if (iconv(toEBDCIC,inStr,&inSize,&temp,&outSize)<0){
            perror("\niconv() failed");
            exit(1);
        }
        formIdPost = formId;
        work = argument + sizeof(termd) - 1;
        memset ( warehouse, '\0', WarehouseFieldSize + 1 );
        for ( counter=0; counter<WarehouseFieldSize;&&
            (work[counter]!='&')&&
            (work[counter]!='+')&&
            counter++)
            warehouse[counter] = work[ counter ];
        if ((work = strstr ( work, "SYNCID=" ))!=NULL)
        {
            printf ( "District not found\n" );
        }
        else
        {
            memset ( district, '\0', DistrictFieldSize + 1 );
            work = work + 7;
            for ( counter=0; counter<DistrictFieldSize;&&
                (work[counter]!='&')&&
                (work[counter]!='+')&&
                counter++)
                district[counter] = work[ counter ];
        }
    }
    /* process transaction based on type */
    switch(formId) {
        case NEW_ORDER:
            count = scan_new_order( argument, nd.new_inp);
            if ( count == 0 ) /* new order data valid */
            {
                ret=tpcall("NEWORDER", (char *)nd.new_inp, send_len,
                    (char **) &nd.new_out, (long *) recv_len, TPNOTIME);

                if(ret == -1) {
                    (void) fprintf(stderr, "Can't send request to NEWORDER \n");
                    (void) fprintf(stderr, "Tperno = %d, ret=%d\n", tperno, ret);
                    tpterm();
                    exit(1);
                }
            }
            else /* invalid new order data */
                formIdPost = set_error_report( count );

            /* Respond to the client */
            postResult(&nd, warehouse, district, formIdPost, scratchBuffer);

            break;

        case PAYMENT:
            count = scan_payment ( argument, pd.pay_inp );
            if(count == 0) /* payment data valid */
            {
                ret=tpcall("PAYMNT", (char *)pd.pay_inp, send_len,
                    (char **) &pd.pay_out, (long *) recv_len, TPNOTIME);

                if(ret == -1) {
                    (void) fprintf(stderr, "Can't send request to service PAYMENT\n");
                    (void) fprintf(stderr, "Tperno = %d\n", tperno);
                    tpterm();
                    exit(1);
                }
            }
            else /* payment data invalid */
                formIdPost = set_error_report( count );

            /* Respond to the client */
            postResult(&pd, warehouse, district, formIdPost, scratchBuffer);

            break;

        case ORDER_STATUS:
            count=scan_order_status ( argument, od.ordsts_inp );
            if (count == 0) /* order status data valid */
            {
                ret=tpcall("ORDSTS", (char *)od.ordsts_inp, send_len,
                    (char **) &od.ordsts_out, (long *) recv_len, TPNOTIME);

                if(ret == -1) {
                    (void) fprintf(stderr, "Can't send request to service ORDSTS\n");
                    (void) fprintf(stderr, "Tperno = %d\n", tperno);
                    tpterm();
                    exit(1);
                }
            }
            else /* order status data invalid */
                formIdPost = set_error_report( count );

            postResult(&od, warehouse, district, formIdPost, scratchBuffer);

            break;
    }
}

```

```

case DELIVERY:
count= scan_delivery( argument, di );
if (count == 0) { /* delivery data valid */
/* do a get time here */
matted ( time_of_day );
time_of_day[7]='B'; /* set ending flag */
getdtmstr (time_of_day, di->time, 'h');

ret=tpacall("DELIVRY",(char *)di,send_len, TPNOREPLY);

if (ret == -1) {
(void) fprintf(stderr, "Can't send request to service DELIVERY\n");
(void) fprintf(stderr, "Tprerno = %d\n", tprerno);
tpterm();
exit(1);
}
} /* delivery data invalid */
else formIdPost = set_error_report( count );
postResult(di, warehouse, district, formIdPost, scratchBuffer);
break;

case STOCK_LEVEL:
count= scan_stock_level ( argument, s_d.stk_inp );
if (count == 0) { /* stock level data valid */
ret=tpacall("STKLVL",(char *)s_d.stk_inp, send_len,
(char **) &s_d.stk_out, (long *) recv_len, TPNOTIME);

if (ret == -1) {
(void) fprintf(stderr, "Can't send request to service STKLVL\n");
(void) fprintf(stderr, "Tprerno = %d\n", tprerno);
tpterm();
exit(1);
}
} /* stock level data invalid */
else formIdPost = set_error_report( count );
postResult(&s_d, warehouse, district, formIdPost, scratchBuffer);
break;

case NUM_FORMS: /* invalid form number */
formIdPost = SIGNON_ERROR;
postResult(&nd, warehouse, district, formIdPost, scratchBuffer);
break;

default: /* return an error if no match above */
printf("Invalid form ID = %d", formId);
return;

} /* end of switch */
inSize = outSize = contentSize = strlen(scratchBuffer);
/* Copy header into output buffer */
sprintf(buffer,"%s",HEADER);
length = strlen(buffer);
/* Convert EBCDIC result to ASCII */
toPtr = buffer + length;
length += inSize;
fromPtr = scratchBuffer;

if (iconv(toIAS,&fromPtr,&inSize,&toPtr,&outSize)<0)
error("unicnv() failed");
exit(1);
}

/* Poke in response length */
binToIAS(contentSize, strchr(buffer,delimiter));
writePtr = buffer; /* set up the output function */
writeln = length; /* set the output length */
}

int set_error_report( int status )
{
if ( status == 100 )
{
return SIGNON_ERROR;
}
else
{
return DATA_ERROR;
}
}

```

## D.7 GETDTTMSTR.C:

```

.....
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <math.h>
#include <ctype.h>
#include <decimal.h>
#include <recio.h>
#include <errno.h>
#include <time.h>
#include <xxfcbk.h>
#include <xxcvc.h>
#include <nicomput.h>
#include <QSYSINC/MIH/MIDTTM>
#include <qwcrsvai.h>
#include <qsysinc/mih/cvtch>
.....
/* Function Prototypes */
.....
void MI_Template ( _INST_Template_Tl **, _DDAT_T **, _DDAT_T **, char );
.....
/* Date and time coversion pointers */
.....
_INST_Template_Tl *inst_t1; /* MI template */
_DDAT_T *ddat_t1, *ddat_t2; /* DDAT pointers */
/* Internal format for end of Gregorian timeline: Jan. 1, 0001 */
#define GREGORIAN_TIMELINE_START 1721424
/* Internal format for end of Gregorian timeline: Jan. 1, 10000 */
#define GREGORIAN_TIMELINE_END 5373485
typedef struct {
int bytes_prov;
int bytes_avail;
char except_id[7];
char reserved[1];
char except_dta[50];
} error_code; /* error code structure used by QUSPTRUS */
.....
/* Start */
.....
getdttmstr(char * timein, char * timestamp, char type)
{
char tmstamp[27];
static char first = 'Y';

if ( timein[7] == '0' )
{
switch ( type )

```

```

{
case 'd':
memset ( timestamp, '0', 8 ); /* set the null date */
break;
case 't':
memset ( timestamp, '0', 6 ); /* set the null date */
break;
case 'h':
memset ( timestamp, '0', 8 ); /* set the null date */
break;
case 'b':
memset ( timestamp, '0', 14 ); /* set the null date */
break;
case 'l':
memcpy( &timestamp[0], /* copy hour min sec */
"0000-00-00.00.00.00.0000000", 27 );
break;
case 'm':
memcpy( &timestamp[0], /* copy hour min sec */
"0000-00-00 00:00:00.0000000", 27 ); /* set null string */
break;
case 'n':
memcpy( &timestamp[0], /* copy hour min sec */
"0000/00/00 00:00:00.0000000", 27 ); /* set null string */
break;
default:
break;
} /* end of switch */
}
else
{
MI_Template(&inst_t1,&ddat_t1,&ddat_t2, 'P');
timein[7] = 0; /* clear the era flag */
cvtts(tmstamp, timein, inst_t1);

switch ( type )
{
case 'd':
memcpy( &timestamp[4], &tmstamp[0], 4 ); /* copy the year */
memcpy( &timestamp[0], &tmstamp[5], 2 ); /* copy month */
memcpy( &timestamp[2], &tmstamp[8], 2 ); /* copy day */
break;
case 't':
case 'h':
memcpy( &timestamp[0], &tmstamp[11], 2 ); /* copy hour */
memcpy( &timestamp[2], &tmstamp[14], 2 ); /* copy minutes */
memcpy( &timestamp[4], &tmstamp[17], 2 ); /* copy second */
if ( type == 'h' )
memcpy( &timestamp[6], &tmstamp[20], 2 ); /* copy second */
break;
case 'b':
memcpy( &timestamp[4], &tmstamp[0], 4 ); /* copy the year */
memcpy( &timestamp[0], &tmstamp[5], 2 ); /* copy month */
memcpy( &timestamp[2], &tmstamp[8], 2 ); /* copy days */
memcpy( &timestamp[11], &tmstamp[11], 2 ); /* copy hour */
memcpy( &timestamp[10], &tmstamp[14], 2 ); /* copy minutes */
memcpy( &timestamp[12], &tmstamp[17], 2 ); /* copy second */
break;
case 'l':
memcpy( &timestamp[0], &tmstamp[0], 27 ); /* copy hour min sec */
break;
case 'm':
case 'n':
memcpy( &timestamp[0], &tmstamp[5], 2 ); /* get month */
memcpy( &timestamp[3], &tmstamp[8], 2 ); /* get day */
memcpy( &timestamp[6], &tmstamp[0], 4 ); /* get year */
memcpy( &timestamp[11], &tmstamp[11], 2 ); /* get hours */
memcpy( &timestamp[14], &tmstamp[14], 2 ); /* get minutes */
memcpy( &timestamp[17], &tmstamp[17], 2 ); /* get seconds */
timestamp[13] = ':'; /* set the time separator */
timestamp[13] = ':'; /* set the time separator */
timestamp[16] = ':'; /* set the time separator */
if ( type == 'n' )
{
timestamp[2] = '/'; /* set the date separator */
timestamp[5] = '/'; /* set the date separator */
}
else
{
timestamp[2] = '-'; /* set the date separator */
timestamp[5] = '-'; /* set the date separator */
}
break;
default:
break;
} /* end of switch */
} /* end of else */
}

void maketime( char * date, char *time, char * flag,
char * output )
{
char working[27]; /* the working 26 character data input */
char cvt_type = 'I';
char tmp1_time[9] = {0x7D,0x66,0x27,0x9F,0xE4,0xF5,0x00,0x00};
char tmstamp[27] = "1998-07-24-13.16.55.000000";

/* format YYYY-MM-DDHH.MI.SS.999999 */
/* location 01234567890123456789012345 */
/* tens 0 1 2 */
memcpy ( &working[0], &date[4], 4 ); /* get the year */
memcpy ( &working[5], &date[0], 2 ); /* get the month */
memcpy ( &working[8], &date[2], 2 ); /* get the day */
memcpy ( &working[11], &time[0], 2 ); /* get the hour */
memcpy ( &working[14], &time[2], 2 ); /* get the minute */
memcpy ( &working[17], &time[4], 2 ); /* get the second */
working[4] = '-'; /* set the dashes */
working[7] = '-'; /* set the dashes */
working[10] = '-'; /* set the dashes */
working[13] = ':'; /* set the time separator */
working[16] = ':'; /* set the porid */
working[19] = ':'; /* set the porid */
working[26] = '\0'; /* terminate the string */
if ( flag[0] == 'H' )
{
memset ( &working[20], '9', 6 ); /* set the value to the maxime */
}
else
{
memset ( &working[20], '0', 6 ); /* set the value to the minime */
}

MI_Template(&inst_t1,&ddat_t1,&ddat_t2, cvt_type);
/* print ( "The time stamp was %s\n", tmstamp ); */
printf ( "The time is %s\n", working );
cvtts( output, working, inst_t1 );
}
.....
..... MI_Template .....
/* Function: MI_Template */
/*
/* Description: Allocate and fill in MI instruction template used
/* in date and time conversions.
.....
void MI_Template ( _INST_Template_Tl **inst_t, _DDAT_T **ddat1,
_DDAT_T **ddat2, char cvt_type)
{
static char first = 'Y';
.....
/* Date and Time conversion (mi template) */
.....
.....
_Era_Table_T *era_t1, *era_t2; /* Era table pointers */
_Calendar_Table_T *cal_t1, *cal_t2; /* Calendar table pointers */
.....
int DDAT_Length, Calendar_Offset;
int DDAT_Size, Template_Size;

```

# D.8 IMPORT.H:

```
int          DDAT_Offset1, DDAT_Offset2;
char         inbuf[80]; /* buffer for input century */
char         value[1][10]={'QCENTURY'};
int          offset, offset_2; /* offset to century */
int          *offset_ptr;
error_code  err_code; /* error code variable */
.....
if (first == 'Y') /* First call ? */
( first = 'N'; /* Yes, allocate template etc. */

DDAT_Length = sizeof(DDAT_T)+2*(sizeof(Calendar_Table_T)
-sizeof(short) + sizeof(Era_Table_T)-1);

Calendar_Offset = offsetof(DDAT_T, Tables)+sizeof(Era_Table_T);

DDAT_Size = 2*DDAT_Length +
2*(sizeof(int)) + /* DDAT_Offset */
10 /* reserved4 */
sizeof(short) + /* Num_DDATS */
sizeof(int); /* DDAT_Size */

Template_Size = 2*DDAT_Length+offsetof(INST_Template_T1,DDAT)
+ sizeof(int); /* DDAT1 offset */
DDAT_Offset1 = (offsetof(INST_Template_T1,DDAT) -
offsetof(INST_Template_T1,DDAT_Size)) + sizeof(int);
/* DDAT2 offset */
DDAT_Offset2 = (offsetof(INST_Template_T1,DDAT) -
offsetof(INST_Template_T1,DDAT_Size))+sizeof(int)+DDAT_Length;

*inst_t = (INST_Template_T1 *)malloc(Template_Size);
/* Fill in Instruction Template */

memset(*inst_t,'\0', sizeof(INST_Template_T1));
(* inst_t)->Template_Size = Template_Size; /* Instruction template size */
(* inst_t)->DDAT_1 = 1; /* Operand 1 DDAT */
(* inst_t)->DDAT_2 = 2; /* Operand 2 DDAT */
(* inst_t)->reserved1[0] = 0; /* set reserved1 */
(* inst_t)->reserved1[1] = 0; /* set reserved1 */
(* inst_t)->Length_1 = 26; /* Result timestamp */
(* inst_t)->Length_2 = 8; /* input mitime */
(* inst_t)->reserved2[0] = 0; /* set reserved2 */
(* inst_t)->reserved2[1] = 0; /* set reserved2 */
(* inst_t)->DDAT_Size = DDAT_Size; /* Size of DDAT list */
(* inst_t)->Num_DDATS = 2; /* Number of DDATS */
memset((* inst_t)->reserved3, 0, 22); /* set the reserved values */
memset((* inst_t)->reserved4, 0, 10); /* set the reserved values */
(* inst_t)->DDAT_Offset[0] = DDAT_Offset1; /* DDAT offset1 */
/* (* inst_t)->DDAT_Offset[1] = DDAT_Offset2; /* DDAT offset2 */

*ddat1 = (DDAT_T *)(&(* inst_t)->DDAT_Offset2);
era_t1 = (Era_Table_T *)(&(* ddat1)->Tables); /* set era table ptr */
cal_t1 = (Calendar_Table_T *)((char *)*ddat1 + Calendar_Offset);
*ddat2 = (DDAT_T *)((char *)*ddat1 + DDAT_Length);
era_t2 = (Era_Table_T *)(&(* ddat2)->Tables);
cal_t2 = (Calendar_Table_T *)((char *)*ddat2 + Calendar_Offset);

/* Fill in DDAT1 */

(* ddat1)->DDAT_Length = DDAT_Length; /* Set DDAT length */
(* ddat1)->Format_Code = SAA_TIMESTAMP; /* Set result format code */
(* ddat1)->Hour_Zone = 24; /* set target hours */
(* ddat1)->Min_Zone = 60; /* set target minutes */
(* ddat1)->Calendar_Offset=Calendar_Offset; /* Calendar offset */
memset((* ddat1)->reserved, 0, 6); /* set the reserved values */
/* Fill in Era Table for DDAT1 */

era_t1->Num_Elems = 1; /* Set number of era elements */
era_t1->Element[0].Origin_Date = GREGORIAN_TIMELINE_START;
era_t1->Element[0].Era_Name[0] = 'A'; /* Set era name, to .... */
era_t1->Element[0].Era_Name[1] = 'D'; /* AD */
memset(era_t1->Element[0].reserved, 0, 12); /* set the reserved v alues */

/* Fill in Calendar Table for DDAT1 */

cal_t1->Num_Elems = 2; /* Set Calendar elements */
cal_t1->Element->Effect_Date = GREGORIAN_TIMELINE_START;
cal_t1->Element->Type = 0x0001; /* Set element type */
memset(cal_t1->Element->reserved,'0',10); /* Initialize reserved */
(cal_t1->Element->Effect_Date = GREGORIAN_TIMELINE_END;
(cal_t1->Element->Type = 0;
memset((cal_t1->Element->reserved,'0',10);

/* Fill in DDAT2 */

(* ddat2)->DDAT_Length = DDAT_Length; /* Set DDAT length */
(* ddat2)->Format_Code = IMP1_CLOCK; /* Set source format code */
(* ddat2)->Hour_Zone = 24; /* set target hours */
(* ddat2)->Min_Zone = 60; /* set target minutes */
err_code.bytes_prov = 0; /* set error reporting mode */
QWCRVAL ( inbuffer, 80, 1, value, &err_code );
/* set current century */
offset_ptr = (int *)inbuffer[4]; /* get the offset */
offset = *offset_ptr; /* get the offset value */
offset_2 = offset + 16; /* get the true offset */
/* if ( inbuffer[offset_2]=='1' )
/* (* ddat2)->Cur_Century = 1; set target current century */
/* else */
/* (* ddat2)->Cur_Century = 0; /* set target current century */
/* (* ddat2)->Century_Div = 0; /* set target centry division */
/* (* ddat2)->Calendar_Offset=Calendar_Offset; /* Calendar offset */
/* (* ddat2)->Calendar_Offset = Calendar_Offset;
memset((* ddat2)->reserved, 0, 6); /* set the reserved values */

/* Fill in Era Table for DDAT2 */

era_t2->Num_Elems = 1;
era_t2->Element[0].Origin_Date = GREGORIAN_TIMELINE_START;
era_t2->Element[0].Era_Name[0] = 'A';
era_t2->Element[0].Era_Name[1] = 'D';
memset(era_t2->Element[0].reserved, 0, 12); /* set the reserved v alues */

/* Fill in Calendar Table for DDAT2 */

cal_t2->Num_Elems = 2;
cal_t2->Element->Effect_Date = GREGORIAN_TIMELINE_START;
cal_t2->Element->Type = 0x0001;
memset(cal_t2->Element->reserved,'0',10);
(cal_t2->Element->Effect_Date = GREGORIAN_TIMELINE_END;
(cal_t2->Element->Type = 0;
memset((cal_t2->Element->reserved,'0',10);
} /* end of if (first) */
if (cvt_type == 'I')
{
/* converting to mitime ? */
/* Yes, then ... */
(* ddat1)->Format_Code = IMP1_CLOCK; /* Set source format code */
(* ddat1)->Format_Code = SAA_TIMESTAMP; /* Set source format code */
(* inst_t)->Length_1 = 8; /* Result timestamp */
(* inst_t)->Length_2 = 26; /* input mitime */
}
else
{
(* ddat1)->Format_Code = SAA_TIMESTAMP; /* Set source format code */
(* ddat2)->Format_Code = IMP1_CLOCK; /* Set source format code */
(* inst_t)->Length_1 = 26; /* Result timestamp */
(* inst_t)->Length_2 = 8; /* input mitime */
}
} /* end of procedure */

.....
/* Defines for parameters passed to DBLOAD */
.....
#define FROMFILE 1
#define TOFILE 2
#define TOFILENAME 3
#define TOLIB 4
#define FROMCMD 5
#define TORCD 6
#define MTASK 7
#define TASKNBR 8
#define NBRTASKS 9
#define FTYPE 10
#define STR_DELIMIT 11
#define DELIMIT 12
#define ROW 13
#define DATFMT 14
#define DATSEP 15
#define TIMFMT 16
#define TIMESEP 17
#define DECMP1 18
.....
/* Miscellaneous Defines */
.....
#define FALSE 0
#define TRUE !FALSE
#define AS400 /* AS400 */
.....
/* MoveField type codes */
.....
#define CHAR_GRAPH 0x00
#define VAR_CHAR_GRAPH 0x01
#define INTEGER 0x02
#define SMALLINT 0x03
#define FLOAT 0x04
#define PACKED 0x05
#define ZONED 0x06
#define DATE 0x07
#define TIME 0x08
#define TIMESTAMP 0x09
#define NUMERIC_CHAR 0x0A
.....
/* Date and time defines */
.....
#define USA 0x00
#define ISO 0x01
#define EUR 0x02
#define JIS 0x03
#define MDY 0x04
#define DMY 0x05
#define YMD 0x06
#define JUL 0x07
#define HMS 0x08
#define SAA 0x09
.....
/* ERRORS */
.....
#define OPEN_IMPORT 1
#define OPEN_DBFILE 2
#define COL_DEL_NOT_FOUND 3
#define COLUMNS_ERROR 4
#define FIELD_LENGTH 5
#define STRING_DELIMIT 6
#define NULL_ERROR 7
#define MEM_ERROR 8
#define MISSING_STR_DELIMIT 9
#define STR_RECORD 10
#define END_RECORD 11
#define OPEN_HDRFILE 12
#define TSKS_ERROR 13
#define NUMERIC_ERROR 14
#define DECIMAL_ERROR 15
#define WRITE_ERROR 16
.....
/* Common structure */
.....
typedef struct
{
char * colptr; /* Data ptr - import file records */
char * inptr; /* Field ptr - import file records */
char * outptr; /* Output buffer pointer */
char * outstart; /* Pointer to start of output buffer */
char * decptr; /* Decimal point pointer */
char * transptr; /* Pointer to transaction input */
int ix; /* Loop index */
decimal(3,0) new_count; /* New order line numbers */
short buflen; /* Import input buffer length */
short columnCnt; /* Number of output columns allowed */
short fidcnt; /* Current number of fields found */
short fidlen; /* Length of current input field */
short blankcnt; /* Trailing blank count (numeric) */
char col_delim; /* Column delimiter */
char row_delim; /* Row delimiter */
char decmpt; /* decimal point */
}
struct
{
unsigned Data : 1; /* Numeric field data flag */
} Numeric;
char header; /* Looking at header data */
char end_trans; /* processing end of new_order */
char tran_type; /* transaction type flag */
char last_name; /* set when last name was detected */
char cust_number; /* set when a customer number used */
char tran_error; /* transaction error flag */
char sign_on_error; /* sign on screen error */
} ImportParms;
.....
/* Output record column descriptions */
.....
typedef struct
{
unsigned char fidtype; /* Field data type */
unsigned char MoveField_code; /* Branch code for MoveField rtn */
int fidlen; /* Field length */
short precision; /* Precision for decimal fields */
short scale; /* Scale for decimal fields */
short left; /* Data length left of decimalpoint */
short offset; /* Offset to this field from start */
}
/* offset from ip->outstart */
/* unsigned char number_chk; /* Validity check flag */
/* unsigned char skip; /* skip saving this item */
} column_info;
/* Current column_info ptr */
/* column_info *ci_str; /* Pointer to start of column_info */
.....
/* Function Prototypes */
.....
void ImportFields(ImportParms * ip);
void ColRowDelimiter(ImportParms * ip);
void MoveField(ImportParms * ip);
void do_new_order(ImportParms * ip);
void do_payment(ImportParms * ip);
void do_order_status(ImportParms * ip);
void check_input_string(ImportParms * ip); /* check the input string */
.....
```

## D.9 NEWORDSRV.C:

```
#include "common.h"

/* tpsvrinit is executed when a server is booted, before it begins
processing requests. It is not necessary to have this function.
Also available is tpsvrdone (not used in this example), which is
called at server shutdown time.
*/

static int sd, sd2, addr=0, length, port=4000;
static pid_t pid;
int rc=670, remain=0;
error_code err_code;
data_area_data data_area_input;
int mod_number=0;
char clnt_ip1[13], clnt_ip2[13];
char svr_ip1[13], svr_ip2[13];
char cmd[40] = "DLYJOB DLY(3600)";

tpsvrinit(int argc, char *argv[])
{
    /* Some compilers warn if argc and argv aren't used. */
    argc = argc;
    argv = argv;

    /* userlog writes to the central TUXEDO message log */
    userlog("Welcome to the tpcsvr server");
    err_code.bytes_prov = 0;
    QWCRDTAA ( &data_area_input, 46, modn_dtaara, 1, 1, &err_code );
    mod_number = atoi(data_area_input.data_block);
    QWCRDTAA ( &data_area_input, 48, clnt_dtaara, 1, 11, &err_code );
    memcpy( clnt_ip1, &data_area_input.data_block, 11 );
    clnt_ip1[12] = '\0';
    QWCRDTAA ( &data_area_input, 48, clnt_dtaara, 13, 11, &err_code );
    memcpy( clnt_ip2, &data_area_input.data_block, 11 );
    clnt_ip2[12] = '\0';
    QWCRDTAA ( &data_area_input, 48, svr_dtaara, 1, 11, &err_code );
    memcpy( svr_ip1, &data_area_input.data_block, 11 );
    svr_ip1[12] = '\0';
    QWCRDTAA ( &data_area_input, 48, svr_dtaara, 13, 11, &err_code );
    memcpy( svr_ip2, &data_area_input.data_block, 11 );
    svr_ip2[12] = '\0';

    pid = getpid();
    remain = pid % mod_number;
    if(remain == 0) {
        addr=inet_addr(clnt_ip1);
        sd=bind(addr, pid);
        addr=inet_addr(svr_ip1);
        sd2=bconnect(sd,addr,port);
    }
    else {
        addr=inet_addr(clnt_ip2);
        sd=bind(addr, pid);
        addr=inet_addr(svr_ip2);
        sd2=bconnect(sd,addr,port);
    }
    return(0);
}

/* This function performs the actual service requested by the client.
Its argument is a structure containing among other things a pointer
to the data buffer, and the length of the data buffer.
*/

/* NEW ORDER */
NEWORDER (TPSVCINFO *rqst)
{
    if(send(sd, rqst->data, 230, 0)<0) {
        perror("nsend() failed");
        tpreturn(TPFALL, 0, rqst->data, 0L, 0);
    }

    memset(rqst->data, 'V', 670);

    rc=recv(sd, rqst->data, 670, 0);

    if(rc<=0) {
        perror("nrecv[0]() failed");
        tpreturn(TPFALL, 0, rqst->data, 0L, 0);
    }

    /* Return the transformed buffer to the requestor. */
    tpreturn(TPSUCCESS, 0, rqst->data, rc, 0);
}

```

## D.10 ORDSTSSRV.C:

```
#include "common.h"

/* tpsvrinit is executed when a server is booted, before it begins
processing requests. It is not necessary to have this function.
Also available is tpsvrdone (not used in this example), which is
called at server shutdown time.
*/

static int sd, sd2, addr=0, length, port=6000;
static pid_t pid;
int rc=670, remain=0;
error_code err_code;
data_area_data data_area_input;
int mod_number=0;
char clnt_ip1[13], clnt_ip2[13];
char svr_ip1[13], svr_ip2[13];
char cmd[40] = "DLYJOB DLY(3600)";

tpsvrinit(int argc, char *argv[])
{
    /* Some compilers warn if argc and argv aren't used. */
    argc = argc;
    argv = argv;

    /* userlog writes to the central TUXEDO message log */
    userlog("Welcome to the tpcsvr server");
    err_code.bytes_prov = 0;
    QWCRDTAA ( &data_area_input, 46, modn_dtaara, 1, 1, &err_code );
    mod_number = atoi(data_area_input.data_block);
    QWCRDTAA ( &data_area_input, 48, clnt_dtaara, 1, 11, &err_code );
    memcpy( clnt_ip1, &data_area_input.data_block, 11 );
    clnt_ip1[12] = '\0';
    QWCRDTAA ( &data_area_input, 48, clnt_dtaara, 13, 11, &err_code );
    memcpy( clnt_ip2, &data_area_input.data_block, 11 );
    clnt_ip2[12] = '\0';
    QWCRDTAA ( &data_area_input, 48, svr_dtaara, 1, 11, &err_code );
    memcpy( svr_ip1, &data_area_input.data_block, 11 );
    svr_ip1[12] = '\0';

    pid = getpid();
    remain = pid % mod_number;
    if(remain == 0) {
        addr=inet_addr(clnt_ip1);
        sd=bind(addr, pid);
        addr=inet_addr(svr_ip1);
        sd2=bconnect(sd,addr,port);
    }
    else {
        addr=inet_addr(clnt_ip2);
        sd=bind(addr, pid);
        addr=inet_addr(svr_ip2);
        sd2=bconnect(sd,addr,port);
    }
    return(0);
}

/* This function performs the actual service requested by the client.
Its argument is a structure containing among other things a pointer
to the data buffer, and the length of the data buffer.
*/

/* PAYMENT */
PAYMNT (TPSVCINFO *rqst)
{
    if(send(sd, rqst->data, 230, 0)<0) {
        perror("nsend() failed");
        tpreturn(TPFALL, 0, rqst->data, 0L, 0);
    }

    memset(rqst->data, 'V', 670);

    rc=recv(sd, rqst->data, 670, 0);

    if(rc<=0) {
        perror("nrecv[0]() failed");
    }
}

```

```
QWCRDTAA ( &data_area_input, 48, svr_dtaara, 13, 11, &err_code );
memcpy( svr_ip2, &data_area_input.data_block, 11 );
svr_ip2[12] = '\0';

    pid = getpid();
    remain = pid % mod_number;
    if(remain == 0) {
        addr=inet_addr(clnt_ip1);
        sd=bind(addr, pid);
        addr=inet_addr(svr_ip1);
        sd2=bconnect(sd,addr,port);
    }
    else {
        addr=inet_addr(clnt_ip2);
        sd=bind(addr, pid);
        addr=inet_addr(svr_ip2);
        sd2=bconnect(sd,addr,port);
    }
    return(0);
}

/* This function performs the actual service requested by the client.
Its argument is a structure containing among other things a pointer
to the data buffer, and the length of the data buffer.
*/

/* ORDER STATUS */
ORDSTS (TPSVCINFO *rqst)
{
    if(send(sd, rqst->data, 230, 0)<0) {
        perror("nsend() failed");
        tpreturn(TPFALL, 0, rqst->data, 0L, 0);
    }

    memset(rqst->data, 'V', 670);

    rc=recv(sd, rqst->data, 670, 0);

    if(rc<=0) {
        perror("nrecv[0]() failed");
        tpreturn(TPFALL, 0, rqst->data, 0L, 0);
    }

    /* Return the transformed buffer to the requestor. */
    tpreturn(TPSUCCESS, 0, rqst->data, rc, 0);
}

```

## D.11 PAYMNTSRV.C:

```
#include "common.h"

/* tpsvrinit is executed when a server is booted, before it begins
processing requests. It is not necessary to have this function.
Also available is tpsvrdone (not used in this example), which is
called at server shutdown time.
*/

static int sd, sd2, addr=0, length, port=5000;
static pid_t pid;
int rc=670, remain=0;
error_code err_code;
data_area_data data_area_input;
int mod_number=0;
char clnt_ip1[13], clnt_ip2[13];
char svr_ip1[13], svr_ip2[13];
char cmd[40] = "DLYJOB DLY(3600)";

tpsvrinit(int argc, char *argv[])
{
    /* Some compilers warn if argc and argv aren't used. */
    argc = argc;
    argv = argv;

    /* userlog writes to the central TUXEDO message log */
    userlog("Welcome to the tpcsvr server");
    err_code.bytes_prov = 0;
    QWCRDTAA ( &data_area_input, 46, modn_dtaara, 1, 1, &err_code );
    mod_number = atoi(data_area_input.data_block);
    QWCRDTAA ( &data_area_input, 48, clnt_dtaara, 1, 11, &err_code );
    memcpy( clnt_ip1, &data_area_input.data_block, 11 );
    clnt_ip1[12] = '\0';
    QWCRDTAA ( &data_area_input, 48, clnt_dtaara, 13, 11, &err_code );
    memcpy( clnt_ip2, &data_area_input.data_block, 11 );
    clnt_ip2[12] = '\0';
    QWCRDTAA ( &data_area_input, 48, svr_dtaara, 1, 11, &err_code );
    memcpy( svr_ip1, &data_area_input.data_block, 11 );
    svr_ip1[12] = '\0';
    QWCRDTAA ( &data_area_input, 48, svr_dtaara, 13, 11, &err_code );
    memcpy( svr_ip2, &data_area_input.data_block, 11 );
    svr_ip2[12] = '\0';

    pid = getpid();
    remain = pid % mod_number;
    if(remain == 0) {
        addr=inet_addr(clnt_ip1);
        sd=bind(addr, pid);
        addr=inet_addr(svr_ip1);
        sd2=bconnect(sd,addr,port);
    }
    else {
        addr=inet_addr(clnt_ip2);
        sd=bind(addr, pid);
        addr=inet_addr(svr_ip2);
        sd2=bconnect(sd,addr,port);
    }
    return(0);
}

/* This function performs the actual service requested by the client.
Its argument is a structure containing among other things a pointer
to the data buffer, and the length of the data buffer.
*/

/* PAYMENT */
PAYMNT (TPSVCINFO *rqst)
{
    if(send(sd, rqst->data, 230, 0)<0) {
        perror("nsend() failed");
        tpreturn(TPFALL, 0, rqst->data, 0L, 0);
    }

    memset(rqst->data, 'V', 670);

    rc=recv(sd, rqst->data, 670, 0);

    if(rc<=0) {
        perror("nrecv[0]() failed");
    }
}

```



```

"body"
"/HTML"
);
char *stklvlResult = (
<html>
<head>
<title>TPC-C Stock Level</title>
</head>
<body>
<FORM ACTION=tpcc.dll METHOD="GET">
<INPUT TYPE="hidden" NAME="FORMID" VALUE="7">
<INPUT TYPE="hidden" NAME="TERMINID" VALUE="sa">
<INPUT TYPE="hidden" NAME="SYNCID" VALUE="sa">
<center><h1>TPC-C Stock Level</h1></center>
<pre>
<br>Warehouse: $5.5d      District: $2.2d
<br>
<br>Stock Level Threshold: $2.2d
<br>
<br>Low Stock: $3.3d<br>
<INPUT TYPE="submit" NAME="CMD" VALUE="..NewOrder..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Payment..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Delivery..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Order-Status..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Stock-Level..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Exit..">
</pre>
</body>
"/HTML"
);
char *signonResult = (
<html>
<head>
<title>TPC-C Sign On Error in CGI</title>
</head>
<body>
<FORM ACTION=tpcc.dll METHOD="GET">
<INPUT TYPE="hidden" NAME="FORMID" VALUE="1">
<INPUT TYPE="hidden" NAME="TERMINID" VALUE="2">
<INPUT TYPE="hidden" NAME="SYNCID" VALUE="0">
<h1>TPC-C Sign error detected</h1>
<center><h2>Please sign on again</h2></center>
<center><h2>The signon information has been lost.</h2></center>
Warehouse ID <INPUT NAME="id" SIZE=5 MAXLENGTH=5><BR>
District ID <INPUT NAME="d_id" SIZE=2 MAXLENGTH=2><BR>
<HR>
<INPUT TYPE="submit" NAME="CMD" VALUE="Submit">
</body>
"/HTML"
);
char *stkvlError = (
<html>
<head>
<title>TPC-C Stock Level Error</title>
</head>
<body>
<FORM ACTION=tpcc.dll METHOD="GET">
<INPUT TYPE="hidden" NAME="FORMID" VALUE="7">
<INPUT TYPE="hidden" NAME="TERMINID" VALUE="sa">
<INPUT TYPE="hidden" NAME="SYNCID" VALUE="sa">
<center><h1>TPC-C Stock Level Error</h1></center>
<pre>
<br>Warehouse: $5.5d      District: $2.2d
<br>
<br>Stock Level Threshold: $2.2d
<br>
<br>Invalid stock level input<br>
<INPUT TYPE="submit" NAME="CMD" VALUE="..NewOrder..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Payment..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Delivery..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Order-Status..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Stock-Level..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Exit..">
</pre>
</body>
"/HTML"
);
char *dataErrorResult = (
<html>
<head>
<title>TPC-C Data Error in CGI</title>
</head>
<body>
<FORM ACTION=tpcc.dll METHOD="GET">
<INPUT TYPE="hidden" NAME="FORMID" VALUE="2">
<INPUT TYPE="hidden" NAME="TERMINID" VALUE="sa">
<INPUT TYPE="hidden" NAME="SYNCID" VALUE="sa">
<h1>TPC-C Data entry error detected</h1>
<center><h2>Please select a transaction from the menu</h2></center>
<center><h2>or select Back and correct the data</h2></center>
<br>
<INPUT TYPE="submit" NAME="CMD" VALUE="..NewOrder..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Payment..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Delivery..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Order-Status..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Stock-Level..">
<INPUT TYPE="submit" NAME="CMD" VALUE="..Exit..">
</body>
"/HTML"
);
int postResult(void *res_buf, char * warehouse, char * district,
int formIdPost, char * scratchBuffer )
(
int i;
unsigned char *toPtr; /* storage location */
char wzip_tag; /* ending character for warehouse zip */
char dzip_tag; /* ending character for district zip */
char czip_tag; /* ending character for customer zip */
char clast_tag; /* ending of customers last name */
char cfirst_tag; /* ending of customers first name */
char waddress1_tag; /* ending of warehouse address one */
char waddress2_tag; /* ending of warehouse address two */
char daddress1_tag; /* ending of district address one */
char daddress2_tag; /* ending of district address two */
char wcity_tag; /* ending of warehouse city */
char dcity_tag; /* ending of district city */
char cphone_tag; /* ending of customers last name */
char caddress1_tag; /* ending for customer address 1 */
char caddress2_tag; /* ending for customer address 1 */
char ccity_tag; /* ending for customer city */
char cdata1_tag; /* ending for customer data 1 */
char cdata2_tag; /* ending for customer data 2 */
char cdata3_tag; /* ending for customer data 3 */
char cdata4_tag; /* ending for customer data 4 */
char item_name_tag; /* last character of the item name */
char div_time[27]; /* time string used for conversion */
time[27]; /* converted output return 27 characters */
char cl_time[27]; /* converted output return */
char _MI_Time time_of_day; /* time of dat for new order transaction */
char cmd[20];
char new_order_display t;
char payment_display p;
char order_status_display o;
char div_input d;
char stock_display s;
#pragma exception_handler(ME,0,0,C2_ALL,CTIA_HANDLE,"MCH1202")
switch(formIdPost) {
case NEW_ORDER:
t = ( new_order_display * ) res_buf;
/* convert ordtm to date */
/* getdtmstr ( time, t->new_out->centm, ) */
mattod ( time_of_day );
time_of_day[7] = 'B'; /* set the period flag */
getdtmstr ( time_of_day, time, 's' );
/* returns a 27 character null terminated string */
/* remove the microsecond */
time[19] = '\0'; /* terminate the string */
if ( t->new_out->neword_result == 'G' )
{
clast_tag = t->new_out->clast[15]; /* get last character */
t->new_out->clast[15] = '\0'; /* terminate string */
sprintf(scratchBuffer,newOrderResult,
warehouse,
district,
t->new_inp->cwid,
t->new_inp->cdid,
time,
t->new_inp->cid,
t->new_out->clast, clast_tag,
t->new_out->ccredit[0],
t->new_out->ccredit[1],
t->new_out->ccdt,
t->new_out->oid,
t->new_inp->number_of_items,
t->new_out->wtax,
t->new_out->dtax
);
toPtr = scratchBuffer + strlen(scratchBuffer);
for (i=0; i<t->new_inp->number_of_items; ++i){
item_name_tag = t->new_out->output_line[i].output_item_name[23];
t->new_out->output_line[i].output_item_name[23]='\0'; /* terminate */
sprintf(toPtr,
"%5d %6d %23s%c %3D(3,0) %3d %c %36.2D(5,2) %87.2D(7 ,2)<br>",
t->new_inp->input_line[i].olspwh,
t->new_inp->input_line[i].oliid,
t->new_out->output_line[i].output_item_name, item_name_tag,
t->new_inp->input_line[i].olqty,
t->new_out->output_line[i].output_qty,
t->new_out->output_line[i].output_borg,
t->new_out->output_line[i].output_item_price,
t->new_out->output_line[i].output_olamt);
toPtr += strlen(toPtr);
}
t->execution_status[0] = '\0'; /* make blank */
}
else
{
sprintf(scratchBuffer,newOrderError,
warehouse,
district,
t->new_inp->cwid,
t->new_inp->cdid,
time,
t->new_inp->cid
);
toPtr = scratchBuffer + strlen(scratchBuffer);
for (i=0; i<t->new_inp->number_of_items; ++i){
sprintf(toPtr,
"%5d %6d %3D(3,0)<br>",
t->new_inp->input_line[i].olspwh,
t->new_inp->input_line[i].oliid,
t->new_inp->input_line[i].olqty );
toPtr += strlen(toPtr);
}
t->new_out->totamt = 0.0;
if ( t->new_out->neword_result == 'I' )
strcpy( t->execution_status, "INVALID INPUT"); /* set to ID */
else
strcpy( t->execution_status, "Item number is not valid " );
}
} /* end of the else */
sprintf(toPtr,newOrderTail,
t->execution_status,
t->new_out->totamt);
break;
case PAYMENT:
p = ( payment_display * ) res_buf;
/* terminate output strings */
if( ( p->pay_out->output_fmt_num[0] != '7' ) ||
( p->pay_out->output_fmt_num[1] != '7' ) )
{
waddress1_tag = p->pay_out->waddr1[19];
daddress1_tag = p->pay_out->daddr1[19];
waddress2_tag = p->pay_out->waddr2[19];
daddress2_tag = p->pay_out->daddr2[19];
wcity_tag = p->pay_out->wcity[19];
dcity_tag = p->pay_out->dcity[19];
clast_tag = p->pay_out->clast[15];
cfirst_tag = p->pay_out->cfirst[15];
caddress1_tag = p->pay_out->caddr1[19];
caddress2_tag = p->pay_out->caddr2[19];
ccity_tag = p->pay_out->ccity[19];
cphone_tag = p->pay_out->cphone[15];
cdata1_tag = p->pay_out->cdat1[49];
cdata2_tag = p->pay_out->cdat2[49];
cdata3_tag = p->pay_out->cdat3[49];
cdata4_tag = p->pay_out->cdat4[49];
p->pay_out->waddr1[19]='\0';
p->pay_out->daddr1[19]='\0';
p->pay_out->waddr2[19]='\0';
p->pay_out->daddr2[19]='\0';
p->pay_out->wcity[19]='\0';
p->pay_out->dcity[19]='\0';
p->pay_out->clast[15]='\0';
p->pay_out->cfirst[15]='\0';
p->pay_out->caddr1[19]='\0';
p->pay_out->caddr2[19]='\0';
p->pay_out->ccity[19]='\0';
p->pay_out->cphone[15]='\0';
p->pay_out->cdat1[49]='\0';
p->pay_out->cdat2[49]='\0';
p->pay_out->cdat3[49]='\0';
p->pay_out->cdat4[49]='\0';
wzip_tag = p->pay_out->wzip[8]; /* copy zip */
dzip_tag = p->pay_out->dzip[8]; /* copy zip */
czip_tag = p->pay_out->czip[8]; /* copy zip */
p->pay_out->wzip[8] = '\0'; /* terminate warehouse zip */
p->pay_out->dzip[8] = '\0'; /* terminate district zip */
p->pay_out->czip[8] = '\0'; /* terminate customer zip */
/* convert time to string */
/* inputs */
p->pay_out->cldate,
getdtmstr ( p->pay_out->cldate, cl_time, 'm' );
/* returns a 27 character null terminated string */
/* remove the microsecond */
cl_time[19] = '\0'; /* terminate the string */
/* p->pay_out->time of day, */
/* output cldate and time */
getdtmstr ( p->pay_out->time_of_day, time, 'm' );
/* returns a 27 character null terminated string */
/* remove the microsecond */
time[19] = '\0'; /* terminate the string */
sprintf(scratchBuffer,paymentResult,

```

```

warehouse,
district,
time,
p->pay_inp->wid,
p->pay_inp->dclid,
p->pay_out->waddr1, waddress1_tag,
p->pay_out->daddr1, daddress1_tag,
p->pay_out->waddr2, waddress2_tag,
p->pay_out->daddr2, daddress2_tag,
p->pay_out->wcity, wcity_tag,
p->pay_out->wstate[0],
p->pay_out->wstate[1],
p->pay_out->wzip, wzip_tag,
p->pay_out->dcity, dcity_tag,
p->pay_out->dstate[0],
p->pay_out->dstate[1],
p->pay_out->dzip, dzip_tag,
p->pay_out->dcid,
p->pay_inp->cwid,
p->pay_inp->ccid,
p->pay_out->clast, clast_tag,
p->pay_out->clint[0],
p->pay_out->clint[1],
p->pay_out->cfirst, cfirst_tag,
cl_time,
p->pay_out->ccaddr1, caddress1_tag,
p->pay_out->ccredit[0],
p->pay_out->ccredit[1],
p->pay_out->ccaddr2, caddress2_tag,
p->pay_out->ccid,
p->pay_out->ccity, ccity_tag,
p->pay_out->ccstate[0],
p->pay_out->ccstate[1],
p->pay_out->cczip, cczip_tag,
p->pay_out->ccphone, ccphone_tag,
p->pay_inp->amount,
p->pay_out->cbal,
p->pay_out->ccrdm,
p->pay_out->cdat1, cdata1_tag,
p->pay_out->cdat2, cdata2_tag,
p->pay_out->cdat3, cdata3_tag,
p->pay_out->cdat4, cdata4_tag
);
}
else
{
clast_tag = p->pay_inp->clast[15];
p->pay_inp->clast[15] = '\0';
sprintf(scratchBuffer, paymentError,
warehouse,
district,
p->pay_inp->wid,
p->pay_inp->dclid,
p->pay_inp->cwid,
p->pay_inp->ccid,
p->pay_inp->clast, clast_tag,
p->pay_inp->payment_type,
p->pay_inp->amount );
}
break;

case ORDER_STATUS:
o = ( order_status_display *) res_buf;
if ( ( o->ordsts_out->ordsts_fmt_num[0] != '6') ||
( o->ordsts_out->ordsts_fmt_num[1] != '2') )
{
/* convert time o->ordsts_out->oetotd, */
/* output is in time */
getdtmstr ( o->ordsts_out->oetotd, time, 'm' );
/* returns a 27 character null terminated string */
/* remove the microsecond */
time[19] = '\0'; /* terminate the string */
/* set termination for the customer first and last name */
clast_tag = o->ordsts_out->clast[15];
cfirst_tag = o->ordsts_out->cfirst[15];
o->ordsts_out->clast[15] = '\0';
o->ordsts_out->cfirst[15] = '\0';
if ( o->ordsts_out->ocarcid[0] == '\0' )
o->ordsts_out->ocarcid[0] = ' ';
if ( o->ordsts_out->ocarcid[1] == '\0' )
o->ordsts_out->ocarcid[1] = ' ';
sprintf(scratchBuffer, ordstsResult,
warehouse,
district,
o->ordsts_inp->wid,
o->ordsts_inp->dclid,
o->ordsts_out->ocid,
o->ordsts_out->clast, clast_tag,
o->ordsts_out->clint[0],
o->ordsts_out->clint[1],
o->ordsts_out->cfirst, cfirst_tag,
o->ordsts_out->cbal,
o->ordsts_out->oid,
time, /* o->ordsts_out->oetotd, */
o->ordsts_out->ocarcid[0],
o->ordsts_out->ocarcid[1]
);
toPtr = scratchBuffer + strlen(scratchBuffer);
for (i=0; i<o->ordsts_out->olenbr; ++i){
/* conver orderline deliver time to time of day */
/*
/* o->ordsts_out->olineinfo[i].do_oldivl */
/* output as div time char 15 output array */
/* convert div time to date time string */
getdtmstr ( o->ordsts_out->olineinfo[i].do_oldivl, div_time, 'm' );
/* returns a 27 character null terminated string */
/* remove the microsecond */
div_time[19] = '\0'; /* terminate the string */
sprintf(toPtr, " %5d %6d %2s %2d %6.2D(7,2) %14s<br>",
o->ordsts_out->olineinfo[i].do_olapwh,
o->ordsts_out->olineinfo[i].do_olliid,
SPACE,
o->ordsts_out->olineinfo[i].do_olqty,
o->ordsts_out->olineinfo[i].do_olamnt,
div_time );
toPtr += strlen(toPtr);
}
sprintf(toPtr, ordstsTail);
}
else
{
clast_tag = o->ordsts_inp->clast[15];
o->ordsts_inp->clast[15] = '\0';
sprintf(scratchBuffer, ordstsError,
warehouse,
district,
o->ordsts_inp->wid,
o->ordsts_inp->dclid,
o->ordsts_inp->cwid,
o->ordsts_inp->ccid,
o->ordsts_inp->clast, clast_tag,
o->ordsts_inp->ordsts_type );
}
break;

case DELIVERY:
d = ( div_input *) res_buf;
sprintf(scratchBuffer, deliveryResult,
warehouse,
district,
d->wid,
d->carrier[0],
d->carrier[1]
);
break;

case STOCK_LEVEL:
s = ( stock_display *) res_buf;
if ( s->stk_out->result != 'I' )
{
sprintf(scratchBuffer, stklvlResult,
warehouse,
district,
s->stk_inp->wid,
s->stk_inp->dclid,
s->stk_inp->threshold,
s->stk_out->blwstk
);
}
else
{
sprintf(scratchBuffer, stklvlError,
warehouse,
district,
s->stk_inp->wid,
s->stk_inp->dclid,
s->stk_inp->threshold
);
}
break;

case SIGNON_ERROR:
sprintf(scratchBuffer, signonResult );
break;

case DATA_ERROR:
sprintf(scratchBuffer, dataErrorResult,
warehouse, district );
break;

default:
printf("postResult: Invalid form ID = %d", formIdPost );
return (-1);
break;
} /* end of switch */

#pragma disable_handler
return 0;
ME:
memcpy(trcoffcmd, "TRCINT SET(*HOLD) TRCTBL(TPCCTRC)", 40);
trcoffcmd[40] = '\0';
system(trcoffcmd);
memcpy(cmd, "DLIJOB DLY(3600)", 20);
cmd[20] = '\0';
system(cmd);
}

D.13 SCANINPUT.C:
#include "Common.h"
#define DATA_ERROR 30
#define SIGNON_ERROR 31
/* External functions */
/* Internal procedures */
int getdtmstr(char * timein, char * timestamp, char type);

char *newOrderResult = {
<html>
<head>
<title>TPC-C New Order</title>
</head>
<body>
<FORM ACTION=tpcc.dll METHOD=GET">
<INPUT TYPE=hidden NAME=FORMID VALUE=3">
<INPUT TYPE=hidden NAME=TERMINID VALUE=3">
<INPUT TYPE=hidden NAME=SYNCID VALUE=3">
<center><h1> TPC-C New Order</h1></center>
<p>
<br>Warehouse: $5.5d District: $2.2d
"
"
"Date: %s<br>
"Customer: $4.4d Name: $15s&c Credit: %8c Disc: $5.2D(5,2)<br>
"Order Number: $8d Number of Lines: $2D(3,0) W_tax: $4.2D(5,2)"
" D_tax: $4.2D(5,2)<br><br>
" <table>
" <tr>
" <th>Supp_w </th>
" <th>Item_id </th>
" <th Colspan=75>Item Name </th>
" /* 1234567890123456789012345 */
" <tr>
" <td>
" <td>Qty </td>
" <td>Stock </td>
" <td>B/G </td>
" <td Colspan=35>Price </td>
" <td>Amount</td>
" </tr>
" </table>
};

char *newOrderError = {
<html>
<head>
<title>TPC-C New Order Error</title>
</head>
<body>
<FORM ACTION=tpcc.dll METHOD=GET">
<INPUT TYPE=hidden NAME=FORMID VALUE=3">
<INPUT TYPE=hidden NAME=TERMINID VALUE=3">
<INPUT TYPE=hidden NAME=SYNCID VALUE=3">
<center><h1> TPC-C New Order Error</h1></center>
<p>
<br>Warehouse: $5.5d District: $2.2d
"
"
"Date: %s<br>
"Customer: $4.4d <br>
" <table>
" <tr>
" <td>
" <td>Supp_w </td>
" <td>Item_id </td>
" <td>
" <td>Qty </td>
" </tr>
" </table>
};

```



```

};
char *newOrderTail = {
"Execution Status: $24s Total: $88.2D(11,2)<br>"
"<br><br>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..NewOrder..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Payment..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Delivery..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Order-Status..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Stock-Level..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Exit..'>"
"/pre>"
"/body>"
"/HTML>"
};
char *paymentResult = {
"<html>"
"<head>"
"<title>TPC-C Payment</title>"
"/head>"
"<body>"
"<FORM ACTION=tpcc.d11 METHOD='GET'>"
"<INPUT TYPE='hidden' NAME='FORMID' VALUE='4'>"
"<INPUT TYPE='hidden' NAME='TERMINID' VALUE='$$'>"
"<INPUT TYPE='hidden' NAME='SYNCID' VALUE='$$'>"
"<center><h1> TPC-C Payment</h1></center>"
"<pre>"
"<br>Warehouse: $5.5d District: $2.2d"
"<br>Warehouse: $5.5d /* whrs address 1 */"
" /* gap */"
" $19s8c /* dstrct address 1 */"
"<br>$19s8c /* whrs address 2 */"
" /* gap */"
" $19s8c /* dstrct address 2 */"
"<br>$19s8c /* whrs city */"
" $8s8c /* whrs state two characters */"
" $19s8c /* whrs zip */"
" $8c8c /* dstrct city */"
" $8s8c /* dstrct state two characters */"
" /* dstrct zip */"
"<br>"
"<br>Customer: $4.4d Cust-Warehouse: $5.5d Cust-District: $2.2d"
"<br>Name: $15s8c $8c $15s8c Since: $9s "
"<br> $19s8c /* whrs state two characters */"
"<br> $19s8c $8c $10s8c $Dstrct: $2.2D(5,2)"
"<br> $19s8c $8c $10s8c Phone: $17s8c"
"<br>"
"<br>Amount Paid: $86.2D(7,2) New Cust-Balance: $88.2D(13,2)"
"<br>Credit Limit: $810.2D(13,2) "
"<br>"
"<br>Cust-data: $49s8c"
"<br> $49s8c"
"<br> $49s8c"
"<br> $49s8c"
"<br><br>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..NewOrder..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Payment..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Delivery..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Order-Status..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Stock-Level..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Exit..'>"
"/pre>"
"/body>"
"/HTML>"
};
char *paymentError = {
"<html>"
"<head>"
"<title>TPC-C Payment Error</title>"
"/head>"
"<body>"
"<FORM ACTION=tpcc.d11 METHOD='GET'>"
"<INPUT TYPE='hidden' NAME='FORMID' VALUE='4'>"
"<INPUT TYPE='hidden' NAME='TERMINID' VALUE='$$'>"
"<INPUT TYPE='hidden' NAME='SYNCID' VALUE='$$'>"
"<center><h1> TPC-C Invalid payment data</h1></center>"
"<pre>"
"<br>Warehouse: $5.5d District: $2.2d"
"<br>Customer: $4.4d Customer-Warehouse: $4.4d Customer-District: $2.2d"
"<br>Name: $15s8c Type: $c"
"<br>Amount Paid: $9D(7,2)"
"<br><br>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..NewOrder..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Payment..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Delivery..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Order-Status..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Stock-Level..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Exit..'>"
"/pre>"
"/body>"
"/HTML>"
};
char *ordstatsResult = {
"<html>"
"<head>"
"<title>TPC-C Order-Status</title>"
"/head>"
"<body>"
"<FORM ACTION=tpcc.d11 METHOD='GET'>"
"<INPUT TYPE='hidden' NAME='FORMID' VALUE='5'>"
"<INPUT TYPE='hidden' NAME='TERMINID' VALUE='$$'>"
"<INPUT TYPE='hidden' NAME='SYNCID' VALUE='$$'>"
"<center><h1> TPC-C Order-Status</h1></center>"
"<pre>"
"<br>Warehouse: $5.5d District: $2.2d"
"<br>Customer: $4.4d"
" Name: $15s8c"
" $8c $15s8c"
"<br>Cust-Balance: $37.2D(13,2)"
"<br> /* gap */"
"<br>Order-Number: $8.2d"
" Entry-Date: $s"
" Carrier-Number: $8c8c"
"<br><br>"
"<table>"
"<tr>"
"<th>Supp-w </th>"
"<th>Item-Id </th>"
"<th>Qty </th>"
"<th>Amount</th>"
"</tr>"
"<tbody>"
"<tr>"
"<td>Delivery-Date</td>"
"</tr>"
"</tbody>"
"</table>"
"<br><br>"
};
char *ordstatsTail = {
"<INPUT TYPE='submit' NAME='CMD' VALUE='..NewOrder..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Payment..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Delivery..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Order-Status..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Stock-Level..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Exit..'>"
"/pre>"
"/body>"
"/HTML>"
};
char *ordstatsError = {
"<html>"
"<head>"
"<title>TPC-C Order-Status Error</title>"
"/head>"
"<body>"
"<FORM ACTION=tpcc.d11 METHOD='GET'>"
"<INPUT TYPE='hidden' NAME='FORMID' VALUE='5'>"
"<INPUT TYPE='hidden' NAME='TERMINID' VALUE='$$'>"
"<INPUT TYPE='hidden' NAME='SYNCID' VALUE='$$'>"
"<center><h1> TPC-C Invalid order status data</h1></center>"
"<pre>"
"<br>Warehouse: $5.5d District: $2.2d"
"<br>Customer: $4.4d"
"<br>Name: $15s8c Type: $c"
"<br><br>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..NewOrder..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Payment..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Delivery..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Order-Status..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Stock-Level..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Exit..'>"
"/pre>"
"/body>"
"/HTML>"
};
char *deliveryResult = {
"<html>"
"<head>"
"<title>TPC-C Delivery</title>"
"/head>"
"<body>"
"<FORM ACTION=tpcc.d11 METHOD='GET'>"
"<INPUT TYPE='hidden' NAME='FORMID' VALUE='6'>"
"<INPUT TYPE='hidden' NAME='TERMINID' VALUE='$$'>"
"<INPUT TYPE='hidden' NAME='SYNCID' VALUE='$$'>"
"<center><h1> TPC-C Delivery</h1></center>"
"<pre>"
"<br>Warehouse: $5.5d"
"<br>"
"<br>Carrier Number: $8c8c"
"<br>"
"<br>Execution Status: Delivery has been queued"
"<br><br>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..NewOrder..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Payment..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Delivery..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Order-Status..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Stock-Level..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Exit..'>"
"/pre>"
"/body>"
"/HTML>"
};
char *stklvlResult = {
"<html>"
"<head>"
"<title>TPC-C Stock Level</title>"
"/head>"
"<body>"
"<FORM ACTION=tpcc.d11 METHOD='GET'>"
"<INPUT TYPE='hidden' NAME='FORMID' VALUE='7'>"
"<INPUT TYPE='hidden' NAME='TERMINID' VALUE='$$'>"
"<INPUT TYPE='hidden' NAME='SYNCID' VALUE='$$'>"
"<center><h1> TPC-C Stock Level</h1></center>"
"<pre>"
"<br>Warehouse: $5.5d District: $2.2d"
"<br>"
"<br>Stock Level Threshold: $2.2d"
"<br>"
"<br>Low Stock: $3.3d<br>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..NewOrder..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Payment..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Delivery..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Order-Status..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Stock-Level..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Exit..'>"
"/pre>"
"/body>"
"/HTML>"
};
char *signonResult = {
"<html>"
"<head>"
"<title>TPC-C Sign On Error in CGI</title>"
"/head>"
"<body>"
"<FORM ACTION=tpcc.d11 METHOD='GET'>"
"<INPUT TYPE='hidden' NAME='FORMID' VALUE='1'>"
"<INPUT TYPE='hidden' NAME='TERMINID' VALUE='-2'>"
"<INPUT TYPE='hidden' NAME='SYNCID' VALUE='0'>"
"<h1>TPC-C Sign error detected</h1>"
"<center><h2>Please sign on again</h2></center>"
"<center><h2>the signon information has been lost.</h2></center>"
"<table border='1'>"
"<tr>"
"<td>Warehouse ID <INPUT NAME='id' SIZE=5 MAXLENGTH=5><br>"
"<td>District ID <INPUT NAME='d_id' SIZE=2 MAXLENGTH=2><br>"
"</tr>"
"</table>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='Submit'>"
"/body>"
"/HTML>"
};
char *stklvlError = {
"<html>"
"<head>"
"<title>TPC-C Stock Level Error</title>"
"/head>"
"<body>"
"<FORM ACTION=tpcc.d11 METHOD='GET'>"
"<INPUT TYPE='hidden' NAME='FORMID' VALUE='7'>"
"<INPUT TYPE='hidden' NAME='TERMINID' VALUE='$$'>"
"<INPUT TYPE='hidden' NAME='SYNCID' VALUE='$$'>"
"<center><h1> TPC-C Stock Level Error</h1></center>"
"<pre>"
"<br>Warehouse: $5.5d District: $2.2d"
"<br>"
"<br>Stock Level Threshold: $2.2d"
"<br>"
"<br>Invalid stock level input<br>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..NewOrder..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Payment..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Delivery..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Order-Status..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Stock-Level..'>"
"<INPUT TYPE='submit' NAME='CMD' VALUE='..Exit..'>"
"/pre>"
"/body>"
"/HTML>"
};
char *dataErrorResult = {
"<html>"
"<head>"
"<title>TPC-C Data Error in CGI</title>"
"/head>"
"<body>"
"<FORM ACTION=tpcc.d11 METHOD='GET'>"
"<INPUT TYPE='hidden' NAME='FORMID' VALUE='2'>"
"<INPUT TYPE='hidden' NAME='TERMINID' VALUE='$$'>"
"<INPUT TYPE='hidden' NAME='SYNCID' VALUE='$$'>"
"<h1>TPC-C Data entry error detected</h1>"
"<center><h2>Please select a transaction from the menu</h2></center>"
};

```



```

o->ordsts_out->oid,
time, /* o->ordsts_out->oetm, */
o->ordsts_out->ocard[0],
o->ordsts_out->ocard[1]
);

toPtr = scratchBuffer + strlen(scratchBuffer);
for (i=0; i<o->ordsts_out->olenbr; ++i){
/* convert orderline deliver time to time of day */
/*
o->ordsts_out->olineinfo[i].do_oldivl
/* output as dlv_time char 15 output array */
/*
convert dlv_time to date time string */
getdtmstr (&o->ordsts_out->olineinfo[i].do_oldivl, dlv_time, 'm' );
/* returns a 27 character null terminated string */
/* remove the microsecond */
dlv_time[19] = '\0'; /* terminate the string */
sprintf(toPtr, "%5d %6d %2s %2d %6.20(7,2) %14s\r\n",
o->ordsts_out->olineinfo[i].do_olapsh,
o->ordsts_out->olineinfo[i].do_olliid,
SPACE,
o->ordsts_out->olineinfo[i].do_olqty,
o->ordsts_out->olineinfo[i].do_olamnt,
dlv_time );
toPtr += strlen(toPtr);
}
sprintf(toPtr,ordstsTail);
}
else
{
clast_tag = o->ordsts_inp->clast[15];
o->ordsts_inp->clast[15] = '\0';
sprintf(scratchBuffer,ordstsError,
warehouse,
district,
o->ordsts_inp->wid,
o->ordsts_inp->did,
o->ordsts_inp->cid,
o->ordsts_inp->clast, clast_tag,
o->ordsts_inp->ordsts_type );
}
break;
}
case DELIVERY:
d = ( dlv input *) res_buf;
sprintf(scratchBuffer,deliveryResult,
warehouse,
district,
d->wid,
d->carrier[0],
d->carrier[1]
);
break;
}
case STOCK_LEVEL:
s = ( stock display *) res_buf;
if ( s->stk_out->result!='I' )
{
sprintf(scratchBuffer,stklvlResult,
warehouse,
district,
s->stk_inp->wid,
s->stk_inp->did,
s->stk_inp->threshold,
s->stk_out->blwstk
);
}
else
{
sprintf(scratchBuffer,stkvlError,
warehouse,
district,
s->stk_inp->wid,
s->stk_inp->did,
s->stk_inp->threshold
);
}
break;
}
case SIGNON_ERROR:
sprintf(scratchBuffer,signonResult );
break;
}
case DATA_ERROR:
sprintf(scratchBuffer,dataErrorResult,
warehouse, district );
break;
}
default:
printf("postResult: Invalid form ID = %d", formIdPost );
return (-1);
break;
}
} /* end of switch */
}
#pragma disable_handler
return 0;
ME:
memcpy(trcoffcmd, "FRCINT SET(*HOLD) TRCTBL(TPCCTRC)", 40);
trcoffcmd[40] = '\0';
system(trcoffcmd);
memcpy(cmd, "DLXJOB DLY(3600)", 20);
cmd[20] = '\0';
system(cmd);
}
}

```

## D.14 STKLVL.SRV.C:

```

#include "common.h"

/* tpsvrint is executed when a server is booted, before it begins
processing requests. It is not necessary to have this function.
Also available is tpsvrdone (not used in this example), which is
called at server shutdown time.
*/

static int sd, sd2, addr=0, length, port=8000;
static pid_t pid;
int rc=670, remain=0;
error_code err_code;
data_area_data data_area_input;
int mod_number=0;
char clnt_ip1[13], clnt_ip2[13];
char srvr_ip1[13], srvr_ip2[13];

tpsvrint(int argc, char *argv[])
{
/* Some compilers warn if argc and argv aren't used. */
argc = argc;
argv = argv;

/* userlog writes to the central TUXEDO message log */

```

```

userlog("Welcome to the tpc server");
err_code.bytes_prov = 0;
QWCRDTAA (&data_area_input, 46, modn_dtaara, 1, 1, &err_code );
mod_number = atoi(data_area_input.data_block);
QWCRDTAA (&data_area_input, 48, clnt_dtaara, 1, 11, &err_code );
memcpy( clnt_ip1, &data_area_input.data_block, 11 );
clnt_ip1[12] = '\0';
QWCRDTAA (&data_area_input, 48, clnt_dtaara, 13, 11, &err_code );
memcpy( clnt_ip2, &data_area_input.data_block, 11 );
clnt_ip2[12] = '\0';
QWCRDTAA (&data_area_input, 48, srvr_dtaara, 1, 11, &err_code );
memcpy( srvr_ip1, &data_area_input.data_block, 11 );
srvr_ip1[12] = '\0';
QWCRDTAA (&data_area_input, 48, srvr_dtaara, 13, 11, &err_code );
memcpy( srvr_ip2, &data_area_input.data_block, 11 );
srvr_ip2[12] = '\0';

pid = getpid();
remain = pid % mod_number;
if(remain == 0) {
addr=inet_addr(clnt_ip1);
sd=bind(addr, pid);
addr=inet_addr(srvr_ip1);
sd2=bconnect(sd,addr,port);
}
else {
addr=inet_addr(clnt_ip2);
sd=bind(addr, pid);
addr=inet_addr(srvr_ip2);
sd2=bconnect(sd,addr,port);
}
return(0);
}

/* This function performs the actual service requested by the client.
Its argument is a structure containing among other things a pointer
to the data buffer, and the length of the data buffer.
*/

/* STOCK LEVEL */
STKLVL(TPSVCINFO *rqst)
{
if(send(sd, rqst->data, 230, 0)<0) {
perror("nsend() failed");
tpreturn(TPFAIL, 0, rqst->data, 0L, 0);
}
memset(rqst->data, 'V', 670);
rc=recv(sd, rqst->data, 670, 0);
if(rc<=0) {
perror("nrecv() failed");
tpreturn(TPFAIL, 0, rqst->data, 0L, 0);
}
/* Return the transformed buffer to the requestor. */
tpreturn(TPSUCCESS, 0, rqst->data, rc, 0);
}

```

## D.15 TUXCONFIG File:

```

#ident "8(#)apps:simpapp/ubbsimple 60.3"

#Skeleton UBBCONFIG file for the TUXEDO Simple Application.
#Replace the <bracketed> items with the appropriate values.

*RESOURCES
IPCKEY 123235

DOMAINID tpcapp
MASTER CLIENT10
MAXACCESSERS 9600
MAXSERVERS 300
MAXSERVICES 300
MODEL SHM
LDBAL Y
SCANUNIT 60
SANITYSCAN 60
BLOCKTIME 60
BBLQUERY 60
DBLWAIT 1

*MACHINES
"CLIENT10.RCHLAND.IBM.COM"
LMID=CLIENT10
APPDIR="/home/tpcckvclnt"
TUXCONFIG="/home/tpcckvclnt/tuxconfig"
TUXDIR="/qopensys/tuxsdk"

SPINCOUNT=2000

*GROUPS
DEFAULT:
srv1 GRPN=1
srv2 GRPN=11
srv3 GRPN=21
srv4 GRPN=31
srv5 GRPN=41
srv6 GRPN=51
srv7 GRPN=61

*SERVES
DEFAULT:
REPLYQ=Y
SRVGRP=srv1 SRVID=1 MIN=72
paymetsrv SRVGRP=srv2 SRVID=101 MIN=54
ordstssrv SRVGRP=srv3 SRVID=201 MIN=18
delvrsrv SRVGRP=srv4 SRVID=301 MIN=18 RQADDR=d1q1
delvrsrv SRVGRP=srv5 SRVID=351 MIN=18 RQADDR=d1q2
stkvlsvr SRVGRP=srv6 SRVID=401 MIN=72

*SERVICES
DEFAULT:
NEWORDER
PAYMNT
DELIVERY
STKLVL
ORDSTS

```

## SERVER:

## D.16 COMMON.H:

```

#ifdef _COMMON_H
#define _COMMON_H

#include <stdio.h>

```

```

#include <stdlib.h>
#include <string.h>
#include <spawn.h>
#include <qp021170.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <mih/mattod>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/msg.h>
#include <netinet6/in.h>
#include <errno.h>
#include <unistd.h>
#include <iconv.h>
#include <sys/uio.h>
#include <decimal.h>
#ifdef SEMAPHORE
typedef struct Semaphore {
    mutex_t    *mutex;
    mutex_lock_t xTemplate;
    int        xCount, xSleep;
} Semaphore_t;
#endif
#define BufferSize 3078
#define MAX_CONN 32

char *DLVRMOD(short *, char *);
char *NEWORMOD(short *, char *, char *);
char *ORDSTMOD(short *, char *, char *);
char *PAYMENTMOD(short *, char *, char *);
char *STKLVLMOD(short *, char *, char *);

extern int Listen(int port);
extern int Connect(int ipAddr, int port);
extern int Bind(int ipAddr, int port);
static short first_time=2; /* first time the operation runs */

#endif /* _COMMON_H */

```

## D.17 DELIVERY.CBL:

```

PROCESS NOTRUNC NORANGE.
IDENTIFICATION DIVISION.
    PROGRAM-ID. DLVRMOD.
    AUTHOR. TPCC.
    INSTALLATION. IBM-ROCHESTER.
    DATE-WRITTEN.
    DATE-COMPILED.
    ENVIRONMENT DIVISION.
    CONFIGURATION SECTION.
        SOURCE-COMPUTER. IBM-AS400.
        * SOURCE-COMPUTER. IBM-AS400 WITH DEBUGGING MODE.
        OBJECT-COMPUTER. IBM-AS400.
    INPUT-OUTPUT SECTION.
    FILE-CONTROL.
        SELECT NEWORD
            ASSIGN TO DATABASE-NEWORDLF
            ORGANIZATION IS INDEXED
            ACCESS MODE IS DYNAMIC
            RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
            WITH DUPLICATES
            FILE STATUS IS NEWORD-FILE-STATUS.
        SELECT DISTRICT-FILE
            ASSIGN TO DATABASE-DSTRCT
            ORGANIZATION IS INDEXED
            ACCESS MODE IS RANDOM
            RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
            WITH DUPLICATES
            FILE STATUS IS DISTRICT-FILE-STATUS.
        SELECT ORDERS
            ASSIGN TO DATABASE-ORDERSLF
            ORGANIZATION IS INDEXED
            ACCESS MODE IS RANDOM
            RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
            WITH DUPLICATES.
        SELECT ORDERLINE
            ASSIGN TO DATABASE-ORDLINLF
            ORGANIZATION IS INDEXED
            ACCESS MODE IS DYNAMIC
            RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
            WITH DUPLICATES.
        SELECT DLVRVLOG
            ASSIGN TO DATABASE-DLVRVLOG
            ORGANIZATION IS SEQUENTIAL.
    D SELECT ISO-IFILE
    D ASSIGN TO DATABASE-ISOIFILE
    D ORGANIZATION IS SEQUENTIAL
    D ACCESS IS SEQUENTIAL.
    D SELECT DBUG-FILE
    D ASSIGN TO FORMATFILE-DBGUPRT2
    D ORGANIZATION IS SEQUENTIAL
    D ACCESS IS SEQUENTIAL.
I-O-CONTROL.
    COMMITMENT CONTROL FOR ORDERS ORDERLINE
        DISTRICT-FILE NEWORD.

DATA DIVISION.
FILE SECTION.
FD NEWORD
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS NEWORD-RECORD.
01 NEWORD-RECORD.
    COPY DDS-WORCD OF NEWORDLF.
FD DISTRICT-FILE
    LABEL RECORDS ARE STANDARD.
01 DISTRICT-RECORD.
    COPY DDS-DSRCD OF DSTRCT.
FD ORDERS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS ORDERS-RECORD.
01 ORDERS-RECORD.
    COPY DDS-ORRCD OF ORDERSLF.
FD ORDERLINE
    LABEL RECORDS ARE STANDARD.
01 ORDERLINE-RECORD.
    COPY DDS-OLRCD OF ORDLINLF.
FD DLVRVLOG
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS DLVRVLOG-RECORD.
01 DLVRVLOG-RECORD.
    COPY DDS-LGRCD OF DLVRVLOG.
    05 LOG-REDEF REDEFINES LGRCD.
    06 TIMEA PIC S9(8).
    06 TIMEB PIC S9(8).
    06 UNDEL PIC S9.
    06 WID PIC S9(4) COMP-4.

DFD DBUG-FILE
D LABEL RECORDS ARE STANDARD
D DATA RECORD IS DBUG-REC2.
D 01 DBUGPRT-REC2.
D COPY DDS-DBGURCD2 OF DBUGPRT2.
D 05 DBUG-REC2 REDEFINES DBUGRC22-O.
D 06 DEBUG-DATA.
D 07 ISONUMBER PIC 99.
D 07 TXTDATA PIC X(40).

```

```

D 07 WID PIC S9(4).
D 07 DID PIC S9(2).
D 07 TIME-DATE.
D 08 DATE6 PIC S9(8).
D 08 TIME6 PIC S9(6).
D 07 CID PIC S9(4).
D 07 CWID PIC S9(4).
D 07 CDID PIC S9(2).
D 07 CBAL PIC S9(11)V99.
D 07 PAYMNT PIC S9(7)V99.
DFD ISO-IFILE
D LABEL RECORDS ARE STANDARD.
D01 ISOIFILE-RECORD.
D COPY DDS-ISOIRCD OF ISOIFILE.
D 05 ISO-IBEC REDEFINES ISOIRCD.
D 06 ISOINUM PIC S9(4) COMP-4.

```

### WORKING-STORAGE SECTION.

```

01 CSTMTR-RECORD.
    05 CUST-KEY.
        06 CID PIC S9(9) COMP-4.
        06 CDID PIC S9(4) COMP-4.
        06 CWID PIC S9(4) COMP-4.
        06 CUST-INFO.
            08 CFIRST PIC X(16).
            08 CINIT PIC X(02).
            08 CLAST PIC X(16).
            08 CLDATE PIC S9(8).
            08 CADDR1 PIC X(20).
            08 CCREDT PIC X(02).
            08 CADDR2 PIC X(20).
            08 CDCT PIC S9(9) COMP-3.
            08 CCITY PIC X(20).
            08 CSTATE PIC X(02).
            08 CZIP PIC X(9).
            08 CPHONE PIC X(16).
            08 CBAL PIC S9(11)V99 COMP-3.
            08 CCRDIM PIC S9(11)V99 COMP-3.
            08 CYTD PIC S9(11)V99 COMP-3.
            08 CPAYCNT PIC S9(3) COMP-3.
            08 CDELNT PIC S9(3) COMP-3.
            08 CDATA PIC X(500).

```

```

01 CSTMTR-KEY.
    03 CSTMTR-KEY-DATA OCCURS 5 TIMES.
        05 KRMW PIC X(10).
        05 KVAL PIC S9(9) COMP-4.
01 CSTMTR-HASH-PTR USAGE POINTER.
01 CSTMTR-HASH-NAME PIC X(10).
01 FUNCT PIC X(1).
01 KEYS PIC S9(9) COMP-4.
01 RET-CODE PIC S9(9) COMP-4.
01 RET-CODE-PTR USAGE POINTER.
01 CSTMTR-DEF-PTR USAGE POINTER.
01 CSTMTR-KEY-PTR USAGE POINTER.
D01 CBAL2 PIC S9(11)V99 COMP-3.
D01 ISOVALUE PIC S9(9) COMP-3 VALUE 0.
01 UPDATER PIC X(1) VALUE "3".
01 FETCHR PIC X(1) VALUE "1".
01 FETCHUPDATE PIC X(1) VALUE "2".
01 CUST-KEYS PIC 9(1) COMP-4 VALUE 3.

```

```

01 TRANSACTION-INPUT.
    06 TXN-TYPE PIC X.
    06 JOBNAM PIC X(10).
    06 CLIENT-INPUT PIC X(202).
    06 DLVRV-I REDEFINES CLIENT-INPUT.
    08 WID PIC S9(4) COMP-4.
    08 CARRIER PIC XX.
    08 D-TIME-OF-DAY PIC X(8).
01 TIME-OF-DAY PIC X(8).
01 TIME-PTR USAGE POINTER.
D 01 TIME-OF-DAY-STRING PIC X(14).
D 01 TORD PIC X(1) VALUE "b".
01 DQDATA-FRM-CLNT.
    05 DQDATA-WHS PIC S9(4) COMP-4.
    05 DQDATA-CARR PIC X(2).
    05 DQDATA-BTIM PIC S9(8).
    05 DQDATA-FILL PIC S9(6).
D 01 TESTNUM PIC 99 VALUE 1.
01 OLINES-LEFT PIC S9(2) COMP-4.
77 ORBAMT PIC 9(9)V99.

```

```

01 FILE-STATUS-ERROR-CDS.
    05 NEWORD-FILE-STATUS PIC X(2) VALUE "00".
    05 DISTRICT-FILE-STATUS PIC X(2) VALUE "00".
01 ORDERS-DELIVERED-TABLE.
    10 ORDER-LINE OCCURS 10 TIMES.
    15 ELEMENT-DISTRICT PIC S99.
    15 ELEMENT-ORDER PIC S9(6).
77 WHS PIC X(4).

```

```

LINKAGE SECTION.
01 FIRST-TIME PIC S9(4) COMP-4.
01 INP-TRAN PIC X(230).
PROCEDURE DIVISION USING FIRST-TIME INP-TRAN.

```

```

DELIVERY-PROGRAM.
    IF FIRST-TIME > 0
        PERFORM SET-UP-ROUTINE.
PROCESS-DELIVERY.
    MOVE INP-TRAN TO TRANSACTION-INPUT.
    MOVE DLVRV-I TO DQDATA-FRM-CLNT.
    MOVE D-TIME-OF-DAY OF DLVRV-I TO
        LOGTIMES OF DLVRVLOG-RECORD.

```

```

D READ ISO-IFILE.
D MOVE ISOINUM TO ISONUMBER, ISOVALUE.
MOVE 0 TO NODID.
MOVE 0 TO NOUID.
MOVE "0" TO UNDELST.
MOVE DQDATA-WHS TO NOWID, OWID, OLWID, CWID OF CSTMTR-RECORD.
MOVE DQDATA-WHS TO DWID.

```

```

IF DQDATA-CARR = "99" THEN
    STOP RUN.
MOVE DQDATA-CARR TO OCARDID.
CALL "gettime" USING BY VALUE TIME-PTR.

```

```

* THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
* FLAG IS SET AND IS FOR ISOLATION TEST USE

```

```

*
D MOVE "DEL PRE-READ " TO TXTDATA.
D MOVE 1 TO CUID OF DBUG-REC2.
D MOVE DQDATA-WHS TO CWID OF DBUG-REC2.
D MOVE 0 TO CID OF DBUG-REC2.
D MOVE 1 TO DID OF DBUG-REC2.
D MOVE 0 TO PAYMNT OF DBUG-REC2.
D MOVE DQDATA-WHS TO WID OF DBUG-REC2.
D MOVE "b" TO TORD.
D CALL "getdtmstr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
D TORD.
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC2.
D MOVE 0 TO CBAL OF DBUG-REC2.
D WRITE DBUGPRT-REC2 FORMAT IS "DBGRCRD2".
RE-DO-DLVRY.
PERFORM 10 TIMES

MOVE 0 TO ORDAMT
ADD 1 TO NODID
READ NEWORD
IF NEWORD-FILE-STATUS = "23"
MOVE NODID TO DID OF DISTRICT-RECORD
MOVE "00" TO NEWORD-FILE-STATUS
READ DISTRICT-FILE
IF DISTRICT-FILE-STATUS NOT EQUAL "00"
ROLLBACK
GOBACK
END-IF
READ NEWORD
END-IF
D IF NEWORD-FILE-STATUS = "23"
D CALL "gettime" USING BY VALUE TIME-PTR
D MOVE "b" TO TORD
D CALL "getdtmstr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
D TORD
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC2
D MOVE NODID TO DID OF DBUG-REC2
D MOVE "DEL NO NEWORD" TO TXTDATA
D WRITE DBUGPRT-REC2 FORMAT IS "DBGRCRD2"
D CALL "delayme"
D READ NEWORD
D CALL "gettime" USING BY VALUE TIME-PTR
D MOVE "b" TO TORD
D CALL "getdtmstr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
D TORD
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC2
D MOVE "DEL NO NEWORD2" TO TXTDATA
D WRITE DBUGPRT-REC2 FORMAT IS "DBGRCRD2"
D END-IF
D IF NEWORD-FILE-STATUS = "23"
D MOVE "00" TO NEWORD-FILE-STATUS
D MOVE "1" TO UNDELST
ELSE
MOVE NODID TO OIID, OIID, CID OF CSTMR-RECORD
MOVE NODID TO OIID, OIID
READ ORDERS
MOVE DQDATA-CARR TO OCAID
REWRITE ORDERS-RECORD
READ ORDERLINE
MOVE TIME-OF-DAY TO OLDLVOTD
REWRITE ORDERLINE-RECORD
ADD OLAMNT TO ORDAMT
SUBTRACT 1 FROM OLINES GIVING OLINES-LEFT
PERFORM OLINES-LEFT TIMES
READ ORDERLINE NEXT
MOVE TIME-OF-DAY TO OLDLVOTD
REWRITE ORDERLINE-RECORD
ADD OLAMNT TO ORDAMT
END-PERFORM

MOVE OCID TO CID OF CSTMR-RECORD
MOVE CID OF CSTMR-RECORD TO KVAL OF CSTMR-KEY-DATA(1)
MOVE CUID OF CSTMR-RECORD TO KVAL OF CSTMR-KEY-DATA(2)
MOVE CWID OF CSTMR-RECORD TO KVAL OF CSTMR-KEY-DATA(3)

CALL "qdbrunha" USING BY VALUE
CSTMR-HASH-PTR FETCHUPDATE CUST-KEYS CSTMR-KEY-PTR
CSTMR-DEF-PTR RET-CODE-PTR
IF RET-CODE > 0
IF RET-CODE = 812
ROLLBACK
MOVE 0 TO NODID
GO TO RE-DO-DLVRY
ELSE
IF RET-CODE = 100
ROLLBACK
GO TO MORE-ACTION
ELSE
ROLLBACK
GO TO PROGRAM-END
END-IF
END-IF
ADD ORDAMT TO CBAL OF CSTMR-RECORD
ADD 1 TO CDELNT
CALL "qdbrunha" USING BY VALUE
CSTMR-HASH-PTR UPDATER CUST-KEYS CSTMR-KEY-PTR
CSTMR-DEF-PTR RET-CODE-PTR
IF RET-CODE > 0
IF RET-CODE = 812
ROLLBACK
MOVE 0 TO NODID
GO TO RE-DO-DLVRY
ELSE
IF RET-CODE = 100
ROLLBACK
GO TO MORE-ACTION
ELSE
ROLLBACK
GO TO PROGRAM-END
END-IF
END-IF
DELETE NEWORD
MOVE OIID OF ORDERS-RECORD
TO ELEMENT-ORDER (NODID)

END-IF
END-PERFORM.

* THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
* FLAG IS SET AND IS FOR ISOLATION TEST USE
*
D IF ISOVALUE = 8 THEN GO TO DBG-SKP1.
D IF ISOVALUE = 6 THEN
D MOVE "DEL PRE-ROLLBK" TO TXTDATA
D ELSE IF ISOVALUE = 5 THEN
D MOVE "DEL PRE-COMMIT" TO TXTDATA.
D MOVE 1 TO CUID OF DBUG-REC2.
D MOVE DQDATA-WHS TO CWID OF DBUG-REC2.
D MOVE 1 TO DID OF DBUG-REC2.
D MOVE DQDATA-WHS TO WID OF DBUG-REC2.
D WRITE DBUGPRT-REC2 FORMAT IS "DBGRCRD2".
D CALL "delayme".
D CALL "gettime" USING BY VALUE TIME-PTR
D MOVE "b" TO TORD.

```

```

D CALL "getdtmstr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
D TORD.
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC2.
D IF ISOVALUE = 6 THEN
D ROLLBACK.
DBG-SKP1.
COMMIT.

* THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
* FLAG IS SET AND IS FOR ISOLATION TEST USE
*
D IF ISOVALUE = 8 THEN GO TO DBG-SKP2.
D IF ISOVALUE = 6 THEN
D MOVE "DEL POST-ROLLBK" TO TXTDATA
D ELSE IF ISOVALUE = 5 THEN
D MOVE "DEL POST-COMMIT" TO TXTDATA.
D MOVE 1 TO CUID OF DBUG-REC2.
D MOVE DQDATA-WHS TO CWID OF DBUG-REC2.
D MOVE 1 TO DID OF DBUG-REC2.
D MOVE 0 TO PAYMNT OF DBUG-REC2.
D MOVE DQDATA-WHS TO WID OF DBUG-REC2.
D MOVE CBAL2 TO CBAL OF DBUG-REC2.
D CALL "gettime" USING BY VALUE TIME-PTR.
D MOVE "b" TO TORD.
D CALL "getdtmstr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
D TORD.
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC2.
D WRITE DBUGPRT-REC2 FORMAT IS "DBGRCRD2".
DBG-SKP2.
MOVE DQDATA-WHS TO WH OF DLVRYLOG-RECORD.
CALL "gettime" USING BY VALUE TIME-PTR.
MOVE TIME-OF-DAY TO LOGTIMEA.
WRITE DLVRYLOG-RECORD.
MORE-ACTION.
GOBACK.

SET-UP-ROUTINE.
PERFORM CLOSE-ROUTINE.
OPEN EXTEND DLVRYLOG.
OPEN I-O ORDERLINE.
OPEN I-O ORDERS.
OPEN I-O DISTRICT-FILE.
OPEN I-O NEWORD.
D OPEN OUTPUT DBUG-FILE.
D OPEN INPUT ISO-IFILE.
MOVE "CSTMR" TO CSTMR-HASH-NAME.
MOVE "CID" TO KNAM OF CSTMR-KEY-DATA(1).
MOVE "CUID" TO KNAM OF CSTMR-KEY-DATA(2).
MOVE "CWID" TO KNAM OF CSTMR-KEY-DATA(3).
SET CSTMR-HASH-PTR TO ADDRESS OF CSTMR-HASH-NAME.
SET RET-CODE-PTR TO ADDRESS OF RET-CODE.
SET CSTMR-DEF-PTR TO ADDRESS OF CSTMR-RECORD.
SET CSTMR-KEY-PTR TO ADDRESS OF CSTMR-KEY.
SET TIME-PTR TO ADDRESS OF TIME-OF-DAY.

CLOSE-ROUTINE.
CLOSE NEWORD
CLOSE DISTRICT-FILE
CLOSE ORDERS
CLOSE ORDERLINE
CLOSE DLVRYLOG.
D CLOSE DBUG-FILE.
D CLOSE ISO-IFILE.
PROGRAM-END.
STOP RUN.

#include <Common.h>
#include <qsoasynch.h>
#include <stdio.h>

main(int argc, char **argv)
{
int sd;/* the input socket that is doing the listen */
int snd_len=670, rcv_len=230, port, rc;
int sd_arr[MAX_CONN]/* the indexes for accepts and use */
int num_conn;/* the number of connections */
int accept_index;/* accept index goes from 0 to num_conn - 1 */
int retry_count;/* number of retries attempted */
char rcv_buf[MAX_CONN] [BufferSize]/* one per connection */
struct sockaddr_in sin;
int length;
Qso_OverlappedIo_t status;
char memset(status,0,sizeof(status));
status.bufferLength=rcv_len;
status.postFlag=1; /* Always post to I/O completion port */
/* status.fillBuffer=1; /* Fill buffer. */
/* get the number of connections */
num_conn = atoi(argv[1]);
/* printf("The number of deliver connections is %d\n", num_conn ); */
/* num_conn = 2; */ /* test action */
if((port=QsoCreateIOCompletionPort())<0) {
perror("\nQsoCreateIOCompletionPort() failed");
exit(-1);
}
length=sizeof(sin);
for( accept_index=0;accept_index<num_conn;accept_index++)
{
if((sd_arr[accept_index]=accept(sd,
(struct sockaddr *)&sin,length))<0)
{
perror("accept() in DLVRSVR failed");
close(sd);
exit(-1);
}
status.descriptorHandle=(void*)sd_arr[accept_index];
status.buffer=rcv_buf[accept_index];
if(QsoStartRecv(sd_arr[accept_index],port,&status)<0) {
perror("\nQsoStartRecv() failed");
exit(-1);
}
}/* end of connect loop */
{
if(QsoWaitForIOCompletion(port,&status,0)<0)
{
perror("\nQsoWaitForIOCompletion() failed");
exit(-1);
}
if(status.returnValue<=0) {
printf("\nRecv() failed, errno=%d",status.errnoValue);
exit(-1);
}
if(status.operationCompleted!=QSOSTARTRECV) {
printf("\nUnexpected completion=%d",status.operationCompleted);
exit(-1);
}
}
}

```

## D.18 DLVRSVR.C:

## D.20 NEWORDER.CBL:

```
system("STRMCTL LCKLVL(*ALL)");
system("CHGJOB RUNPT(34)");
system("OVRDBF FILE(NEWORLDF) TOFILE(NEWORLDF) NBRRCDS(10)");
system("OVRDBF FILE(ORLNLNLF) TOFILE(ORLNLNLF) NBRRCDS(120)");
DLVRVMD(4first_time, status.buffer);/* first transaction */
first_time++;
if(QsoStartRecv((int)status.descriptorHandle,port,&status)<0) {
    perror("\nQsoStartRecv() failed");
    exit(-1);
}
}
while ( first_time > 0 )
for(;;) {
    if(QsoWaitForIOCompletion(port,&status,0)<0) {
        perror("\nQsoWaitForIOCompletion() failed");
        exit(-1);
    }
    if(status.returnValue<=0) {
        printf("\nRecv() failed, errno=%d",status.errnoValue);
        exit(-1);
    }
    if(status.operationCompleted!=QSOSTARTRECV) {
        printf("\nUnexpected completion=%d",status.operationCompleted);
        exit(-1);
    }
    DLVRVMD(4first_time, status.buffer);/* normal transaction */
    if(QsoStartRecv((int)status.descriptorHandle,port,&status)<0) {
        perror("\nQsoStartRecv() failed");
        exit(-1);
    }
}
} /* for loop */
return 0;
}
```

## D.19 GETTIME.C:

```
/*.....*/
/*
/* OCO Source Materials
/*
/* The Source code for this program is not published or otherwise
/* divested of its trade secrets, irrespective of what has been
/* deposited with the U.S. Copyright office
/*
/* TPCCCLIENT LIBRARY
/* (C) Copyright IBM Corp. 1997,1998
/*.....*/

#pragma comment (copyright, \
"TPCCCLIENT LIBRARY\
(C) Copyright IBM Corp. 1997,1998.\
All rights reserved.\
US Government Users Restricted Rights-\
Use, duplication or disclosure restricted \
by GSA ADP Schedule Contract with IBM Corp.\
Licensed Materials-Property of IBM")

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <leawi.h>
#include <QSYSINC/MIH/MAT70D>
#include <qwordss.h>
#define BASE_OFFSET 4503599627.370496
#define OFFSET 10914267600.0 /* set to a fixed value */
/* Define the time range flag */
#define HIGH_FLAG 'B'
/* #include <CEDATM> */
struct dateTime {
    _MI_Time returnTimestamp;
};
void gettime(struct dateTime * new_date_time)
{
    mattod(new_date_time->returnTimestamp);/* get the time of day */
    new_date_time->returnTimestamp[7] = HIGH_FLAG;/* set the high flag */
}

void gettime_string(char * timein, char * timestamp )
{
    double converted = 0;/* the output value */
    double seconds;/* number of seconds in floating point since 14 October 1582 */
    double scale=0;/* scale factor from MI_Time to seconds */
    double second = 16000000;/* number of MI units in one second */
    double offset;/*number of seconds from MI_base to seconds base */
    OFFSET;
    /* note this has a one hour offset for daylight time */
    /* base date is Aug 23, 1928 noon */
    /* the daylight time offset gives 1:00 PM time */
    double base =1;/* the base for the value used */
    double Base_sec = 0;/* the base for the value used */
    char picture[] = "MM DD, YYYY HH:MI:SS AP.,";/* timestamp definition */
    _FEEDBACK fb;/* feedback value */
    int count;/* loop counter */
    char high_flag;
    high_flag = timein[7];/* get the high byte flag */
    if ( high_flag != 'B' )
    {
        if (( high_flag == 'A' ) && (( timein[0]&0xF0 ) > 0x70))
        /* first part of the A date range this section is smaller than */
        /* the base section
        {
            offset = OFFSET - BASE_OFFSET; /* next lower range */
        }
        else
        {
            if (( high_flag == 'C' ) && (( timein[0]&0xF0 ) < 0x80))
            /* second part of the C date range this section is larger than */
            {
                offset = OFFSET + BASE_OFFSET; /* next higher range */
            }
        }
    }
    for ( count = 0; count < 7; count++)
    {
        converted = converted + ((timein[6-count]&255) * base);
        base = base * 256;
    }
    /* printf ("second is %f\n", second) ; */
    seconds = (converted/second) + offset;
    memset (timestamp, ' ', 27) ;/* reset time stamp */
    CEDATM ( &seconds,&picture[0],timestamp,&fb);/* convert to timestamp */
    timestamp[26] = '\0';/* terminate the time stamp */
}
void printtime(char * timein )
{
    char time_stamp[27];/* storage for text */
    gettime_string(timein, time_stamp);/* get the time stamp */
    printf ( "TIME: %s\n", time_stamp) ;/* print output */
}
}
```

```
PROCESS NOTRUNC NORANGE.
IDENTIFICATION DIVISION.
PROGRAM-ID. NEWORMOD.
AUTHOR. TPCC.
INSTALLATION. IBM-ROCHESTER.
DATE-WRITTEN.
DATE-COMPILED.
*.....*
*
* PART NAME: NEWORDER
*
*.....*
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-AS400.
* SOURCE-COMPUTER. IBM-AS400 WITH DEBUGGING MODE.
OBJECT-COMPUTER. IBM-AS400.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT WAREHOUSE-FILE ASSIGN TO DATABASE-WRHS
ORGANIZATION IS INDEXED
ACCESS MODE IS RANDOM
RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
WITH DUPLICATES
FILE STATUS IS WAREHOUSE-FILE-STATUS.
SELECT DISTRICT-FILE ASSIGN TO DATABASE-DSTRCT
ORGANIZATION IS INDEXED
ACCESS MODE IS RANDOM
RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
WITH DUPLICATES
FILE STATUS IS DISTRICT-FILE-STATUS.
SELECT ITEM-FILE ASSIGN TO DATABASE-ITEM
ORGANIZATION IS INDEXED
ACCESS MODE IS RANDOM
RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
WITH DUPLICATES
FILE STATUS IS ITEM-FILE-STATUS.
SELECT NEWORDER-FILE ASSIGN TO DATABASE-NEWORD
ORGANIZATION IS SEQUENTIAL
FILE STATUS IS NEWORDER-FILE-STATUS.
SELECT ORDERS-FILE ASSIGN TO DATABASE-ORDERSLF
ORGANIZATION IS SEQUENTIAL.
SELECT ORDERLINE-FILE ASSIGN TO DATABASE-ORLINF
ACCESS MODE IS SEQUENTIAL.
D SELECT DBUG-FILE
D ASSIGN TO FORMATFILE-DBGUPRT
D ORGANIZATION IS SEQUENTIAL
D ACCESS IS SEQUENTIAL.
D SELECT DBUG-FILE2
D ASSIGN TO FORMATFILE-DBGUPRT2
D ORGANIZATION IS SEQUENTIAL
D ACCESS IS SEQUENTIAL.
D SELECT ISO-FILE
D ASSIGN TO DATABASE-ISOFILE
D ORGANIZATION IS SEQUENTIAL
D ACCESS IS SEQUENTIAL.
I-O-CONTROL.
COMMITMENT CONTROL FOR WAREHOUSE-FILE
COMMITMENT CONTROL FOR DISTRICT-FILE
COMMITMENT CONTROL FOR ITEM-FILE
COMMITMENT CONTROL FOR NEWORDER-FILE
COMMITMENT CONTROL FOR ORDERS-FILE
COMMITMENT CONTROL FOR ORDERLINE-FILE.
DATA DIVISION.
FILE SECTION.
FD WAREHOUSE-FILE
LABEL RECORDS ARE STANDARD.
01 WAREHOUSE-REC.
COPY DDS-WRRCD OF WRHS.
05 WRHS-INFO REDEFINES WRRCD.
06 FILLER PIC X(12).
06 WH-ADDRESS.
08 WADDR1 PIC X(20).
08 WADDR2 PIC X(20).
08 CITYW PIC X(20).
08 STATEW PIC X(2).
08 ZIPW PIC X(10).
FD DISTRICT-FILE
LABEL RECORDS ARE STANDARD.
01 DISTRICT-REC.
COPY DDS-DSRCD OF DSTRCT.
05 DISTRICT-INFO REDEFINES DSRCD.
06 DIST-KEY.
08 DID PIC S9(4) COMP-4.
08 DWID PIC S9(4) COMP-4.
06 FILLER PIC X(10).
06 DIST-ADDRESS.
08 DADDR1 PIC X(20).
08 DADDR2 PIC X(20).
08 CITY PIC X(20).
08 STATE PIC X(2).
08 ZIP PIC X(10).
FD ITEM-FILE
LABEL RECORDS ARE STANDARD.
01 ITEM-REC.
COPY DDS-ITRCD OF ITEM.
05 ITEM-DEF REDEFINES ITRCD.
06 ID-ID PIC S9(9) COMP-4.
06 ID-IMID PIC S9(9) COMP-4.
06 ITEM-INFO.
08 ID-NAME PIC X(24).
08 ID-IPRICE PIC S9(3)V99 COMP-3.
06 ID-DATA PIC X(50).
FD NEWORDER-FILE
LABEL RECORDS ARE STANDARD.
01 NEWORDER-REC.
COPY DDS-NORCD OF NEWORD.
FD ORDERS-FILE
LABEL RECORDS ARE STANDARD.
01 ORDERS-REC.
COPY DDS-ORCD OF ORDERSLF.
FD ORDERLINE-FILE
LABEL RECORDS ARE STANDARD.
01 ORDERLINE-REC.
COPY DDS-OLRCD OF ORLINF.
05 ORDERLIN-INPUT REDEFINES OLRCD.
07 ORDER-KEY.
08 OLOID PIC S9(9) COMP-4.
08 OLOID PIC S9(4) COMP-4.
08 OLWID PIC S9(4) COMP-4.
08 OLNBR PIC S9(3) COMP-3.
07 OL-INPUT.
08 OL-INPUT-OLSPWH PIC S9(4) COMP-4.
08 OL-INPUT-OLIID PIC S9(9) COMP-4.
08 OL-INPUT-QTY PIC S9(3) COMP-3.
07 FILLER PIC X(4).
07 OLTIME-OF-DAY PIC X(8).
07 TIME-FILLER PIC X(6).
DFD DBUG-FILE
D LABEL RECORDS ARE STANDARD
D DATA RECORD IS DBUG-REC.
D 01 DBUGPRT-REC.
D COPY DDS-DBGURCD OF DBGUPRT.
D 05 DBUG-REC REDEFINES DBGURCD-O.
D 06 DBUG-DATA.
```

```

D          07 ISONUM          PIC XX.
D          07 TXTDATA        PIC X(40).
D          07 WID            PIC S9(4).
D          07 DID            PIC S9(2).
D          07 TIME-DATE.
D          08 DATE          PIC S9(8).
D          08 TIME          PIC S9(6).
D          07 CID            PIC S9(4).
D          07 OLD            PIC S9(8).
D          07 DEBUG-ITEM-INFO OCCURS 15 TIMES.
D          08 DEBUG-OLSPWH  PIC S9(4).
D          08 DEBUG-OLIID   PIC S9(6).
D          08 DEBUG-OLQTY   PIC S9(3).
D          08 DEBUG-IPRICE  PIC S9(5)V9(2).
DFD DBUG-FILE2
D LABEL RECORDS ARE STANDARD
D DATA RECORD IS DBUG-REC2.
D 01 DBUGPRT-REC2.
D COPY DDS-DEBUGRCD2 OF DBUGPRT2.
D 05 DBUG-REC2 REDEFINES DEBUGRCD2-O.
D 06 DBUG-DATA.
D 07 ISONUM          PIC XX.
D 07 TXTDATA        PIC X(40).
D 07 WID            PIC S9(4).
D 07 DID            PIC S9(2).
D 07 TIME-DATE.
D 08 DATE          PIC S9(8).
D 08 TIME          PIC S9(6).
D 07 CID            PIC S9(4).
D 07 CWID           PIC S9(4).
D 07 CDID           PIC S9(2).
D 07 CBAL           PIC S9(11)V99.
D 07 PAYMNT        PIC S9(7)V99.
DFD ISO-IFILE
D LABEL RECORDS ARE STANDARD.
D01 ISOFILE-RECORD.
D COPY DDS-ISOIFRCD OF ISOIFILE.
D 05 ISO-IREC REDEFINES ISOIFRCD.
D 06 ISOINUM PIC S9(4) COMP-4.

WORKING-STORAGE SECTION.
01 TRAN-OUTPUT.
05 TRAN-OUT PIC X(670).

05 NEWORD-O REDEFINES TRAN-OUT.
06 NEWORD-RESULT PIC X.
06 OENTTM        PIC S9(6).
06 CLAST         PIC X(16).
06 CCREDT        PIC X(2).
06 CDCT          PIC S999V99 COMP-3.
06 OLD           PIC S9(9) COMP-4.
06 WTX          PIC S999999 COMP-3.
06 DTAX         PIC S999999 COMP-3.
06 TOTAMT       PIC S9(9)V9(2) COMP-3.
06 OUTPUT-TABLE.
10 OUTPUT-LINE OCCURS 15 TIMES.
15 OUTPUT-STQTY PIC S9(4) COMP-4.
15 OUTPUT-BORG  PIC X(1).
15 OUTPUT-ITEM-INFO.
20 OUTPUT-NAME PIC X(24).
20 OUTPUT-IPRICE PIC S9(3)V9(2) COMP-3.
15 OUTPUT-OLAMNT PIC S9(5)V9(2) COMP-3.
06 OENTTOD     PIC X(8).

01 STOCK-KEY-PTR USAGE POINTER.
01 CSTMR-KEY-PTR USAGE POINTER.

01 STOCK-KEY.
03 STOCK-KEY-DATA OCCURS 5 TIMES.
05 KNAM PIC X(10).
05 KVAL PIC S9(9) COMP-4.

01 CSTMR-KEY.
03 CSTMR-KEY-DATA OCCURS 5 TIMES.
05 KNAM PIC X(10).
05 KVAL PIC S9(9) COMP-4.

01 STOCK-HASH-NAME PIC X(10).
01 CSTMR-HASH-NAME PIC X(10).

01 STOCK-HASH-PTR USAGE POINTER.
01 CSTMR-HASH-PTR USAGE POINTER.

01 FUNCT PIC X(1).
01 KEYS  PIC S9(9) COMP-4.
01 RET-CODE PIC S9(9) COMP-4.

01 RET-CODE-PTR USAGE POINTER.

01 STOCK-DEF-PTR USAGE POINTER.
01 CSTMR-DEF-PTR USAGE POINTER.

01 UPDATER PIC X(1) VALUE "3".
01 FETCHR  PIC X(1) VALUE "1".
01 FETCHUPDATE PIC X(1) VALUE "2".

01 CUST-KEYS PIC 9(1) COMP-4 VALUE 3.
01 STOCK-KEYS PIC 9(1) COMP-4 VALUE 2.

01 CUSTOMER-REC.
05 CUST-KEY.
06 CID PIC S9(9) COMP-4.
06 CDID PIC S9(4) COMP-4.
06 CWID PIC S9(4) COMP-4.
06 CUST-INFO.
08 CFIRST PIC X(16).
08 CINIT  PIC X(02).
08 CLAST  PIC X(16).
08 CLDATE PIC S9(8).
08 CADDR1 PIC X(20).
08 CCREDT PIC X(02).
08 CADDR2 PIC X(20).
08 CDCT   PIC S999999 COMP-3.
08 CCITY  PIC X(20).
08 CSTATE PIC X(02).
08 CZIP   PIC X(9).
08 CPHONE PIC X(16).
08 CRAL   PIC S9(11)V99 COMP-3.
08 CCRDIM PIC S9(11)V99 COMP-3.
08 CYTD   PIC S9(11)V99 COMP-3.
08 CPAYCNT PIC S9(3) COMP-3.
08 COELCNT PIC S9(3) COMP-3.
08 CDATA  PIC X(500).

01 STOCK-REC.
05 STOCK-DEF.
06 STOCK-IN.
08 STIID1 PIC S9(9) COMP-4.
08 STWID1 PIC S9(4) COMP-4.
08 STQTY  PIC S9(3) COMP-3.
06 DIST-INFO OCCURS 10 TIMES.
07 DIST-IN PIC X(24).
06 STYTD  PIC S9(9) COMP-4.
06 STORDS PIC S9(3) COMP-3.
06 STREMORD PIC S9(3) COMP-3.
06 STDATA PIC X(50).

01 TRANSACTION-INPUT.
06 TXN-TYPE PIC X.
06 JOBNAME  PIC X(10).
06 CLIENT-INPUT PIC X(202).
06 NEWORD-I REDEFINES CLIENT-INPUT.
08 CWID PIC S9(4) COMP-4.
08 CDID PIC S9(4) COMP-4.
08 CID  PIC S9(9) COMP-4.
08 NUMBER-OF-ITEMS PIC 9(3) COMP-3.
08 INPUT-LINE OCCURS 15 TIMES.
10 OLSPWH PIC S9(4) COMP-4.
10 OLIID  PIC S9(9) COMP-4.
10 OLQTY  PIC S9(3) COMP-3.

01 INPUT-ROW.
05 STOCK-KEY-INPUT.
10 OLIID PIC S9(9) COMP-4.
10 OLSPWH PIC S9(4) COMP-4.
05 OLQTY PIC S9(3) COMP-3.

01 ROLL-BACK-REQUIRED PIC X(1) VALUE '0'.
01 CONTR-1 PIC S99 COMP-4.

D 01 TESTNUM PIC 99 VALUE 1.
D 01 J PIC S999 COMP-3.

01 DATETIME-INIT.
05 FPCDRT-INIT PIC X(8) VALUE "00000000".
01 DATETIME-CHARINIT REDEFINES DATETIME-INIT PIC X(8).

01 TIME-OF-DAY PIC X(8).

D 01 TIME-OF-DAY-STRNG PIC X(14).

D 01 TORD PIC X(1) VALUE "b".

01 ERROR-HDLNG-PARAMETERS.
05 NEWORDER-FILE-STATUS PIC X(2) VALUE "00".
05 ITEM-FILE-STATUS PIC X(2) VALUE "00".
05 DISTRICT-FILE-STATUS PIC X(2) VALUE "00".
05 WAREHOUSE-FILE-STATUS PIC X(2) VALUE "00".
05 ORDLIN-FILE-STATUS PIC X(2) VALUE "00".

01 I PIC S9(3) COMP-3.

LINKAGE SECTION.
01 FIRST-TIME PIC S9(4) COMP-4.
01 INP-TRAN PIC X(230).
01 OUT-TRAN PIC X(670).

PROCEDURE DIVISION USING FIRST-TIME INP-TRAN OUT-TRAN.

DECLARATIVES.

HANDLE-ERROR SECTION.

I-O-ERROR-PARA.

END DECLARATIVES.

MAIN-LINE-ROUTINE.

IF FIRST-TIME > 0
PERFORM SET-UP-ROUTINE.

MOVE INP-TRAN TO TRANSACTION-INPUT.

D READ ISO-IFILE.
D MOVE ISOINUM TO TESTNUM, ISONUM OF DBUG-REC, ISONUM OF
D DBUG-REC2.

NEW-ORDER-TRANSACTION.

CALL "gettime" USING TIME-OF-DAY.

MOVE CWID OF NEWORD-I TO CWID OF CUSTOMER-REC
WID OF WAREHOUSE-REC
DWID OF DSRCD
NOWID OF NEWORDER-REC
OWID OF ORDERS-REC.

MOVE CDID OF NEWORD-I TO
CDID OF CUSTOMER-REC
DID OF DSRCD
NODID OF NEWORDER-REC
ODID OF ORDERS-REC.

MOVE TIME-OF-DAY TO OENTTOD OF ORDERS-REC.
MOVE TIME-OF-DAY TO OENTTOD OF NEWORD-O.

MOVE CID OF NEWORD-I TO CID OF CUSTOMER-REC
OCID OF ORDERS-REC.

MOVE NUMBER-OF-ITEMS TO OLINES OF ORDERS-REC.
MOVE 1 TO OLOCAL OF ORDERS-REC.

MOVE 0 TO TOTAMT.

RE-DO-NEWORD.

* THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
* FLAG IS SET AND IS FOR ISOLATION TEST USE
*
D MOVE "NEW ORDER PRE-READ " TO TXTDATA OF DBUG-REC.
D MOVE 9999 TO OLD OF DBUG-REC.
D PERFORM GET-ISOLATION-DATA-1.
D WRITE DBUGPRT-REC FORMAT IS "DEBUGRCD".

MOVE CID OF CUSTOMER-REC TO KVAL OF CSTMR-KEY-DATA(1)
MOVE CDID OF CUSTOMER-REC TO KVAL OF CSTMR-KEY-DATA(2)
MOVE CWID OF CUSTOMER-REC TO KVAL OF CSTMR-KEY-DATA(3)

CALL "qsbbrunha" USING BY VALUE
CSTMR-HASH-PTR FETCHUPDATE CUST-KEYS CSTMR-KEY-PTR
CSTMR-DEF-PTR RET-CODE-PTR

IF RET-CODE > 0
IF RET-CODE = 100
MOVE "2" TO ROLL-BACK-REQUIRED
GO TO ROLLBACK-PARA
ELSE
IF RET-CODE = 812
ROLLBACK
GO TO RE-DO-NEWORD
ELSE
ROLLBACK
GO TO PROGRAM-END
END-IF
END-IF

ELSE

READ DISTRICT-FILE
END-READ

```

```

IF DISTRICT-FILE-STATUS NOT EQUAL "00"
  IF DISTRICT-FILE-STATUS EQUAL "9D"
    ROLLBACK
    GO TO RE-DO-NEWORD
  ELSE
    MOVE "2" TO ROLL-BACK-REQUIRED
    GO TO ROLLBACK-PARA
  END-IF
ELSE
  MOVE DNXTOR OF DISTRICT-REC TO OID OF ORDERS-REC
  MOVE NOID OF NEWORDER-REC
  NOID OF NEWORDER-REC

PERFORM VARYING I FROM 1 BY 1 UNTIL I > NUMBER-OF-ITEMS
  MOVE OLSFWH OF INPUT-LINE(I) TO OLSFWH OF INPUT-ROW
  MOVE OLIID OF INPUT-LINE(I) TO OLIID OF INPUT-ROW
  MOVE OLQTY OF INPUT-LINE(I) TO OLQTY OF INPUT-ROW
  MOVE OLIID OF INPUT-ROW TO IID OF ITEM-REC

READ ITEM-FILE
IF ITEM-FILE-STATUS NOT EQUAL "00"
  MOVE "1" TO ROLL-BACK-REQUIRED
  GO TO ROLLBACK-PARA
ELSE
  MOVE ITEM-INFO TO OUTPUT-ITEM-INFO(I)
  MOVE ZERO TO CONTR-1
  INSPECT IDATA OF ITEM-REC
  TALLYING CONTR-1 FOR ALL "ORIGINAL"
  MOVE STOCK-KEY-INPUT TO STOCK-IN

MOVE STWID1 TO KVAL OF STOCK-KEY-DATA(2)
MOVE STIID1 TO KVAL OF STOCK-KEY-DATA(1)

CALL "qdbrunha" USING BY VALUE
  STOCK-HASH-PTR RECHURPATE STOCK-KEYS STOCK-KEY-PTR
  STOCK-DEF-PTR RET-CODE-PTR
IF RET-CODE > 0
  IF RET-CODE = 100
    MOVE "2" TO ROLL-BACK-REQUIRED
    GO TO ROLLBACK-PARA
  ELSE
    IF RET-CODE = 812
      ROLLBACK
      GO TO RE-DO-NEWORD
    ELSE
      ROLLBACK
      GO TO PROGRAM-END
    END-IF
  END-IF
ELSE
  * THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
  * FLAG IS SET AND IS FOR ISOLATION TEST USE
  *
  D IF TESTNUM = 7 THEN
  D IF I = 1 THEN
  D CALL "delayme"
  D END-IF
  D END-IF

  SUBTRACT OLQTY OF INPUT-ROW
  FROM STQTY OF STOCK-REC
  IF STQTY OF STOCK-REC < 10 THEN
  ADD 91 TO STQTY OF STOCK-REC
  END-IF

  IF (OLSPWH OF INPUT-ROW NOT EQUAL CWID OF
  NEWORD-I)
  MOVE 0 TO OLOCAL OF ORDERS-REC
  ADD 1 TO STREMOD OF STOCK-REC
  END-IF

  ADD 1 TO STORDS OF STOCK-REC
  ADD OLQTY OF INPUT-ROW TO STYTD OF STOCK-REC

CALL "qdbrunha" USING BY VALUE
  STOCK-HASH-PTR UPDATER STOCK-KEYS STOCK-KEY-PTR
  STOCK-DEF-PTR RET-CODE-PTR
IF RET-CODE > 0 THEN
  IF RET-CODE = 1 THEN
    ROLLBACK
    GO TO PROGRAM-END
  END-IF
  IF RET-CODE = 2 THEN
    ROLLBACK
  END-IF
END-IF
IF CONTR-1 > ZERO THEN
  MOVE ZERO TO CONTR-1
  INSPECT STDATA OF STOCK-REC
  TALLYING CONTR-1 FOR ALL "ORIGINAL"
END-IF
IF CONTR-1 > ZERO
  THEN MOVE "B" TO OUTPUT-BORG(I)
  ELSE MOVE "G" TO OUTPUT-BORG(I)
END-IF
MULTIPLY OLQTY OF INPUT-ROW BY IPRICE
  GIVING OUTPUT-OLAMNT(I)
ADD OUTPUT-OLAMNT(I) TO TOTAMT
MOVE NORCD TO ORDER-KEY
MOVE I TO OLNBR OF OLRCDD
MOVE INPUT-ROW TO OL-INPUT
MOVE OLIID OF INPUT-ROW TO OL-INPUT-OLIID
MOVE OLSFWH OF INPUT-ROW TO OL-INPUT-OLSPWH
MOVE OLQTY OF INPUT-ROW TO OL-INPUT-OLQTY
MOVE OUTPUT-OLAMNT(I) TO OLAMNT OF OLRCDD
  OF ORDERLINE-REC
MOVE DIST-INFO(CCID OF NEWORD-I)
  TO OLDSTI OF OLRCDD
  OF ORDERLINE-REC
MOVE DATETIME-CHARINIT TO OLTIME-OF-DAY
WRITE ORDERLINE-REC
MOVE STQTY OF STOCK-REC TO OUTPUT-STQTY(I)
END-IF
END-IF
END-PERFORM
END-IF
END-IF
D MOVE OID OF ORDERS-REC TO OID OF DBUG-REC.
ADD 1 TO DNXTOR OF DISTRICT-REC.
REWRITE DISTRICT-REC.
WRITE ORDERS-REC.

WRITE NEWORDER-REC.
READ-WAREHOUSE.
READ WAREHOUSE-FILE.
IF WAREHOUSE-FILE-STATUS NOT EQUAL "00"
  IF WAREHOUSE-FILE-STATUS EQUAL "9D"
    GO TO READ-WAREHOUSE
  ELSE
    MOVE "2" TO ROLL-BACK-REQUIRED
    END-IF
  END-IF
ROLLBACK-PARA.
IF ROLL-BACK-REQUIRED NOT EQUAL "0"
  THEN
  * THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
  * FLAG IS SET AND IS FOR ISOLATION TEST USE
  *
  D MOVE "NEW ORDER PRE-RBACK" TO TXTDATA OF DBUG-REC
  D PERFORM GET-ISOLATION-DATA-1
  D WRITE DBUGPRT-REC FORMAT IS "DBUGRCD"
  D CALL "delayme"
  ROLLBACK
  * THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
  * FLAG IS SET AND IS FOR ISOLATION TEST USE
  *
  D MOVE "NEW ORDER POST-RBACK" TO TXTDATA OF DBUG-REC
  D PERFORM GET-ISOLATION-DATA-2
  D WRITE DBUGPRT-REC FORMAT IS "DBUGRCD"
  D CLOSE DBUG-FILE
  D OPEN OUTPUT DBUG-FILE
  MOVE CLAST OF CUSTOMER-REC
  TO CLAST OF NEWORD-O
  MOVE CCREDIT OF CUSTOMER-REC
  TO CCREDIT OF NEWORD-O
  MOVE OID OF ORDERS-REC
  TO OID OF NEWORD-O
  IF ROLL-BACK-REQUIRED = "1"
    PERFORM GET-ISOLATION-DATA-2.
  ELSE IF ROLL-BACK-REQUIRED = "2"
    MOVE "I" TO NEWORD-RESULT OF NEWORD-O
  END-IF
  END-IF
  MOVE "0" TO ROLL-BACK-REQUIRED
  GO TO RETURN-DATA
  ELSE
  * THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
  * FLAG IS SET AND IS FOR ISOLATION TEST USE
  *
  D MOVE "NEW ORDER PRE-COMMIT" TO TXTDATA OF DBUG-REC
  D PERFORM GET-ISOLATION-DATA-1
  D WRITE DBUGPRT-REC FORMAT IS "DBUGRCD"
  D CALL "delayme"
  COMMIT.
  * THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
  * FLAG IS SET AND IS FOR ISOLATION TEST USE
  *
  D MOVE "NEW ORDER POST-COMIT" TO TXTDATA OF DBUG-REC.
  D PERFORM GET-ISOLATION-DATA-2.
  D WRITE DBUGPRT-REC FORMAT IS "DBUGRCD".
  D CLOSE DBUG-FILE.
  D OPEN OUTPUT DBUG-FILE.
  MOVE "G" TO NEWORD-RESULT OF NEWORD-O
  MOVE CLAST OF CUSTOMER-REC
  TO CLAST OF NEWORD-O
  MOVE CCREDIT OF CUSTOMER-REC
  TO CCREDIT OF NEWORD-O
  MOVE CDCT OF CUSTOMER-REC TO CDCT OF NEWORD-O
  MOVE WTAX OF WAREHOUSE-REC TO WTAX OF NEWORD-O
  MOVE DTAX OF DISTRICT-REC TO DTAX OF NEWORD-O
  MOVE OID OF ORDERS-REC TO
  OID OF NEWORD-O.
  GO TO RETURN-DATA.
NEW-ORDER-TRANSACTION-EXIT.
EXIT.
D GET-ISOLATION-DATA-1.
D MOVE CCID OF NEWORD-I TO DID OF DBUG-REC
D MOVE CID OF NEWORD-I TO CID OF DBUG-REC
D MOVE CWID OF NEWORD-I TO WID OF DBUG-REC
D CALL "gettime" USING TIME-OF-DAY
D MOVE "b" TO TORD
D CALL "getdtmatr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
  TORD
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC
D MOVE TESTNUM TO ISONUM OF DBUG-REC
D PERFORM VARYING J FROM 1 BY 1 UNTIL J = 16
D MOVE OLSFWH OF INPUT-LINE(J) TO DEBUG-OLSPWH(J)
D MOVE OLIID OF INPUT-LINE(J) TO DEBUG-OLIID(J)
D MOVE OLQTY OF INPUT-LINE(J) TO DEBUG-OLQTY(J)
D MOVE 0 TO DEBUG-IPRICE(J)
D END-PERFORM.
D GET-ISOLATION-DATA-2.
D MOVE OID OF ORDERS-REC TO OID OF DBUG-REC.
D MOVE CCID OF NEWORD-I TO DID OF DBUG-REC
D MOVE CID OF NEWORD-I TO CID OF DBUG-REC
D MOVE CWID OF NEWORD-I TO WID OF DBUG-REC
D CALL "gettime" USING TIME-OF-DAY
D MOVE "b" TO TORD
D CALL "getdtmstr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
  TORD
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC
D MOVE TESTNUM TO ISONUM OF DBUG-REC
D PERFORM VARYING J FROM 1 BY 1 UNTIL J = 16
D MOVE OLSFWH OF INPUT-LINE(J) TO DEBUG-OLSPWH(J)
D MOVE OLIID OF INPUT-LINE(J) TO DEBUG-OLIID(J)
D MOVE OLQTY OF INPUT-LINE(J) TO DEBUG-OLQTY(J)
D IF OLIID OF INPUT-LINE(J) <= 100000
  MOVE OUTPUT-IPRICE(J) TO DEBUG-IPRICE(J)
  ELSE
  MOVE 0 TO DEBUG-IPRICE(J)
  END-IF
D END-IF
D END-PERFORM.
OPEN-ROUTINE.
*OPEN FILES
PERFORM CLOSE-ROUTINE.
OPEN I-O WAREHOUSE-FILE.
OPEN EXTEND NEWORDER-FILE.
OPEN EXTEND ORDERS-FILE.
OPEN EXTEND ORDERLINE-FILE.
OPEN INPUT ITEM-FILE.
OPEN I-O DISTRICT-FILE.
OPEN INPUT ISO-FILE.
OPEN OUTPUT DBUG-FILE.
OPEN OUTPUT DBUG-FILE2.
CLOSE-ROUTINE.
*CLOSE FILES
CLOSE DISTRICT-FILE.

```



```

CLOSE WAREHOUSE-FILE.
CLOSE ITEM-FILE.
CLOSE ORDERLINE-FILE.
CLOSE ORDERS-FILE.
CLOSE NEWORDER-FILE.
CLOSE ISO-FILE.
D CLOSE DEBUG-FILE.
D CLOSE DEBUG-FILE2.

SET-UP-ROUTINE.

CALL "gettime" USING TIME-OF-DAY.
PERFORM OPEN-ROUTINE.

MOVE "STOCK" TO STOCK-HASH-NAME.
MOVE "STIID" TO KNAM OF STOCK-KEY-DATA(1).
MOVE "STWID" TO KNAM OF STOCK-KEY-DATA(2).

SET STOCK-HASH-PTR TO ADDRESS OF STOCK-HASH-NAME.
SET RET-CODE-PTR TO ADDRESS OF RET-CODE.
SET STOCK-DEF-PTR TO ADDRESS OF STOCK-DEF.
SET STOCK-KEY-PTR TO ADDRESS OF STOCK-KEY.

MOVE "CSTMR" TO CSTMR-HASH-NAME.
MOVE "CID" TO KNAM OF CSTMR-KEY-DATA(1).
MOVE "CIDI" TO KNAM OF CSTMR-KEY-DATA(2).
MOVE "CUID" TO KNAM OF CSTMR-KEY-DATA(3).

SET CSTMR-HASH-PTR TO ADDRESS OF CSTMR-HASH-NAME.
SET RET-CODE-PTR TO ADDRESS OF RET-CODE.
SET CSTMR-DEF-PTR TO ADDRESS OF CUSTOMER-REC.
SET CSTMR-KEY-PTR TO ADDRESS OF CSTMR-KEY.

MOVE SPACES TO OCCARD OF ORDERS-REC.
MOVE ZEROS TO OLDLIVOT OF ORDERLINE-REC.

RETURN-DATA.
MOVE NEWORD-O TO OUT-TRAN.

GOBACK.

PROGRAM-END.
STOP RUN.

```

## D.21 NEWSRVR.C:

```

#include <Common.h>
#include <qsoasync.h>
#include <stdio.h>

main(int argc, char **argv)
{
    int sd, snd_len=670, rcv_len=230, port, rc;
    int sd_arr[MAX_CONN] /* the indexes for accepts and use */
    int num_conn /* the number of connections */
    int accept_index /* accept index goes from 0 to num_conn - 1 */
    int retry_count /* number of retries attempted */
    char rcv_buf[MAX_CONN] [BufferSize] /* one per connection */
    char snd_buf [BufferSize];
    char start_error /* start transaction error flag */
    char cmd [120] /* command holder */
    struct sockaddr_in sin;
    int length;
    int Qso_OverlappedIO_t status;
    char Qso_OverlappedIO_t buffer [256];

    memset (&status, 0, sizeof (status));
    status.bufferLength=rcv_len;
    status.postFlag=1; /* Always post to I/O completion port */
    status.fillBuffer=1; /* Fill buffer. */
    /* get the number of connections */
    num_conn = atoi (argv [1]);
    sd = atoi (argv [2]); /* get the socket number */
    /* num_conn = 2; /* test action */

    if ((port=QsoCreateIOCompletionPort ()) < 0) {
        perror ("QsoCreateIOCompletionPort () failed");
        exit (-1);
    }

    length=sizeof (sin);
    for (accept_index=0; accept_index<num_conn; accept_index++)
    {
        if ((sd_arr [accept_index]=accept (sd,
            (struct sockaddr *) &sin, &length)) < 0)
        {
            perror ("accept () in NEWSRVR failed\n");
            printf ("input socket is %d\n", sd);
            close (sd);
            exit (-1);
        }
        status.descriptorHandle=(void *) sd_arr [accept_index];
        status.buffer=rcv_buf [accept_index];
        if (QsoStartRecv (sd_arr [accept_index], port, &status) < 0) {
            perror ("QsoStartRecv () failed");
            exit (-1);
        }
    }

    /* end of connect loop */
    system ("STRMCTL LCKLVL (*ALL)");
    system ("CHGJOB RUNPTY (16)");
    do
    {
        if (QsoWaitForIOCompletion (port, &status, 0) < 0)
        {
            perror ("QsoWaitForIOCompletion () failed");
            exit (-1);
        }

        if (status.returnValue < 0) {
            printf ("\nRecv () failed, errno=%d", status.errnoValue);
            exit (-1);
        }

        if (status.operationCompleted!=QSOSTARTRECV) {
            printf ("\nUnexpected completion=%d", status.operationCompleted);
            exit (-1);
        }

        NEWORDMOD (&first_time, status.buffer, snd_buf);
        if (snd_buf [0] == 'G') /* check the NEWORD-RESULT */
        {
            first_time--;
        }
        if (send ((int) status.descriptorHandle, snd_buf, snd_len, 0) < 0)
        {
            if ((errno != EPIPE) && (errno != EUNKNOWN))
            {
                perror ("nsend (2) failed");
                exit (-1);
            }
        }

        if (QsoStartRecv ((int) status.descriptorHandle, port, &status) < 0)
        {
            perror ("QsoStartRecv () failed");
            exit (-1);
        }
    }
}

```

```

} while (first_time > 0);
for (;;) {
    if (QsoWaitForIOCompletion (port, &status, 0) < 0)
    {
        perror ("QsoWaitForIOCompletion () failed");
        exit (-1);
    }

    if (status.returnValue < 0)
    {
        printf ("\nRecv () failed, errno=%d", status.errnoValue);
        exit (-1);
    }

    if (status.operationCompleted!=QSOSTARTRECV) {
        printf ("\nUnexpected completion=%d", status.operationCompleted);
        exit (-1);
    }

    NEWORDMOD (&first_time, status.buffer, snd_buf);
    if (send ((int) status.descriptorHandle, snd_buf, snd_len, 0) < 0)
    {
        if ((errno != EPIPE) && (errno != EUNKNOWN))
        {
            perror ("nsend (2) failed");
            exit (-1);
        }
    }

    if (QsoStartRecv ((int) status.descriptorHandle, port, &status) < 0)
    {
        perror ("QsoStartRecv () failed");
        exit (-1);
    }
}

} /* for loop */
return 0;
}

```

## D.22 ORDSRVR.C:

```

#include <Common.h>
#include <qsoasync.h>
#include <stdio.h>

void main(int argc, char **argv)
{
    int sd, snd_len=670, rcv_len=230, port, rc;
    int num_conn /* the number of connections */
    int accept_index /* accept index goes from 0 to num_conn - 1 */
    int retry_count /* number of retries attempted */
    char rcv_buf[MAX_CONN] [BufferSize] /* one per connection */
    char snd_buf [BufferSize];
    struct sockaddr_in sin;
    int length;
    int Qso_OverlappedIO_t status;
    char Qso_OverlappedIO_t buffer [256];

    memset (&status, 0, sizeof (status));
    status.bufferLength=rcv_len;
    status.postFlag=1; /* Always post to I/O completion port */
    status.fillBuffer=1; /* Fill buffer. */
    /* get the number of connections */
    num_conn = atoi (argv [1]);
    /* num_conn = 2; /* test action */

    if ((port=QsoCreateIOCompletionPort ()) < 0) {
        perror ("QsoCreateIOCompletionPort () failed");
        exit (-1);
    }

    length=sizeof (sin);
    for (accept_index=0; accept_index<num_conn; accept_index++)
    {
        if ((sd_arr [accept_index]=accept (sd,
            (struct sockaddr *) &sin, &length)) < 0)
        {
            perror ("accept () in DLVSRVR failed");
            close (sd);
            exit (-1);
        }
        status.descriptorHandle=(void *) sd_arr [accept_index];
        status.buffer=rcv_buf [accept_index];
        if (QsoStartRecv (sd_arr [accept_index], port, &status) < 0) {
            perror ("QsoStartRecv () failed");
            exit (-1);
        }
    }

    /* end of connect loop */
    system ("STRMCTL LCKLVL (*ALL)");
    system ("CHGJOB RUNPTY (16)");
    system ("OVDRBF FILE (ORDLNLFL) TOFILE (*FILE) NBRRCD (15)");
    do
    {
        if (QsoWaitForIOCompletion (port, &status, 0) < 0)
        {
            perror ("QsoWaitForIOCompletion () failed");
            exit (-1);
        }

        if (status.returnValue < 0)
        {
            printf ("\nRecv () failed, errno=%d", status.errnoValue);
            exit (-1);
        }

        if (status.operationCompleted!=QSOSTARTRECV)
        {
            printf ("\nUnexpected completion=%d", status.operationCompleted);
            exit (-1);
        }

        ORDSTSMOD (&first_time, status.buffer, snd_buf);
        if ((snd_buf [0] != '6') || /* check the ORDSTS-FMT-NUM */
            ((snd_buf [0] == '6') && (snd_buf [1] != '2')))
        {
            first_time--;
        }
        if (send ((int) status.descriptorHandle, snd_buf, snd_len, 0) < 0)
        {
            if ((errno != EPIPE) && (errno != EUNKNOWN))
            {
                perror ("nsend (1) failed");
                exit (-1);
            }
        }

        if (QsoStartRecv ((int) status.descriptorHandle, port, &status) < 0)
        {
            perror ("QsoStartRecv () failed");
            exit (-1);
        }
    }
}

```

```

    }
    exit(-1);
}

} while ( first_time > 0 );
for(;;) {

if(QsoWaitForIOCompletion(port, &status, 0) < 0 {
    perror("\nQsoWaitForIoCompletion() failed");
    exit(-1);
}

if(status.returnValue <= 0) {
    printf("\nRecv() failed, errno=%d", status.errnoValue);
    exit(-1);
}

if(status.operationCompleted != QSOSTARTRECV) {
    printf("\nUnexpected completion=%d", status.operationCompleted);
    exit(-1);
}

ORDSTSMOD(&first_time, status.buffer, &nd_buf);
if(send((int) status.descriptorHandle, &nd_buf, &nd_len, 0) < 0)
{
    if ((errno != EPIPE) && (errno != EUNKNOWN))
    {
        perror("\nsend(2) failed");
        exit(-1);
    }
}

if(QsoStartRecv((int) status.descriptorHandle, port, &status) < 0)
{
    perror("\nQsoStartRecv() failed");
    exit(-1);
}
} /* for loop */
}

```

## D.23 ORDSTS.CBL:

```

PROCESS NOTRUNC NORANGE.
IDENTIFICATION DIVISION.
PROGRAM-ID. ORDSTSMOD.
AUTHOR. TPCC.
INSTALLATION. IBM-ROCHESTER.
DATE-WRITTEN.
DATE-COMPILED.
*****
*
* PART NAME: ORDERSTATUS
*
*****
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-AS400.
* SOURCE-COMPUTER. IBM-AS400 WITH DEBUGGING MODE.
OBJECT-COMPUTER. IBM-AS400.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ORDERS-VIEW ASSIGN TO DATABASE-ORDERS
        ORGANIZATION IS INDEXED
        ACCESS MODE IS DYNAMIC
        RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
        WITH DUPLICATES.
    SELECT ORDERLINE-VIEW ASSIGN TO DATABASE-ORDLINEF
        ORGANIZATION IS INDEXED
        ACCESS MODE IS DYNAMIC
        RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
        WITH DUPLICATES.
    SELECT DEBUG-FILE
        ASSIGN TO FORMATFILE-DEBUGPRT
        ORGANIZATION IS SEQUENTIAL
        ACCESS IS SEQUENTIAL.
    SELECT ISO-IFILE
        ASSIGN TO DATABASE-ISOIFILE
        ORGANIZATION IS SEQUENTIAL
        ACCESS IS SEQUENTIAL.

I-O-CONTROL.
COMMITMENT CONTROL FOR ORDERS-VIEW
COMMITMENT CONTROL FOR ORDERLINE-VIEW.
DATA DIVISION.
FILE SECTION.

FD ORDERS-VIEW
LABEL RECORDS ARE STANDARD.
01 ORDERS-VIEW-REC.
COPY DDS-ORRCD OF ORDERS.
FD ORDERLINE-VIEW
LABEL RECORDS ARE STANDARD.
01 ORDERLINE-VIEW-REC.
COPY DDS-OLRCD OF ORDLINLF.
05 OL-STATUS-OUTPUT REDEFINES OLACD.
08 FILLER PIC X(10).
08 OL-INFO.
10 OL-INFO-OLSPWH PIC S9(4) COMP-4.
10 OL-INFO-OLIID PIC S9(9) COMP-4.
10 OL-INFO-OLQTY PIC S9(3) COMP-3.
10 OL-INFO-OLAMNT PIC S9(5)V9(2) COMP-3.
10 OL-INFO-OLDLVTD PIC X(8).

DEFD DEBUG-FILE
LABEL RECORDS ARE STANDARD
DATA RECORD IS DEBUG-REC.
D 01 DEBUGPRT-REC.
D 05 DEBUG-REC REDEFINES DEBUGRCD-O.
D 06 DEBUG-DATA.
D 07 ISONUM PIC XX.
D 07 TXIDATA PIC X(40).
D 07 WID PIC S9(4).
D 07 DID PIC S9(2).
D 07 TIME-DATE.
D 08 DATES PIC S9(8).
D 08 TIMES PIC S9(6).
D 07 CID PIC S9(4).
D 07 OID PIC S9(8).
D 07 DEBUG-ITEM-INFO OCCURS 15 TIMES.
D 08 DEBUG-OLSPWH PIC S9(4).
D 08 DEBUG-OLIID PIC S9(6).
D 08 DEBUG-OLQTY PIC S9(3).
D 08 DEBUG-IPRICE PIC S9(5)V9(2).

DEFD ISO-IFILE
LABEL RECORDS ARE STANDARD.
D01 ISOIFILE-RECORD.
D COPY DDS-ISOIRCD OF ISOIFILE.
D 05 ISO-REC REDEFINES ISOIRCD.
D 06 ISOINUM PIC S9(4) COMP-4.

WORKING-STORAGE SECTION.
01 TRAN-OUTPUT.

```

```

05 TRAN-OUT PIC X(670).

05 ORDSTS-O REDEFINES TRAN-OUT.
06 ORDSTS-FMT-NUM PIC XX.
06 OCID PIC S9(9) COMP-4.
06 CUSTINFO.
08 CFIRST PIC X(16).
08 CINIT PIC X(2).
08 CLAST PIC X(16).
06 CBAL PIC S9(11)V9(2) COMP-3.
06 ORDERINFO.
08 OID PIC S9(9) COMP-4.
08 ODT.
10 GENTTOD PIC X(8).
10 GENTTM PIC 9(6).
08 OCARID PIC X(2).
08 OLINENBR PIC S9(4) COMP-4.
06 OLLINEINFO OCCURS 15 TIMES.
07 DO-OLSPWH PIC S9(4) COMP-4.
07 DO-OLIID PIC S9(9) COMP-4.
07 DO-OLQTY PIC S9(3) COMP-3.
07 DO-OLAMNT PIC S9(5)V9(2) COMP-3.
07 DO-OLDLVTD PIC X(8).

01 CSTM-KEY-PTR USAGE POINTER.

01 CSTM-KEY.
03 CSTM-KEY-DATA OCCURS 5 TIMES.
05 ENAM PIC X(10).
05 KVAL PIC S9(9) COMP-4.

01 CSTM-HASH-NAME PIC X(10).

01 CSTM-HASH-PTR USAGE POINTER.

01 FUNCN PIC X(1).
01 KEYS PIC S9(9) COMP-4.
01 RET-CODE PIC S9(9) COMP-4.

01 RET-CODE-PTR USAGE POINTER.

01 CSTM-DEF-PTR USAGE POINTER.

01 FETCHR PIC X(1) VALUE "1".

01 CUST-KEYS PIC 9(1) comp-4 VALUE 3.

01 CUSTOMER-REC.
05 CUST-KEY.
06 CID PIC S9(9) COMP-4.
06 CDID PIC S9(4) COMP-4.
06 CWID PIC S9(4) COMP-4.
06 CUST-INFO.
08 CFIRST PIC X(16).
08 CINIT PIC X(02).
08 CLAST PIC X(16).
08 CLDATE PIC S9(8).
08 CADDR1 PIC X(20).
08 CCKRED PIC X(02).
08 CADDR2 PIC X(20).
08 CDCT PIC S9V9999 COMP-3.
08 CCITY PIC X(20).
08 CSTATE PIC X(02).
08 CZIP PIC X(9).
08 CPHONE PIC X(16).
08 CBAL PIC S9(11)V99 COMP-3.
08 CCKRDM PIC S9(11)V99 COMP-3.
08 CYTD PIC S9(11)V99 COMP-3.
08 CPAYCNT PIC S9(3) COMP-3.
08 CDELCTN PIC S9(3) COMP-3.
08 CDATA PIC X(500).

01 TRANSACTION-INPUT.
06 TXN-TYPE PIC X.
06 JOBNAME PIC X(10).
06 CLIENT-INPUT PIC X(202).
06 ORDSTS-I REDEFINES CLIENT-INPUT.
08 ORDSTS-TYPE PIC X.
08 ORD-CUST-KEY.
09 CID PIC S9(9) COMP-4.
09 DID PIC S9(4) COMP-4.
09 WID PIC S9(4) COMP-4.
06 CLAST PIC X(16).
08 FILLER PIC X(165).

01 CWIDR PIC S9(4) COMP-4.
01 CDIDR PIC S9(4) COMP-4.
01 CLASTR PIC X(16).
01 CLASTK PIC X(16).
01 CUSTREL PIC 9(9) COMP-4.

D 01 TESTNUM PIC 99 VALUE 1.
D 01 J PIC S999 COMP-3.

01 TIME-OF-DAY PIC X(8).

D 01 TIME-OF-DAY-STRING PIC X(14).

D 01 TORD PIC X(1) VALUE "b".

01 CNTR PIC 99.
01 CUST-INDEX PIC S9(4) USAGE BINARY.
01 CUSTR-INDEX PIC S9(4) USAGE BINARY.
01 CUSTOMER-ARRAY.
03 CUSTOMER-ARRAY-ELEMENT OCCURS 100 TIMES.
05 CID PIC S9(9) COMP-4.
05 CDID PIC S9(4) COMP-4.
05 CWID PIC S9(4) COMP-4.
01 TEMP-DATA2 PIC X(468).

01 HOST-DATA.
03 HOST-ARRAY-ELEMENT OCCURS 100 TIMES.
05 CID PIC S9(9) COMP-4.

01 ERROR-HDLNG-PARAMETERS.
05 NEWORDER-FILE-STATUS PIC X(2).

01 I PIC S999 BINARY.

01 TOO-MANY.
05 TOO-MANY-1.
06 TOO-MANY-CID PIC S9(9) COMP-4.
06 TOO-MANY-DID PIC S9(4) COMP-4.
06 TOO-MANY-WID PIC S9(4) COMP-4.
06 TOO-MANY-CLAST PIC X(16).

EXEC SQL
INCLUDE SQLCA
END-EXEC.

* SQL ERROR/WARNING Messages *

EXEC SQL
WHENEVER SQLERROR CONTINUE
END-EXEC.

```

```

EXEC SQL
WHENEVER SQLWARNING CONTINUE
END-EXEC.

01 XX PIC S9(2) COMP-4.

LINKAGE SECTION.
01 FIRST-TIME PIC S9(4) COMP-4.
01 INP-TRAN PIC X(230).
01 OUT-TRAN PIC X(670).

PROCEDURE DIVISION USING FIRST-TIME INP-TRAN OUT-TRAN.

DECLARATIVES.

HANDLE-ERROR SECTION.

I-O-ERROR-PARA.

END DECLARATIVES.

MAIN-LINE-ROUTINE.

IF FIRST-TIME > 0
PERFORM SET-UP-ROUTINE.

MOVE INP-TRAN TO TRANSACTION-INPUT.

D READ ISO-IFILE.
D MOVE ISOINUM TO TESTNUM, ISOINUM OF DBUG-REC.

ORDER-STATUS-TRANSACTION.

* THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
* FLAG IS SET AND IS FOR ISOLATION TEST USE
*
D MOVE "ORDER STATUS PRE-READ " TO TXTDATA OF DBUG-REC.
D MOVE 0 TO OIID OF DBUG-REC
D PERFORM GET-ISOLATION-DATA-3.
D WRITE DBUGPRT-REC FORMAT IS "DEBUGRCD".

MOVE WID OF ORDSTS-I TO OWID OF ORDERS-VIEW-REC.
MOVE DID OF ORDSTS-I TO ODID OF ORDERS-VIEW-REC.
MOVE ORD-CUST-KEY OF ORDSTS-I TO CUST-KEY OF CUSTOMER-REC.
ORDER-STATUS-READS.
IF ORDSTS-TYPE = "C" THEN
PERFORM CUSTOMER-BY-NUMBER
ELSE
MOVE CLAST OF ORDSTS-I TO CLASTK
MOVE WID OF ORDSTS-I TO CWID OF CUSTOMER-REC
MOVE DID OF ORDSTS-I TO CDID OF CUSTOMER-REC
PERFORM CUSTOMER-BY-NAME THROUGH
CUSTOMER-BY-NAME-EXIT
END-IF.
MOVE CID OF CUSTOMER-REC TO OCID OF ORDERS-VIEW-REC
OCID OF ORDSTS-O.
MOVE 999999 TO OIID OF ORDERS-VIEW-REC.

START ORDERS-VIEW KEY NOT < EXTERNALLY-DESCRIBED-KEY
INVALID KEY
MOVE "62" TO ORDSTS-FMT-NUM
GO TO ORDER-STATUS-TRANSACTION-EXIT.
READ ORDERS-VIEW PRIOR
IF TESTNUM = 9 THEN
CALL "delayme"
READ ORDERS-VIEW
END-IF

MOVE OWID OF ORDERS-VIEW-REC TO
OLIID OF ORDERLINE-VIEW-REC.
MOVE ODID OF ORDERS-VIEW-REC TO
OLDID OF ORDERLINE-VIEW-REC.
MOVE OIID OF ORDERS-VIEW-REC TO
OLOIID OF ORDERLINE-VIEW-REC OIID OF ORDSTS-O.
MOVE 1 TO OLINBR OF ORDERLINE-VIEW-REC.
MOVE OLINES OF ORDERS-VIEW-REC TO
OLINENBR OF ORDSTS-O.
MOVE OCARID OF ORDERS-VIEW-REC TO
OCARID OF ORDSTS-O.
MOVE OENTTOD OF ORDERS-VIEW-REC TO
OENTTOD OF ORDSTS-O.

READ ORDERLINE-VIEW
MOVE OL-INFO TO OLININFO(1)

PERFORM VARYING I
FROM 2 BY 1 UNTIL I > OLINES OF ORDERS-VIEW-REC

READ ORDERLINE-VIEW NEXT
MOVE OL-INFO TO OLININFO(1)

END-PERFORM.

MOVE I TO ORDSTS-FMT-NUM.
D MOVE OIID OF ORCRD OF ORDERS-VIEW-REC TO OIID OF DBUG-REC.
COMMIT.

* THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
* FLAG IS SET AND IS FOR ISOLATION TEST USE
*
D MOVE "ORDER STATUS POST-COMMIT " TO TXTDATA OF DBUG-REC.
* MOVE 0 TO OIID OF DBUG-REC
D PERFORM GET-ISOLATION-DATA-4.
D WRITE DBUGPRT-REC FORMAT IS "DEBUGRCD".
D CLOSE DBUG-FILE.
D OPEN OUTPUT DBUG-FILE.

MOVE CBAL OF CUSTOMER-REC TO CBAL OF ORDSTS-O.
MOVE CFIRST OF CUSTOMER-REC TO CFIRST OF ORDSTS-O.
MOVE CINIT OF CUSTOMER-REC TO CINIT OF ORDSTS-O.
MOVE CLAST OF CUSTOMER-REC TO CLAST OF ORDSTS-O.

MOVE ORDSTS-O TO OUT-TRAN.
GOBACK.
ORDER-STATUS-TRANSACTION-EXIT.
MOVE ORDSTS-O TO OUT-TRAN.
GOBACK.

CUSTOMER-BY-NUMBER.
MOVE CID OF CUSTOMER-REC TO KVAL OF CSTMR-KEY-DATA(1)
MOVE CDID OF CUSTOMER-REC TO KVAL OF CSTMR-KEY-DATA(2)
MOVE CWID OF CUSTOMER-REC TO KVAL OF CSTMR-KEY-DATA(3)

CALL "qdbrunha" USING BY VALUE
CSTMR-HASH-PTR FETCHR CUST-KEYS
CSTMR-KEY-PTR CSTMR-DEF-PTR RET-CODE-PTR

IF RET-CODE > 0
IF RET-CODE = 100
MOVE "62" TO ORDSTS-FMT-NUM
ROLLBACK
GO TO ORDER-STATUS-TRANSACTION-EXIT
ELSE
IF RET-CODE = 812
ROLLBACK
GO TO ORDER-STATUS-TRANSACTION
ELSE
ROLLBACK
GO TO PROGRAM-END
END-IF

```

```

END-IF
END-IF.

CUSTOMER-BY-NAME.

MOVE CWID OF CUSTOMER-REC TO CWIDR.
MOVE CDID OF CUSTOMER-REC TO CDIDR.
MOVE CLASTK TO CLASTR.

EXEC SQL DECLARE C1 CURSOR FOR
SELECT CID
FROM CSTMRPFCT
WHERE CLAST = :CLASTR AND CDID = :CDIDR AND
CWID = :CWIDR
END-EXEC.

EXEC SQL OPEN C1
END-EXEC.

EXEC SQL FETCH C1 FOR 100 ROWS INTO :HOST-ARRAY-ELEMENT
END-EXEC.
MOVE SQLERRD OF SQLCA(3) TO XX.

EXEC SQL CLOSE C1
END-EXEC.

IF XX = 0
MOVE "62" TO ORDSTS-FMT-NUM
ROLLBACK
GO TO ORDER-STATUS-TRANSACTION-EXIT
END-IF

IF XX = 100
CALL "TOOMANY" USING CWIDR, CDIDR, CLASTR, CUSTREL
ELSE
COMPUTE CUST-INDEX = (XX + 1) / 2.
MOVE HOST-ARRAY-ELEMENT(CUST-INDEX) TO
CID OF CUSTOMER-REC.
PERFORM CUSTOMER-BY-NUMBER.

CUSTOMER-BY-NAME-EXIT.

D GET-ISOLATION-DATA-3.
D MOVE DID OF ORDSTS-I TO DID OF DBUG-REC
D MOVE CID OF ORDSTS-I TO CID OF DBUG-REC
D MOVE WID OF ORDSTS-I TO WID OF DBUG-REC
D CALL "gettime" USING TIME-OF-DAY
D MOVE "b" TO TORD
D CALL "getdtmstr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
D TORD
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC
D MOVE TESTNUM TO ISOINUM OF DBUG-REC
D PERFORM VARYING J FROM 1 BY 1 UNTIL J = 16
D MOVE 0 TO DEBUG-OLSPWH(J)
D MOVE 0 TO DEBUG-OLIID(J)
D MOVE 0 TO DEBUG-OLQTY(J)
D MOVE 0 TO DEBUG-IFPRICE(J)
D END-PERFORM.

D GET-ISOLATION-DATA-4.
D MOVE DID OF ORDSTS-I TO DID OF DBUG-REC
D MOVE CID OF ORDSTS-I TO CID OF DBUG-REC
D MOVE WID OF ORDSTS-I TO WID OF DBUG-REC
D CALL "gettime" USING TIME-OF-DAY
D MOVE "b" TO TORD
D CALL "getdtmstr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
D TORD
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC
D MOVE TESTNUM TO ISOINUM OF DBUG-REC
D PERFORM VARYING J FROM 1 BY 1 UNTIL J >
D OLINES OF ORDERS-VIEW-REC
D MOVE DO-OLSPWH(J) TO DEBUG-OLSPWH(J)
D MOVE DO-OLIID(J) TO DEBUG-OLIID(J)
D MOVE DO-OLQTY(J) TO DEBUG-OLQTY(J)
D MOVE 0 TO DEBUG-IFPRICE(J)
D END-PERFORM.

SET-UP-ROUTINE.

CALL "gettime" USING TIME-OF-DAY.

*OPEN FILES
PERFORM CLOSE-ROUTINE.
OPEN INPUT ORDERLINE-VIEW
ORDERLINE-VIEW
ORDERS-VIEW.
D OPEN INPUT ISO-IFILE.
D OPEN OUTPUT DBUG-FILE.

MOVE "CSTMR" TO CSTMR-HASH-NAME.
MOVE "CID" TO KNAM OF CSTMR-KEY-DATA(1).
MOVE "CDID" TO KNAM OF CSTMR-KEY-DATA(2).
MOVE "CWID" TO KNAM OF CSTMR-KEY-DATA(3).

SET CSTMR-HASH-PTR TO ADDRESS OF CSTMR-HASH-NAME.
SET RET-CODE-PTR TO ADDRESS OF RET-CODE.
SET CSTMR-DEF-PTR TO ADDRESS OF CUSTOMER-REC.
SET CSTMR-KEY-PTR TO ADDRESS OF CSTMR-KEY.

CLOSE-ROUTINE.
CLOSE ORDERLINE-VIEW.
CLOSE ORDERS-VIEW.
CLOSE ISO-IFILE.
CLOSE DBUG-FILE.

PROGRAM-END.
STOP RUN.

```

## D.24 PAYMENT.CBL:

```

PROCESS NOTRUNC NORANGE.
IDENTIFICATION DIVISION.
PROGRAM-ID, PAYMENTWGD.
AUTHOR, TPCC.
INSTALLATION, IBM-ROCHESTER.
DATE-WRITTEN.
DATE-COMPILED.
*****
* PART NAME: PAYMENT
*****
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER, IBM-AS400.
* SOURCE-COMPUTER, IBM-AS400 WITH DEBUGGING MODE.
OBJECT-COMPUTER, IBM-AS400.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT WAREHOUSE-FILE ASSIGN TO DATABASE-WRHS
ORGANIZATION IS INDEXED
ACCESS MODE IS RANDOM
RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
WITH DUPLICATES
FILE STATUS IS WAREHOUSE-FILE-STATUS.

```

```

SELECT DISTRICT-FILE ASSIGN TO DATABASE-DSTRCT
ORGANIZATION IS INDEXED
ACCESS MODE IS RANDOM
RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
WITH DUPLICATES
FILE STATUS IS DISTRICT-FILE-STATUS.
SELECT HISTORY-FILE ASSIGN TO DATABASE-HSTRY
ORGANIZATION IS SEQUENTIAL.
D SELECT DEBUG-FILE2
D ASSIGN TO FORMATFILE-DEBUGPRT2
D ORGANIZATION IS SEQUENTIAL
D ACCESS IS SEQUENTIAL.
D SELECT ISO-IFILE
D ASSIGN TO DATABASE-ISOIFILE
D ORGANIZATION IS SEQUENTIAL
D ACCESS IS SEQUENTIAL.
D
I-O-CONTROL.
COMMITMENT CONTROL FOR WAREHOUSE-FILE
COMMITMENT CONTROL FOR DISTRICT-FILE.
COMMITMENT CONTROL FOR HISTORY-FILE.
DATA DIVISION.
FILE SECTION.
FD WAREHOUSE-FILE
LABEL RECORDS ARE STANDARD.
01 WAREHOUSE-REC.
COPY DDS-WRRCDC OF WRHS.
05 WRHS-INFO REDEFINES WRRCDC.
06 FILLER PIC X(12).
06 WH-ADDRESS.
08 WADDR1 PIC X(20).
08 WADDR2 PIC X(20).
08 CITYW PIC X(20).
08 STATEW PIC X(2).
08 ZIPW PIC X(10).
FD DISTRICT-FILE
LABEL RECORDS ARE STANDARD.
01 DISTRICT-REC.
COPY DDS-DSRDC OF DSTRCT.
05 DISTRICT-INFO REDEFINES DSRDC.
06 DIST-KEY.
08 DID PIC S9(4) COMP-4.
08 DWDID PIC S9(4) COMP-4.
06 FILLER PIC X(10).
06 DIST-ADDRESS.
08 DADDR1 PIC X(20).
08 DADDR2 PIC X(20).
08 CITY PIC X(20).
08 STATE PIC X(2).
08 ZIP PIC X(10).
FD HISTORY-FILE
LABEL RECORDS ARE STANDARD
DATA RECORD IS HISTORY-REC.
01 HISTORY-REC.
COPY DDS-HSRDC OF HSTRY.
05 HIST-INFO REDEFINES HSRDC.
07 HISTORY-DATA.
09 PAY-DIST-KEY.
10 DID PIC S9(4) COMP-4.
10 WID PIC S9(4) COMP-4.
09 PAY-CUST-KEY.
10 CID PIC S9(9) COMP-4.
10 CDID PIC S9(4) COMP-4.
10 CWID PIC S9(4) COMP-4.
07 DATE-TIME.
09 HTOD PIC X(8).
07 AMOUNT PIC S9(5)V99 COMP-3.
07 HDATA.
09 WNAME PIC X(10).
09 FILLER PIC XXXX.
09 DNAME PIC X(10).
DFD DEBUG-FILE2
D LABEL RECORDS ARE STANDARD
D DATA RECORD IS DEBUG-REC2.
D 01 DEBUGPRT-REC2.
D COPY DDS-DEBUGRCD2 OF DEBUGPRT2.
D 05 DEBUG-REC2 REDEFINES DEBUGRCD2-0.
D 06 DEBUG-DATA.
D 07 ISONUM PIC XX.
D 07 TXTDATA PIC X(40).
D 07 WID PIC S9(4).
D 07 DID PIC S9(2).
D 07 TIME-DATE.
D 08 DATEG PIC S9(8).
D 08 TIMEG PIC S9(6).
D 07 CID PIC S9(4).
D 07 CWID PIC S9(4).
D 07 CDID PIC S9(2).
D 07 CBAL PIC S9(11)V99.
D 07 PAYMNT PIC S9(7)V99.
DFD ISO-IFILE
D LABEL RECORDS ARE STANDARD.
D01 ISOFILE-RECORD.
D COPY DDS-ISOIRDC OF ISOIFILE.
D 05 ISO-IREC REDEFINES ISOIRDC.
D 06 ISOINUM PIC S9(4) COMP-4.
WORKING-STORAGE SECTION.
01 TRAN-OUTPUT.
05 TRAN-OUT PIC X(670).
05 PAYMNT-0 REDEFINES TRAN-OUT.
06 OUTPUT-FMT-NUM-3 PIC XX.
* 17 = bad credit
* 18 = no cid or clast
* 19 = good credit
* 88 = bad last name
06 HDT.
08 HODOD PIC X(8).
08 HTIME pic 9(6).
06 WH-DATA.
08 WADDR1 PIC X(20).
08 WADDR2 PIC X(20).
08 WCITY PIC X(20).
08 WSTATE PIC X(02).
08 WZIP PIC X(9).
06 DST-DATA.
08 DADDR1 PIC X(20).
08 DADDR2 PIC X(20).
08 DCITY PIC X(20).
08 DSTATE PIC X(02).
08 DZIP PIC X(9).
06 CID PIC S9(9) COMP-4.
06 PAYMENT-CUST-INFO.
08 CFIRST PIC X(16).
08 CINIT PIC X(02).
08 CLAST PIC X(16).
08 CLDATE PIC S9(8).
08 CADDR1 PIC X(20).
08 CREDIT PIC X(02).
08 CADDR2 PIC X(20).
08 CDCT PIC S999V99 COMP-3.
08 CCITY PIC X(20).
08 CSTATE PIC X(02).
08 CSTATE PIC X(02).
08 CZIP PIC X(9).
08 CPHONE PIC X(16).
08 CBAL PIC S9(11)V99(2) COMP-3.
08 CCRDLM PIC S9(11)V99(2) COMP-3.
06 MISC-CDATA.
08 CDAT1 PIC X(50).
08 CDAT2 PIC X(50).
08 CDAT3 PIC X(50).
08 CDAT4 PIC X(50).
01 CSTM-KEY-PTR USAGE POINTER.
01 CSTM-KEY.
03 CSTM-KEY-DATA OCCURS 5 TIMES.
05 KNAM PIC X(10).
05 KVAL PIC S9(9) COMP-4.
01 CSTM-HASH-NAME PIC X(10).
01 CSTM-HASH-PTR USAGE POINTER.
01 FUNCT PIC X(1).
01 KEYS PIC S9(9) COMP-4.
01 RET-CODE PIC S9(9) COMP-4.
01 RET-CODE-PTR USAGE POINTER.
01 CSTM-DEF-PTR USAGE POINTER.
01 UPDATER PIC X(1) VALUE "3".
01 FETCHR PIC X(1) VALUE "1".
01 FETCHUPDATE PIC X(1) VALUE "2".
01 CUST-KEYS PIC 9(1) COMP-4 VALUE 3.
01 CMNF-INDIC-AREA.
03 CMNF-INDIC PIC 1 OCCURS 99 TIMES
INDICATOR 1.
01 CUSTOMER-REC.
05 CUST-KEY.
06 CID PIC S9(9) COMP-4.
06 CDID PIC S9(4) COMP-4.
06 CWID PIC S9(4) COMP-4.
06 CUST-INFO.
08 CFIRST PIC X(16).
08 CINIT PIC X(02).
08 CLAST PIC X(16).
08 CLDATE PIC S9(8).
08 CADDR1 PIC X(20).
08 CREDIT PIC X(02).
08 CADDR2 PIC X(20).
08 CDCT PIC S999V99 COMP-3.
08 CCITY PIC X(20).
08 CSTATE PIC X(02).
08 CZIP PIC X(9).
08 CPHONE PIC X(16).
08 CBAL PIC S9(11)V99 COMP-3.
08 CCRDLM PIC S9(11)V99 COMP-3.
08 CYTD PIC S9(11)V99 COMP-3.
08 CPAYCNT PIC S9(3) COMP-3.
08 CDELCONT PIC S9(3) COMP-3.
08 CDATA PIC X(500).
01 TRANSACTION-INPUT.
06 TXN-TYPE PIC X.
06 JOBNAME PIC X(10).
06 CLIENT-INPUT PIC X(202).
06 NEWORD-I REDEFINES CLIENT-INPUT.
08 CWID PIC S9(4) COMP-4.
08 CDID PIC S9(4) COMP-4.
08 CID PIC S9(9) COMP-4.
08 NUMBER-OF-ITEMS PIC 9(3) COMP-3.
08 INPUT-LINE OCCURS 15 TIMES.
10 OLSPPH PIC S9(4) COMP-4.
10 GLIID PIC S9(9) COMP-4.
10 GLQTY PIC S9(3) COMP-3.
06 PAYMENT-I REDEFINES CLIENT-INPUT.
08 PAYMENT-TYPE PIC X.
* Payment type = C if by CID
* Payment type = L if by CLAST
08 HISTORY-DATA.
09 PAY-DIST-KEY.
10 DID PIC S9(4) COMP-4.
10 WID PIC S9(4) COMP-4.
09 PAY-CUST-KEY.
10 CID PIC S9(9) COMP-4.
10 CDID PIC S9(4) COMP-4.
10 CWID PIC S9(4) COMP-4.
08 AMOUNT PIC S9(5)V99 COMP-3.
08 CLAST PIC X(16).
08 FILLER PIC X(155).
06 ORDSTS-I REDEFINES CLIENT-INPUT.
08 ORDSTS-TYPE PIC X.
08 ORD-CUST-KEY.
09 CID PIC S9(9) COMP-4.
09 DID PIC S9(4) COMP-4.
09 WID PIC S9(4) COMP-4.
08 CLAST PIC X(16).
08 FILLER PIC X(165).
06 STKVL-I REDEFINES CLIENT-INPUT.
08 WID PIC S9(4) COMP-4.
08 DID PIC S9(4) COMP-4.
08 THRESHOLD PIC S9(4) COMP-4.
08 BLASTK PIC S9(4) COMP-4.
06 DLVRY-I REDEFINES CLIENT-INPUT.
08 WID PIC S9(4) COMP-4.
08 CARRIER PIC XX.
08 D-DATE PIC 9(8).
08 D-TIME PIC 9(8).
01 CWIDR PIC S9(4) COMP-4.
01 CDIDR PIC S9(4) COMP-4.
01 CLASTR PIC X(16).
01 CLASTK PIC X(16).
01 CUSTREL PIC 9(9) COMP-4.
01 CHAR-HIST-DATA.
05 BID PIC S99.
05 WID PIC S9(4).
05 CID PIC S9(6).
05 CDID PIC S99.
05 CWID PIC S9(4).
05 AMOUNT PIC S9(5)V99.
01 ROLL-BACK-REQUIRED PIC X(1).
01 CONTR-1 PIC S99 COMP-4.
01 DATEINT.
06 YY PIC 9(2).
06 MMDD PIC 9(4).
01 DATETIME-INIT.
05 TPCDATE-INIT PIC X(8) VALUE "00000000".
01 DATETIME-CHARINIT REDEFINES DATETIME-INIT PIC X(8).
01 TIME-OF-DAY PIC X(8).
01 CNTR PIC 99.
01 CUST-INDEX PIC S9(4) USAGE BINARY.
01 CUSTR-INDEX PIC S9(4) USAGE BINARY.
01 CUSTOMER-ARRAY.
03 CUSTOMER-ARRAY-ELEMENT OCCURS 100 TIMES.
05 CID PIC S9(9) COMP-4.

```

```

05 CDID PIC S9(4) COMP-4.
05 CWID PIC S9(4) COMP-4.
01 TEMP-DATA2 PIC X(468).

01 HOST-DATA.
03 HOST-ARRAY-ELEMENT OCCURS 100 TIMES.
05 CID PIC S9(9) COMP-4.

01 ERROR-HDLNG-PARAMETERS.
05 DISTRICT-FILE-STATUS PIC X(2).
05 WAREHOUSE-FILE-STATUS PIC X(2).

01 I PIC S999 BINARY.
D 01 TESTNUM PIC 99 VALUE 1.
D 01 J PIC S999 COMP-3.
D 01 TIME-OF-DAY-STRING PIC X(14).
D 01 TORD PIC X(1) VALUE "b".

EXEC SQL
INCLUDE SQLCA
END-EXEC.

* SQL ERROR/WARNING Messages *

EXEC SQL
WHENEVER SQLERROR CONTINUE
END-EXEC.

EXEC SQL
WHENEVER SQLWARNING CONTINUE
END-EXEC.

01 XX PIC S9(2) COMP-4.

LINKAGE SECTION.
01 FIRST-TIME PIC S9(4) COMP-4.
01 INP-TRAN PIC X(230).
01 OUT-TRAN PIC X(670).

PROCEDURE DIVISION USING FIRST-TIME INP-TRAN OUT-TRAN.

DECLARATIVES.

HANDLE-ERROR SECTION.

I-O-ERROR-PARA.

END DECLARATIVES.

MAIN-LINE-ROUTINE.

IF FIRST-TIME > 0
PERFORM SET-UP-ROUTINE.

MOVE INP-TRAN TO TRANSACTION-INPUT.

D READ ISO-IFILE.
D MOVE ISOINUM TO TESTNUM, ISONUM OF DBUG-REC2.

PAYMENT-TRANSACTION.

MOVE HISTORY-DATA OF PAYMENT-I TO HISTORY-DATA OF HIST-INFO.
MOVE AMOUNT OF PAYMENT-I TO AMOUNT OF HIST-INFO.

CALL "gettime" USING TIME-OF-DAY.

MOVE TIME-OF-DAY TO DATE-TIME OF HIST-INFO, HDT.

* THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
* FLAG IS SET AND IS FOR ISOLATION TEST USE
*
D MOVE "PAY PRE-READ " TO TXTDATA OF DBUG-REC2.
D MOVE CDID OF CUSTOMER-REC TO CDID OF DBUG-REC2.
D MOVE CWID OF CUSTOMER-REC TO CWID OF DBUG-REC2.
D MOVE CID OF PAYMENT-I TO CID OF DBUG-REC2.
D MOVE CID OF CUSTOMER-REC TO DID OF DBUG-REC2.
D MOVE AMOUNT OF PAYMENT-I TO
D PAYMNT OF DBUG-REC2.
D MOVE WID OF PAYMENT-I TO WID OF DBUG-REC2.
D MOVE 0 TO CBAL OF DBUG-REC2.
D MOVE "b" TO TORD.
D CALL "getdtmstr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
D TORD.
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC2.
D WRITE DBUGPRT-REC2 FORMAT IS "DBGRC02".

IF PAYMENT-TYPE EQUAL "C"
MOVE PAY-CUST-KEY OF PAYMENT-I TO CUST-KEY
OF CUSTOMER-REC
PERFORM CUSTOMER-BY-NUMBER
ELSE
MOVE CWID OF PAYMENT-I TO CWID OF CUSTOMER-REC
MOVE CID OF PAYMENT-I TO CID OF CUSTOMER-REC
MOVE CLAST OF PAYMENT-I TO CLASTK

PERFORM CUSTOMER-BY-NAME THRU CUSTOMER-BY-NAME-EXIT.

D MOVE CBAL OF CUSTOMER-REC TO CBAL OF DBUG-REC2.
SUBTRACT AMOUNT OF PAYMENT-I
FROM CBAL OF CUSTOMER-REC.

ADD AMOUNT OF PAYMENT-I
TO CYTD OF CUSTOMER-REC.

ADD 1 TO CPAYCNT OF CUSTOMER-REC.
MOVE CUST-INFO TO PAYMENT-CUST-INFO.
MOVE CBAL OF CUSTOMER-REC TO CBAL OF PAYMNT-O.
MOVE CCRDLM OF CUSTOMER-REC
TO CCRDLM OF PAYMNT-O.
MULTIPLY CDCT OF CUSTOMER-REC BY 100 GIVING
CDCT OF PAYMNT-O.

MOVE SPACES TO MISC-CDATA.
IF CCREDT OF CUSTOMER-REC = "BC" THEN
MOVE CDATA OF CUSTOMER-REC (1:468)
TO TEMP-DATA2
MOVE TEMP-DATA2
TO CDATA OF CUSTOMER-REC (32:468)

MOVE DID OF HISTORY-DATA OF HIST-INFO
TO DID OF CHAR-HIST-DATA
MOVE WID OF HISTORY-DATA OF HIST-INFO
TO WID OF CHAR-HIST-DATA
MOVE CID OF HISTORY-DATA OF HIST-INFO
TO CID OF CHAR-HIST-DATA
MOVE CDID OF HISTORY-DATA OF HIST-INFO
TO CDID OF CHAR-HIST-DATA
MOVE CWID OF HISTORY-DATA OF HIST-INFO
TO CWID OF CHAR-HIST-DATA
MOVE AMOUNT OF HIST-INFO
TO AMOUNT OF CHAR-HIST-DATA
MOVE CHAR-HIST-DATA TO CDATA OF CUSTOMER-REC(1:31)
MOVE CDATA OF CUSTOMER-REC TO MISC-CDATA
END-IF.

CALL "qdbrunha" USING BY VALUE
CSTMW-HASH-PTR UPDATER CUST-KEYS CSTMW-KEY-PTR
CSTMW-DEF-PTR RET-CODE-PTR
IF RET-CODE > 0
IF RET-CODE = 100
MOVE "77" TO OUTPUT-FMT-NUM-3
ROLLBACK
GO TO PAYMENT-TRANSACTION-EXIT
ELSE
IF RET-CODE = 812
ROLLBACK
GO TO PAYMENT-TRANSACTION
ELSE
ROLLBACK
GO TO PROGRAM-END
END-IF
END-IF
MOVE PAY-DIST-KEY OF PAYMENT-I TO DIST-KEY.

READ-DISTRCT.
READ DISTRICT-FILE.
IF DISTRICT-FILE-STATUS NOT EQUAL "00"
IF DISTRICT-FILE-STATUS EQUAL "9d"
GO TO READ-DISTRCT
ELSE
MOVE "77" TO OUTPUT-FMT-NUM-3
ROLLBACK
GO TO PAYMENT-TRANSACTION-EXIT
END-IF
END-IF.

ADD AMOUNT OF PAYMENT-I
TO DFTD OF DISTRICT-REC.

REWRITE DISTRICT-REC.

MOVE WID OF PAYMENT-I TO WID OF WAREHOUSE-REC.

READ WAREHOUSE-FILE.
IF WAREHOUSE-FILE-STATUS NOT EQUAL "00"
IF WAREHOUSE-FILE-STATUS EQUAL "9d"
ROLLBACK
GO TO PAYMENT-TRANSACTION
ELSE
MOVE "77" TO OUTPUT-FMT-NUM-3
ROLLBACK
GO TO PAYMENT-TRANSACTION-EXIT
END-IF
END-IF.

ADD AMOUNT OF PAYMENT-I
TO WYTD OF WAREHOUSE-REC.

REWRITE WAREHOUSE-REC.

MOVE CID OF CUSTOMER-REC TO
CID OF HISTORY-REC.
MOVE WNAME OF WAREHOUSE-REC TO
WNAME OF HDATA OF HIST-INFO.
MOVE DNAME OF DISTRICT-REC TO
DNAME OF HDATA OF HIST-INFO.

WRITE HISTORY-REC.

* THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
* FLAG IS SET AND IS FOR ISOLATION TEST USE
*
D MOVE "PAY PRE-COMMIT" TO TXTDATA OF DBUG-REC2.
D MOVE CDID OF CUSTOMER-REC TO CDID OF DBUG-REC2.
D MOVE CWID OF CUSTOMER-REC TO CWID OF DBUG-REC2.
D MOVE CID OF PAYMENT-I TO CID OF DBUG-REC2.
D MOVE CID OF CUSTOMER-REC TO DID OF DBUG-REC2.
D MOVE WID OF PAYMENT-I TO WID OF DBUG-REC2.
* MOVE AMOUNT OF PAYMENT-I TO
* PAYMNT OF DBUG-REC2.
D CALL "gettime" USING TIME-OF-DAY.
D MOVE "b" TO TORD.
D CALL "getdtmstr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
D TORD.
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC2.
D WRITE DBUGPRT-REC2 FORMAT IS "DBGRC02".
D CALL "delayme".

***** For Atomicity Rollback case *****
D IF TESTNUM EQUAL "20"
D ROLLBACK
D GO TO PAY-POST-COMMIT
D END-IF.

COMMIT.

* THE FOLLOWING CODE WILL ONLY BE INCLUDED WHEN THE
* FLAG IS SET AND IS FOR ISOLATION TEST USE
*
D PAY-POST-COMMIT.
D MOVE "PAY POST-COMMIT" TO TXTDATA OF DBUG-REC2.
D MOVE CDID OF CUSTOMER-REC TO CDID OF DBUG-REC2.
D MOVE CWID OF CUSTOMER-REC TO CWID OF DBUG-REC2.
D MOVE CID OF PAYMENT-I TO CID OF DBUG-REC2.
D MOVE CID OF CUSTOMER-REC TO DID OF DBUG-REC2.
* MOVE AMOUNT OF PAYMENT-I TO
* PAYMNT OF DBUG-REC2.
D MOVE WID OF PAYMENT-I TO WID OF DBUG-REC2.
D MOVE CBAL OF CUSTOMER-REC TO CBAL OF DBUG-REC2.
D CALL "gettime" USING TIME-OF-DAY.
D MOVE "b" TO TORD.
D CALL "getdtmstr" USING TIME-OF-DAY, TIME-OF-DAY-STRING,
D TORD.
D MOVE TIME-OF-DAY-STRING TO TIME-DATE OF DBUG-REC2.
D WRITE DBUGPRT-REC2 FORMAT IS "DBGRC02".

MOVE DIST-ADDRESS TO DST-DATA.
MOVE WH-ADDRESS TO WH-DATA.
MOVE CID OF CUSTOMER-REC TO CID OF PAYMNT-O.

IF CCREDT OF CUSTOMER-REC = "BC"
THEN MOVE "17" TO OUTPUT-FMT-NUM-3
ELSE MOVE "19" TO OUTPUT-FMT-NUM-3
END-IF.

PAYMENT-TRANSACTION-EXIT.
MOVE PAYMNT-O TO OUT-TRAN.
GOBACK.

CUSTOMER-BY-NUMBER.
MOVE CID OF CUSTOMER-REC TO KVAL OF CSTMW-KEY-DATA(1)
MOVE CDID OF CUSTOMER-REC TO KVAL OF CSTMW-KEY-DATA(2)
MOVE CWID OF CUSTOMER-REC TO KVAL OF CSTMW-KEY-DATA(3)

CALL "qdbrunha" USING BY VALUE
CSTMW-HASH-PTR FETCHDATE CUST-KEYS
CSTMW-KEY-PTR CSTMW-DEF-PTR RET-CODE-PTR

IF RET-CODE > 0
IF RET-CODE = 100
MOVE "77" TO OUTPUT-FMT-NUM-3
ROLLBACK
GO TO PAYMENT-TRANSACTION-EXIT
ELSE
IF RET-CODE = 812
ROLLBACK

```

```

GO TO PAYMENT-TRANSACTION
ELSE
    ROLLBACK
    GO TO PROGRAM-END
END-IF
END-IF
END-IF.
CUSTOMER-BY-NAME.
MOVE CWID OF CUSTOMER-REC TO CWIDR.
MOVE CDID OF CUSTOMER-REC TO CDIDR.
MOVE CLASTK TO CLASTR.
EXEC SQL DECLARE C1 CURSOR FOR
SELECT CID
FROM CSTMRLFCRT
WHERE CLAST = :CLASTR AND CDID = :CDIDR AND
CWID = :CWIDR
END-EXEC.
EXEC SQL OPEN C1
END-EXEC.
EXEC SQL FETCH C1 FOR 100 ROWS INTO :HOST-ARRAY-ELEMENT
END-EXEC.
MOVE SQLERRD OF SQLCA(3) TO XX.
EXEC SQL CLOSE C1
END-EXEC.
IF XX = 0
MOVE *77* TO OUTPUT-FMT-NUM-3
ROLLBACK
GO TO PAYMENT-TRANSACTION-EXIT
END-IF
IF XX = 100
CALL "TOOMANY" USING CWIDR, CDIDR, CLASTR, CUSTREL
ELSE
COMPUTE CUST-INDEX = (XX + 1) / 2.
MOVE HOST-ARRAY-ELEMENT(CUST-INDEX) TO
CID OF CUSTOMER-REC.
PERFORM CUSTOMER-BY-NUMBER.
CUSTOMER-BY-NAME-EXIT.
SET-UP-ROUTINE.
CALL "gettime" USING TIME-OF-DAY.
PERFORM OPEN-ROUTINE.
MOVE "CSTMTR" TO CSTMTR-HASH-NAME.
SET CSTMTR-HASH-PTR TO ADDRESS OF CSTMTR-HASH-NAME.
SET KEY-CODE-PTR TO ADDRESS OF RET-CODE.
SET CSTMTR-DEF-PTR TO ADDRESS OF CUSTOMER-REC.
SET CSTMTR-KEY-PTR TO ADDRESS OF CSTMTR-KEY.
MOVE SPACES TO HDATA OF HIST-INFO.
*OPEN FILES
PERFORM CLOSE-ROUTINE.
OPEN EXTEND HISTORY-FILE.
OPEN I-O WAREHOUSE-FILE.
OPEN I-O DISTRICT-FILE.
D OPEN INPUT ISO-IFILE.
D OPEN OUTPUT DEBUG-FILE2.
CLOSE-ROUTINE.
*CLOSE FILES
CLOSE DISTRICT-FILE.
CLOSE WAREHOUSE-FILE.
CLOSE HISTORY-FILE.
D CLOSE ISO-IFILE.
D CLOSE DEBUG-FILE2.
PROGRAM-END.
STOP RUN.

```

## D.25 PAYSRRV.C:

```

#include <Common.h>
#include <qsoasynch.h>
#include <stdio.h>
char *PAYMENTMOD(short *, char *, char *);
#define MAX_CONN 32
main(int argc, char **argv)
{
    int sd, snd_len=670, rcv_len=230, port, rc;
    int sd_arr[MAX_CONN]/* the indexes for accepts and use */
    int num_conn;/* the number of connections */
    int accept_index;/* accept index goes from 0 to num_conn - 1 */
    char rcv_buf[MAX_CONN][BufferSize]/* one per connection */
    char snd_buf[BufferSize];
    char cmd[120]/* command holder */
    struct sockaddr_in sin;
    int length;
    Qso_OverlappedIO_t status;
    char buffer[256];
    memset(&status,0,sizeof(status));
    status.bufferLength=rcv_len;
    status.postFlag=1;
    status.fillBuffer=1; /* Fill buffer. */
    /* get the number of connections */
    num_conn = atoi(argv[1]);
    /* num_conn = 2; /* test action */
    if((port=QsoCreateIOCompletionPort())<0) {
        perror("QsoCreateIOCompletionPort() failed");
        exit(-1);
    }
    length=sizeof(sin);
    for( accept_index=0;accept_index<num_conn;accept_index++)
    {
        if((sd_arr[accept_index]=accept(sd,
            (struct sockaddr *)&sin,length))<0)
        {
            perror("accept() in DLVSRVR failed");
            close(sd);
            exit(-1);
        }
        status.descriptorHandle=(void*)sd_arr[accept_index];
        status.buffer=rcv_buf[accept_index];
        if(QsoStartRecv(sd_arr[accept_index],port,&status)<0) {
            perror("QsoStartRecv() failed");
            exit(-1);
        }
    }
    /* end of connect loop */
    system("STRMCTL LCKLVL(*ALL)");
    system("CHGJOB RUNPT(16)");
    do
    {
        if(QsoWaitForIOCompletion(port,&status,0)<0)
        {
            perror("QsoWaitForIOCompletion() failed");
            exit(-1);
        }
        if(status.returnValue<0) {
            printf("QsoRecv() failed, errno=%d",status.errnoValue);
            exit(-1);
        }
        if(status.operationCompleted!=QSOSTARTREC'V) {
            printf("Unexpected completion=%d",status.operationCompleted);
            exit(-1);
        }
        PAYMENTMOD(&first_time, status.buffer, snd_buf);
        if (( snd_buf[0] != '7' )||((snd_buf[0]!='7')&&
            (snd_buf[1] != '7')))
        {
            first_time++;
        }
        if(send((int)status.descriptorHandle,snd_buf,snd_len,0)<0)
        {
            if (( errno != EPIPE )&&( errno != EUNKNOWN ))
            {
                perror("QsoSend(1) failed");
                exit(-1);
            }
        }
        if(QsoStartRecv((int)status.descriptorHandle,port,&status)<0)
        {
            perror("QsoStartRecv() failed");
            exit(-1);
        }
    } while(first_time> 0 );
    for(;;)
    {
        if(QsoWaitForIOCompletion(port,&status,0)<0) {
            perror("QsoWaitForIOCompletion() failed");
            exit(-1);
        }
        if(status.returnValue<0) {
            printf("QsoRecv() failed, errno=%d",status.errnoValue);
            exit(-1);
        }
        if(status.operationCompleted!=QSOSTARTREC'V) {
            printf("Unexpected completion=%d",status.operationCompleted);
            exit(-1);
        }
        PAYMENTMOD(&first_time, status.buffer, snd_buf);
        if(send((int) status.descriptorHandle,snd_buf,snd_len,0)<0)
        {
            if (( errno != EPIPE )&&( errno != EUNKNOWN ))
            {
                perror("QsoSend(2) failed");
                exit(-1);
            }
        }
        if(QsoStartRecv((int)status.descriptorHandle,port,&status)<0)
        {
            perror("QsoStartRecv() failed");
            exit(-1);
        }
    } /* for loop */
    return 0;
}

```

```

        perror("QsoWaitForIOCompletion() failed");
        exit(-1);
    }
    if(status.returnValue<0) {
        printf("QsoRecv() failed, errno=%d",status.errnoValue);
        exit(-1);
    }
    if(status.operationCompleted!=QSOSTARTREC'V) {
        printf("Unexpected completion=%d",status.operationCompleted);
        exit(-1);
    }
    PAYMENTMOD(&first_time, status.buffer, snd_buf);
    if (( snd_buf[0] != '7' )||((snd_buf[0]!='7')&&
        (snd_buf[1] != '7')))
    {
        first_time++;
    }
    if(send((int)status.descriptorHandle,snd_buf,snd_len,0)<0)
    {
        if (( errno != EPIPE )&&( errno != EUNKNOWN ))
        {
            perror("QsoSend(1) failed");
            exit(-1);
        }
    }
    if(QsoStartRecv((int)status.descriptorHandle,port,&status)<0)
    {
        perror("QsoStartRecv() failed");
        exit(-1);
    }
} while(first_time> 0 );
for(;;)
{
    if(QsoWaitForIOCompletion(port,&status,0)<0) {
        perror("QsoWaitForIOCompletion() failed");
        exit(-1);
    }
    if(status.returnValue<0) {
        printf("QsoRecv() failed, errno=%d",status.errnoValue);
        exit(-1);
    }
    if(status.operationCompleted!=QSOSTARTREC'V) {
        printf("Unexpected completion=%d",status.operationCompleted);
        exit(-1);
    }
    PAYMENTMOD(&first_time, status.buffer, snd_buf);
    if(send((int) status.descriptorHandle,snd_buf,snd_len,0)<0)
    {
        if (( errno != EPIPE )&&( errno != EUNKNOWN ))
        {
            perror("QsoSend(2) failed");
            exit(-1);
        }
    }
    if(QsoStartRecv((int)status.descriptorHandle,port,&status)<0)
    {
        perror("QsoStartRecv() failed");
        exit(-1);
    }
} /* for loop */
return 0;
}

```

## D.26 STKLVL.CBL:

```

PROCESS NOTRUNC NORANGE.
IDENTIFICATION DIVISION.
PROGRAM-ID. STKLVLMOD.
AUTHOR. TPCC.
INSTALLATION. IBM-ROCHESTER.
DATE-WRITTEN.
DATE-COMPILED.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-AS400.
OBJECT-COMPUTER. IBM-AS400.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT DSTRCT
ASSIGN TO DATABASE-DSTRCT
ORGANIZATION IS INDEXED
ACCESS MODE IS RANDOM
RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
WITH DUPLICATES
FILE STATUS IS DISTRICT-FILE-STATUS.
SELECT ORDERLINE
ASSIGN TO DATABASE-ORDLINLF
ORGANIZATION IS INDEXED
ACCESS MODE IS DYNAMIC
FILE STATUS IS ORDERLINE-FILE-STATUS
RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
WITH DUPLICATES.
I-O-CONTROL.
COMMITMENT CONTROL FOR DSTRCT ORDERLINE.
DATA DIVISION.
FILE SECTION.
FD DSTRCT
LABEL RECORDS ARE STANDARD
DATA RECORD IS DSTRCT-RECORD.
01 DSTRCT-RECORD.
COPY DDS-DSRCD OF DSTRCT.
FD ORDERLINE
LABEL RECORDS ARE STANDARD
DATA RECORD IS ORDERLINE-RECORD.
01 ORDERLINE-RECORD.
COPY DDS-OLRCD OF ORDERLINE.
*****
WORKING-STORAGE SECTION.
*****
01 TRAN-OUTPUT.
05 STKLVL-C REDEFINES TRAN-OUT.
06 RESULT PIC X(1).
06 FILLER PIC X(1).
06 BLWSTK PIC S9(2) COMP-4.
01 STOCK-REC.
05 STOCK-DEF.
06 STOCK-IN.
08 STIID PIC S9(9) COMP-4.
08 STIID PIC S9(4) COMP-4.
06 STQTY PIC S9(3) COMP-3.
06 DIST-INFO OCCURS 10 TIMES.
07 DIST-IN PIC X(24).

```

```

06 STYTD PIC S9(9) COMP-4.
06 STORDRS PIC S9(3) COMP-3.
06 STREMODR PIC S9(3) COMP-3.
06 STDATA PIC X(50).

01 STOCK-KEY.
03 STOCK-KEY-DATA OCCURS 5 TIMES.
05 KNAM PIC X(10).
05 KVAL PIC S9(9) COMP-4.

01 STOCK-HASH-PTR USAGE POINTER.

01 STOCK-HASH-NAME PIC X(10).
01 FUNCT PIC X(1).
01 KEYS PIC S9(9) COMP-4.
01 RET-CODE PIC S9(9) COMP-4.

01 RET-CODE-PTR USAGE POINTER.
01 STOCK-DEF-PTR USAGE POINTER.
01 STOCK-KEY-PTR USAGE POINTER.

01 UPDATER PIC X(1) VALUE "3".
01 FETCHR PIC X(1) VALUE "1".
01 FETCHUPDATE PIC X(1) VALUE "2".

01 STOCK-KEYS PIC 9(1) COMP-4 VALUE 2.

01 TRANSACTION-INPUT.
06 TXN-TYPE PIC X.
06 JOBNAME PIC X(10).
06 CLIENT-INPUT PIC X(202).
06 STKLVL-I REDEFINES CLIENT-INPUT.
08 WID PIC S9(4) COMP-4.
08 DID PIC S9(4) COMP-4.
08 THRESHOLD PIC S9(4) COMP-4.
08 BLWSTK PIC S9(4) COMP-4.

01 ITEM-TABLE.
05 ITEM-TABLE-ELEMENT OCCURS 300 TIMES.
07 ITEM-TABLE-KEY.
09 ITEM-TABLE-IID PIC S9(6).

77 DISTRICT-FILE-STATUS PIC X(2).
77 ORDERLINE-FILE-STATUS PIC X(2).

01 NUMBER-OF-ITEMS PIC S9(5) COMP-4.

77 DUPS PIC S9(5) COMP-4.
77 HIORDER PIC S9(9) COMP-4.
77 LOORDER PIC S9(9) COMP-4.
77 IN-LOOP PIC S9(5) COMP-4.
77 OUT-LOOP PIC S9(5) COMP-4.

*****
LINKAGE SECTION.
*****
01 FIRST-TIME PIC S9(4) COMP-4.
01 INP-TRAN PIC X(230).
01 OUT-TRAN PIC X(670).

PROCEDURE DIVISION USING FIRST-TIME INP-TRAN OUT-TRAN.
*****
START-OF-PROGRAM.
*****
STOCK-WATCH-PROGRAM.

IF FIRST-TIME > 0
PERFORM SET-UP-ROUTINE.

MOVE INP-TRAN TO TRANSACTION-INPUT.

MOVE WID OF STKLVL-I TO DWID, OLWID, STWID.

MOVE DID OF STKLVL-I TO DID OF DSRCT, OLDID.

READ-DSTRCT.
READ DSTRCT.
IF DISTRICT-FILE-STATUS NOT EQUAL "00"
IF DISTRICT-FILE-STATUS EQUAL "9D"
GO TO READ-DSTRCT
ELSE
ROLLBACK
MOVE "I" TO RESULT OF STKLVL-I
MOVE STKLVL-I TO OUT-TRAN
GOBACK
END-IF
END-IF.

* Select all Detail Order Lines for the Last 20 Orders
* that are below the threshold and also match a unique
* stock item.

SUBTRACT 1 FROM DNXTOR GIVING HIORDER.
SUBTRACT 20 FROM DNXTOR GIVING LOORDER.
READ-ORDERLINE-FILE.

MOVE LOORDER TO OLOID OF ORDERLINE-RECORD.

MOVE 0 TO NUMBER-OF-ITEMS.

READ ORDERLINE.

MOVE OLIID TO STIID.

MOVE WID OF STKLVL-I TO KVAL OF STOCK-KEY-DATA(2)
MOVE STIID TO KVAL OF STOCK-KEY-DATA(1)

CALL "qdbrunha" USING BY VALUE
STOCK-HASH-PTR FETCHR STOCK-KEYS STOCK-KEY-PTR
STOCK-DEF-PTR RET-CODE-PTR
IF RET-CODE > 0 THEN
IF RET-CODE = 1 THEN
GO TO END-PGM
END-IF
IF RET-CODE = 2 THEN
GO TO END-PGM
END-IF
END-IF
IF STQTY IS LESS THAN THRESHOLD
ADD 1 TO NUMBER-OF-ITEMS
MOVE STIID TO ITEM-TABLE-IID(NUMBER-OF-ITEMS)
END-IF.

READ-ORDERLINE-FILE-NEXT.

READ ORDERLINE NEXT AT END GO TO SORT-ITEM-TABLE
END-READ

IF OLDID NOT EQUAL TO DID OF STKLVL-I
GO TO SORT-ITEM-TABLE
END-IF

IF (OLOID > HIORDER) THEN GO TO SORT-ITEM-TABLE
ELSE
MOVE OLIID TO STIID

MOVE WID OF STKLVL-I TO KVAL OF STOCK-KEY-DATA(2)
MOVE STIID TO KVAL OF STOCK-KEY-DATA(1)

CALL "qdbrunha" USING BY VALUE

```

```

STOCK-HASH-PTR FETCHR STOCK-KEYS STOCK-KEY-PTR
STOCK-DEF-PTR RET-CODE-PTR
IF RET-CODE > 0 THEN
IF RET-CODE = 1 THEN
GO TO END-PGM
END-IF
IF RET-CODE = 2 THEN
GO TO END-PGM
END-IF
END-IF
IF STQTY IS LESS THAN THRESHOLD
ADD 1 TO NUMBER-OF-ITEMS
MOVE STIID TO ITEM-TABLE-IID(NUMBER-OF-ITEMS)
END-IF
END-IF

GO TO READ-ORDERLINE-FILE-NEXT.

SORT-ITEM-TABLE.
COMMIT

PERFORM DISTINCT-SORT.

DISPLAY-ANSWER.

MOVE "C" TO RESULT OF STKLVL-I
MOVE STKLVL-I TO OUT-TRAN.

GOBACK.

DISTINCT-SORT.
IF NUMBER-OF-ITEMS > 0
MOVE 1 TO BLWSTK OF STKLVL-I
PERFORM VARYING OUT-LOOP FROM 2 BY 1 UNTIL OUT-LOOP
IS GREATER THAN NUMBER-OF-ITEMS
MOVE 0 TO DUPS
PERFORM VARYING IN-LOOP FROM 1 BY 1 UNTIL IN-LOOP
IS EQUAL TO OUT-LOOP
IF ITEM-TABLE-ELEMENT (IN-LOOP) =
ITEM-TABLE-ELEMENT (OUT-LOOP)
ADD 1 TO DUPS
END-IF
END-PERFORM
IF DUPS = 0
ADD 1 TO BLWSTK OF STKLVL-I
END-IF
END-PERFORM
ELSE MOVE 0 TO BLWSTK OF STKLVL-I.
MOVE "C" TO RESULT OF STKLVL-I
MOVE STKLVL-I TO OUT-TRAN.

GOBACK.

SET-UP-ROUTINE.
CLOSE DSTRCT
ORDERLINE.

OPEN INPUT ORDERLINE
INPUT DSTRCT.

MOVE "STOCK" TO STOCK-HASH-NAME.
MOVE "STWID" TO KNAM OF STOCK-KEY-DATA(2).
MOVE "STIID" TO KNAM OF STOCK-KEY-DATA(1).

SET STOCK-HASH-PTR TO ADDRESS OF STOCK-HASH-NAME.
SET RET-CODE-PTR TO ADDRESS OF RET-CODE.
SET STOCK-DEF-PTR TO ADDRESS OF STOCK-REC.
SET STOCK-KEY-PTR TO ADDRESS OF STOCK-KEY.

END-PGM.
STOP RUN.

```

## D.27 STKSRRV.C:

```

#include <Common.h>
#include <qsosync.h>
#include <stdio.h>

main(int argc, char **argv)
{
    int sd, snd_len=670, rcv_len=230, port, rc;
    int sd_arr[MAX_CONN]; /* The indexes for accepts and use */
    int num_conn; /* the number of connections */
    int accept_index; /* accept index goes from 0 to num_conn - 1 */
    char rcv_buf[MAX_CONN][BufferSize]; /* one per connection */
    char snd_buf[BufferSize];

    struct sockaddr_in sin;
    int length;
    char Qso_OverlappedIo_t status;
    char buffer[256];

    memset(&status, 0, sizeof(status));
    status.bufferLength=rcv_len;
    status.postFlag=1; /* Always post to I/O completion port */
    status.fillBuffer=1; /* Fill buffer. */
    /* get the number of connections */
    num_conn = atoi(argv[1]);

    if((port=QsoCreateIOCompletionPort())<0) {
        perror("nQsoCreateIOCompletionPort() failed");
        exit(-1);
    }

    length=sizeof(sin);
    for(accept_index=0;accept_index<num_conn;accept_index++)
    {
        if((sd_arr[accept_index]=accept(sd,
            (struct sockaddr *)&sin, length))<0)
        {
            perror("accept() in DLVSRV failed");
            close(sd);
            exit(-1);
        }
        status.descriptorHandle=(void*)sd_arr[accept_index];
        status.buffer=rcv_buf[accept_index];
        if(QsoStartRecv(sd_arr[accept_index], port, &status)<0) {
            perror("nQsoStartRecv() failed");
            exit(-1);
        }
    }

    /* end of connect loop */
    system("STRMCTCL LCKLVL(*CS)");
    system("CHGJOB RUNPT(20)");
    system("QVRDBF FILE(ORDLINLF) TOFILE(ORDLINLF) NBRRCDS(16)");
    do
    {
        if(QsoWaitForIOCompletion(port, &status, 0)<0)
        {
            perror("nQsoWaitForIOCompletion() failed");
            exit(-1);
        }
    }
    if(status.returnValue<=0)
    {
        printf("nRecv() failed, errno=%d", status.errnoValue);
        exit(-1);
    }
}

```

```

}
if(status.operationCompleted!=QSOSTARTREC)
{
    printf("\nUnexpected completion=sd", status.operationCompleted);
    exit(-1);
}
STKLVMOD(&first_time, status.buffer, snd_buf);
if ( snd_buf[0] == 'G' )/* check the NEWORD-RESULT */
{
    first_time--;
}
if(send((int)status.descriptorHandle, snd_buf, snd_len, 0)<0)
{
    if (( errno != EPIPE ) && ( errno != EUNKNOWN ))
    {
        perror("nsend(1) failed");
        exit(-1);
    }
}
if(QsoStartRecv((int)status.descriptorHandle, port, &status)<0)
{
    perror("nQsoStartRecv() failed");
    exit(-1);
}
} while(first_time > 0);
for(;;) {
    if(QsoWaitForIOCompletion(port, &status, 0)<0) {
        perror("nQsoWaitForIOCompletion() failed");
        exit(-1);
    }
    if(status.returnValue<=0) {
        printf("nRecv() failed, errno=sd", status.errnoValue);
        exit(-1);
    }
}
if(status.operationCompleted!=QSOSTARTREC) {
    printf("\nUnexpected completion=sd", status.operationCompleted);
    exit(-1);
}
STKLVMOD(&first_time, status.buffer, snd_buf);
if(send((int) status.descriptorHandle, snd_buf, snd_len, 0)<0)
{
    if (( errno != EPIPE ) && ( errno != EUNKNOWN ))
    {
        perror("nsend(2) failed");
        exit(-1);
    }
}
if(QsoStartRecv((int)status.descriptorHandle, port, &status)<0)
{
    perror("nQsoStartRecv() failed");
    exit(-1);
}
} /* for loop */
return 0;
}

```

## D.28 TOOMANY.CBL:

```

PROCESS NOTRUNC NORANGE.
IDENTIFICATION DIVISION.

* TOOMANY Program
*
* Files Accessed: - CSTM (Read)
* Views Used: - CSTMRFNAM (cl - SQL)

PROGRAM-ID. TOOMANY.
AUTHOR. DEPT-53g.
INSTALLATION. IBM-ROCHESTER.
DATE-WRITTEN. 09-03-95.
DATE-COMPILED. 12-14-92.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-AS400.
** SOURCE-COMPUTER. IBM-AS400. WITH DEBUGGING MODE.
OBJECT-COMPUTER. IBM-AS400.

DATA DIVISION.

*****
WORKING-STORAGE SECTION
*****
01 HOST-VARIABLES.
05 VWID PIC S9(4) COMP-4.
05 VVID PIC S9(2) COMP-4.
05 VCID PIC S9(6) COMP-4.
05 VCLAST PIC X(16).
05 VCFIRST PIC X(16).
05 VCOUNT PIC S9(4) COMP-4.
01 I PIC S9(4) COMP-4.

*****
* SQL INCLUDE *
*****
EXEC SQL
INCLUDE SQLCA
END-EXEC.

*****
* SQL ERROR/WARNING Messages *
*****
EXEC SQL
WHENEVER SQLERROR CONTINUE
END-EXEC.

EXEC SQL
WHENEVER SQLWARNING CONTINUE
END-EXEC.

*****
LINKAGE SECTION.
*****
01 FULL-KEY.
05 PART-KEY.
06 CID PIC S9(6) COMP-4.
06 DSTRCT PIC S9(2) COMP-4.
06 WRHS PIC S9(4) COMP-4.
05 CLAST PIC X(16).
PROCEDURE DIVISION USING FULL-KEY.

*****
START-OF-PROGRAM.
*****
MOVE WRHS TO VWID.
MOVE DSTRCT TO VVID.
MOVE CLAST TO VCLAST.

EXIT.

```

```

EXEC SQL
SELECT COUNT(*)
INTO :VCOUNT
FROM CSTMRLFCT
WHERE (CWID = :VWID AND CDID = :VVID
AND CLAST = :VCLAST)
END-EXEC.

COMPUTE VCOUNT = 10 * VCOUNT + 1.
COMPUTE VCOUNT = VCOUNT / 20.

EXEC SQL
DECLARE C BY NAME CURSOR FOR
SELECT CWID, CDID, CID, CLAST, CFIRST
FROM CSTMRLFCT
WHERE CWID = :VWID AND CDID = :VVID
AND CLAST = :VCLAST
ORDER BY CWID, CDID, CLAST, CFIRST
END-EXEC.

EXEC SQL
OPEN C BY NAME
END-EXEC.

PERFORM VARYING I FROM 1 BY 1 UNTIL I > VCOUNT

EXEC SQL
FETCH C BY NAME INTO :VWID, :VVID, :VCID,
:VCLAST, :VCFIRST
END-EXEC

END-PERFORM.

MOVE VCID TO CID.

EXIT.

```

## D.29 TPCCUSER.H:

```

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <decimal.h>
#include <quadr.h>
#include <qsrjobi.h>
#pragma mapinc("wrhs", "LIBL/WRHS01(*ALL)", "both", "d_p", "tpcc")
#pragma mapinc("dstrct", "LIBL/DSTRCT01(*ALL)", "both", "d_p", "tpcc")
#pragma mapinc("hstry", "LIBL/HSTRV01(*ALL)", "both", "d_p", "tpcc")
#pragma mapinc("ordin", "LIBL/ORDLINPF01(*ALL)", "both", "d_p", "tpcc")
#pragma mapinc("orders", "LIBL/ORDERSPF01(*ALL)", "both", "d_p", "tpcc")
#pragma mapinc("neword", "LIBL/NEWORDPF01(*ALL)", "both", "d_p", "tpcc")
#pragma mapinc("custm", "LIBL/CSTMRF01(*ALL)", "both", "d_p", "tpcc")
#pragma mapinc("stock", "LIBL/STOCKPF01(*ALL)", "both", "d_p", "tpcc")

typedef struct { /* space rtn error code structure */
    int bytes_prov;
    int bytes_avail;
    char except_id[7];
    char reserved[1];
    char except_data[50];
} error_code;

/* Format name and data space number structure */
typedef _Packed struct {
    short ds_num;
    long zero1;
    long zero2;
} Format_Name;

```



# Appendix E. RTE Scripts

## E.1 RTE Parameters

RTE RUN-PROFILE (TO BE SENT TO AUDITOR)

-----  
Customer Lastname Constant = 117

Transaction Weights

-----  
Payment: 0.000000  
Order Status: 0.000000  
Delivery: 0.000000  
Stock Level: 0.000000

DELAY CONSTANTS

-----  
Menu Delay= 0.100000  
Response Delay= 0.100000

Think Times

-----  
NEW ORDER: 12.020000  
PAYMENT: 12.020000  
ORDER STATUS: 10.020000  
DELIVERY: 5.020000  
STOCK LEVEL: 5.020000

SLAVE #0

-----  
RTE MACHINE: slave1  
CONNECTS TO CLIENT: 8.5.8.10  
# OF USERS: 1000

SLAVE #1

-----  
RTE MACHINE: slave2  
CONNECTS TO CLIENT: 8.5.8.10  
# OF USERS: 1000

SLAVE #2

-----  
RTE MACHINE: slave3  
CONNECTS TO CLIENT: 8.5.8.10  
# OF USERS: 1000

SLAVE #3

-----  
RTE MACHINE: slave4  
CONNECTS TO CLIENT: 8.5.8.10  
# OF USERS: 1000

SLAVE #4

-----  
RTE MACHINE: slave5  
CONNECTS TO CLIENT: 8.5.8.10  
# OF USERS: 1000

SLAVE #5

-----  
RTE MACHINE: slave6  
CONNECTS TO CLIENT: 8.5.8.10  
# OF USERS: 1000

SLAVE #6

-----  
RTE MACHINE: slave7  
CONNECTS TO CLIENT: 8.5.8.10  
# OF USERS: 1000

SLAVE #7

-----  
RTE MACHINE: slave8  
CONNECTS TO CLIENT: 8.5.8.10  
# OF USERS: 1000

SLAVE #8

-----  
RTE MACHINE: slave9  
CONNECTS TO CLIENT: 8.5.7.11  
# OF USERS: 1000

SLAVE #9

-----  
RTE MACHINE: slave10  
CONNECTS TO CLIENT: 8.5.7.11  
# OF USERS: 1000

SLAVE #10

-----  
RTE MACHINE: slave11  
CONNECTS TO CLIENT: 8.5.7.11  
# OF USERS: 1000

SLAVE #11

-----  
RTE MACHINE: slave12  
CONNECTS TO CLIENT: 8.5.7.11  
# OF USERS: 1000

SLAVE #12

-----  
RTE MACHINE: slave13  
CONNECTS TO CLIENT: 8.5.7.11  
# OF USERS: 1000

SLAVE #13

-----  
RTE MACHINE: slave14  
CONNECTS TO CLIENT: 8.5.7.11  
# OF USERS: 1000

SLAVE #14

-----  
RTE MACHINE: slave15  
CONNECTS TO CLIENT: 8.5.7.11  
# OF USERS: 1000

SLAVE #15

-----  
RTE MACHINE: slave16  
CONNECTS TO CLIENT: 8.5.7.11  
# OF USERS: 1000

SLAVE #16

-----  
RTE MACHINE: slave17  
CONNECTS TO CLIENT: 8.5.6.12  
# OF USERS: 1000

SLAVE #17

-----  
RTE MACHINE: slave18  
CONNECTS TO CLIENT: 8.5.6.12  
# OF USERS: 1000

SLAVE #18

-----  
RTE MACHINE: slave19  
CONNECTS TO CLIENT: 8.5.6.12  
# OF USERS: 1000

SLAVE #19

-----  
RTE MACHINE: slave20  
CONNECTS TO CLIENT: 8.5.6.12  
# OF USERS: 1000

SLAVE #20

-----  
RTE MACHINE: slave21  
CONNECTS TO CLIENT: 8.5.6.12  
# OF USERS: 1000

SLAVE #21

-----  
RTE MACHINE: slave22  
CONNECTS TO CLIENT: 8.5.6.12  
# OF USERS: 1000

SLAVE #22

-----  
RTE MACHINE: slave23  
CONNECTS TO CLIENT: 8.5.6.12  
# OF USERS: 1000

SLAVE #23

-----  
RTE MACHINE: slave24  
CONNECTS TO CLIENT: 8.5.6.12  
# OF USERS: 1000

SLAVE #24

-----  
RTE MACHINE: slave25  
CONNECTS TO CLIENT: 8.5.8.13  
# OF USERS: 1000

SLAVE #25

-----  
RTE MACHINE: slave26  
CONNECTS TO CLIENT: 8.5.8.13  
# OF USERS: 1000

SLAVE #26

-----  
RTE MACHINE: slave27  
CONNECTS TO CLIENT: 8.5.8.13  
# OF USERS: 1000

SLAVE #27

-----  
RTE MACHINE: slave28  
CONNECTS TO CLIENT: 8.5.8.13  
# OF USERS: 1000

SLAVE #28

-----  
RTE MACHINE: slave29  
CONNECTS TO CLIENT: 8.5.8.13  
# OF USERS: 1000

SLAVE #29

-----  
RTE MACHINE: slave30  
CONNECTS TO CLIENT: 8.5.8.13  
# OF USERS: 1000

SLAVE #30

-----  
RTE MACHINE: slave31  
CONNECTS TO CLIENT: 8.5.8.13  
# OF USERS: 1000

SLAVE #31

-----  
RTE MACHINE: slave32  
CONNECTS TO CLIENT: 8.5.8.13  
# OF USERS: 1000

SLAVE #32

-----  
RTE MACHINE: slave33  
CONNECTS TO CLIENT: 8.5.7.14  
# OF USERS: 1000

SLAVE #33

-----  
RTE MACHINE: slave34  
CONNECTS TO CLIENT: 8.5.7.14  
# OF USERS: 1000

SLAVE #34

-----  
RTE MACHINE: slave35  
CONNECTS TO CLIENT: 8.5.7.14  
# OF USERS: 1000

SLAVE #35

-----  
RTE MACHINE: slave36  
CONNECTS TO CLIENT: 8.5.7.14  
# OF USERS: 1000

SLAVE #36

-----  
RTE MACHINE: slave37  
CONNECTS TO CLIENT: 8.5.7.14  
# OF USERS: 1000

SLAVE #37

-----  
RTE MACHINE: slave38  
CONNECTS TO CLIENT: 8.5.7.14  
# OF USERS: 1000

SLAVE #38

-----  
RTE MACHINE: slave39  
CONNECTS TO CLIENT: 8.5.7.14  
# OF USERS: 1000

SLAVE #39

-----  
RTE MACHINE: slave40  
CONNECTS TO CLIENT: 8.5.7.14  
# OF USERS: 1000

SLAVE #40



```

RTE MACHINE: slave89
CONNECTS TO CLIENT: 8.5.6.21
# OF USERS: 1000

SLAVE #89
-----
RTE MACHINE: slave90
CONNECTS TO CLIENT: 8.5.6.21
# OF USERS: 1000

SLAVE #90
-----
RTE MACHINE: slave91
CONNECTS TO CLIENT: 8.5.6.21
# OF USERS: 1000

SLAVE #91
-----
RTE MACHINE: slave92
CONNECTS TO CLIENT: 8.5.6.21
# OF USERS: 1000

SLAVE #92
-----
RTE MACHINE: slave93
CONNECTS TO CLIENT: 8.5.6.21
# OF USERS: 1000

SLAVE #93
-----
RTE MACHINE: slave94
CONNECTS TO CLIENT: 8.5.8.22
# OF USERS: 1000

SLAVE #94
-----
RTE MACHINE: slave95
CONNECTS TO CLIENT: 8.5.8.22
# OF USERS: 1000

SLAVE #95
-----
RTE MACHINE: slave96
CONNECTS TO CLIENT: 8.5.8.22
# OF USERS: 1000

SLAVE #96
-----
RTE MACHINE: slave97
CONNECTS TO CLIENT: 8.5.8.22
# OF USERS: 1000

SLAVE #97
-----
RTE MACHINE: slave98
CONNECTS TO CLIENT: 8.5.8.22
# OF USERS: 1000

SLAVE #98
-----
RTE MACHINE: slave99
CONNECTS TO CLIENT: 8.5.8.22
# OF USERS: 1000

SLAVE #99
-----
RTE MACHINE: slave100
CONNECTS TO CLIENT: 8.5.8.22
# OF USERS: 1000

SLAVE #100
-----
RTE MACHINE: slave101
CONNECTS TO CLIENT: 8.5.8.22
# OF USERS: 1000

SLAVE #101
-----
RTE MACHINE: slave102
CONNECTS TO CLIENT: 8.5.7.26
# OF USERS: 1000

SLAVE #102
-----
RTE MACHINE: slave103
CONNECTS TO CLIENT: 8.5.7.26
# OF USERS: 1000

SLAVE #103
-----
RTE MACHINE: slave104
CONNECTS TO CLIENT: 8.5.7.26
# OF USERS: 1000

SLAVE #104
-----
RTE MACHINE: slave105
CONNECTS TO CLIENT: 8.5.7.26
# OF USERS: 1000

SLAVE #105
-----
RTE MACHINE: slave106
CONNECTS TO CLIENT: 8.5.7.26
# OF USERS: 1000

SLAVE #106
-----
RTE MACHINE: slave107
CONNECTS TO CLIENT: 8.5.7.26
# OF USERS: 1000

SLAVE #107
-----
RTE MACHINE: slave108
CONNECTS TO CLIENT: 8.5.7.26
# OF USERS: 1000

SLAVE #108
-----
RTE MACHINE: slave109
CONNECTS TO CLIENT: 8.5.7.26
# OF USERS: 1000

SLAVE #109
-----
RTE MACHINE: slave110
CONNECTS TO CLIENT: 8.5.6.24
# OF USERS: 1000

SLAVE #110
-----
RTE MACHINE: slave111
CONNECTS TO CLIENT: 8.5.6.24
# OF USERS: 1000

SLAVE #111
-----
RTE MACHINE: slave112
CONNECTS TO CLIENT: 8.5.6.24
# OF USERS: 1000

SLAVE #112
-----

```

```

RTE MACHINE: slave113
CONNECTS TO CLIENT: 8.5.6.24
# OF USERS: 1000

SLAVE #113
-----
RTE MACHINE: slave114
CONNECTS TO CLIENT: 8.5.6.24
# OF USERS: 1000

SLAVE #114
-----
RTE MACHINE: slave115
CONNECTS TO CLIENT: 8.5.6.24
# OF USERS: 1000

SLAVE #115
-----
RTE MACHINE: slave116
CONNECTS TO CLIENT: 8.5.6.24
# OF USERS: 1000

SLAVE #116
-----
RTE MACHINE: slave117
CONNECTS TO CLIENT: 8.5.6.24
# OF USERS: 1000

SLAVE #117
-----
RTE MACHINE: slave118
CONNECTS TO CLIENT: 8.5.8.25
# OF USERS: 1000

SLAVE #118
-----
RTE MACHINE: slave119
CONNECTS TO CLIENT: 8.5.8.25
# OF USERS: 1000

SLAVE #119
-----
RTE MACHINE: slave120
CONNECTS TO CLIENT: 8.5.8.25
# OF USERS: 1000

SLAVE #120
-----
RTE MACHINE: slave121
CONNECTS TO CLIENT: 8.5.8.25
# OF USERS: 1000

```

## E.2 TRANSACTIONS.H

```

#include "Random.h"
#include "SharedMem.h"
#include "User_tpc.h"

#include "ClientSocket.h"

struct UserInfo {
    int UserNum;
    int Warehouse;
    int District;
    int StatusID;
    int Error;
    int FormID;
    int TermID;
    int SyncID;
    int Version;
    ClientSocket socket;
    struct drand48_data seedbuf;
    struct timeb t1;
    struct timeb t2;
    TransData tdata;
};

long NURand(int A, int x, int y, long cval, struct drand48_data* seed);
void getname(char* lastname, struct drand48_data* seed);
int Delivery(UserInfo *Info);
int NewOrder(UserInfo *Info);
int OrderStatus(UserInfo *Info);
int Payment(UserInfo *Info);
int StockLevel(UserInfo *Info);

```

## E.3 TRANSACTIONS.C

```

//-----
// MODULE NAME: transactions.C
//
// DESCRIPTION: This file contains "user_transaction" which selects
// and executes each TPC-C transaction. Each of these
// routines generate the data for the TPC-C transaction,
// formats it into appropriate HTML strings, transmits it
// to the web client interface, receives, validates,
// records the output, and collects timestamps during
// key points in the transaction.
//
// FUNCTIONS: NURand getname
// Delivery
// NewOrder
// OrderStatus
// Payment
// StockLevel
//
// (C) COPYRIGHT International Business Machines Corp. 1997
// All Rights Reserved
// Licensed Materials - Property of IBM
//
// MODIFICATION HISTORY:
// Date By Description
// -----
// 2/17/97 T.Thomas Initial version for NT
//
// 8/02/99 C. Floyd Overhaul of code in preparation for TPCC Version 4.
//
//-----
/**
*** TPCC FILE FOR ALL USERS ***
***/
#include <stdio.h>
#include <sys/time.h>

#include "iostream.h"

#include "memory.h"
#include "Transactions.h"

```

```

#include "Sleep.h"

extern SharedMem *pSHM;

/*-----*/
/* NURand() */
/* A: 255 for C_LAST, 1023 for C_ID, 8191 for OL_I_ID */
/* x: 0 for C_LAST, 1 for C_ID and OL_I_ID */
/* y: 999 for C_LAST, 3000 for C_ID, 100000 for OL_I_ID */
/* The "uniform()" function has range of the absolute value of the */
/* difference between the 1st and 2nd parms up to a maximum of 2147483647.*/
/*-----*/
long NURand(int A, int x, int y, long cval, struct drand48_data* seed)
{
    return (((long) uniform(OL, (long)A, seed) |
            (long) uniform((long)x, (long)y, seed)) + cval) % (y - x + 1) + x;
}

/*-----*/
/* getname() */
/* Generates a random number from 0 to 999 inclusive. */
/* A random name is generated by associating a random */
/* string with each digit of the generated number. */
/* Three strings are concatenated to generate "lastname".*/
/*-----*/
void getname(char* lastname, struct drand48_data* seed)
{
    char *last_name_parts[] =
    {
        "BAR",
        "GUSH",
        "ABLE",
        "PRI",
        "PRES",
        "ESE",
        "ANTI",
        "CALLY",
        "ATION",
        "EING"
    };

    int random_num;

    // Generate a random number between 0 and 999 inclusive.
    random_num = NURand(255, 0, 999, pSHM->LASTC, seed);

    // Use the first digit as an index into the last_name_parts array.
    // Then throw away the high order digit.
    strcpy(lastname, last_name_parts[random_num / 100]);
    random_num %= 100;

    // Use the 10s digit as an index into the last_name_parts array.
    // Then throw away the 10s digit.
    strcat(lastname, last_name_parts[random_num / 10]);
    random_num %= 10;

    // Finally use the last digit as an index into the last_name_parts array.
    strcat(lastname, last_name_parts[random_num]);
}

/*-----*/
/* Delivery Transaction */
/*-----*/
int Delivery(UserInfo *Info)
{
    char Result[4096];
    char Request[256];
    char theUrl[128];
    char* pStr;

    char buf[4097];
    buf[sizeof(buf)-1] = 0;

    int bytes, totalBytes;
    unsigned int t1, t2;
    int status;
    int carrier;

    // Identify the transaction ID and state (for debug).
    pSHM->User[Info->UserNum].transaction_id = Delivery_ID;
    pSHM->User[Info->UserNum].transaction_state = Transaction_State0;

    // Build Url to request Delivery Menu
    // Locally mark the beginning of this routine by getting
    // the time relative to when RTE was started.
    cout << "(Transactions.C) Delivery() invoked by " << Info->UserNum << endl;
    t1 = pSHM->timestamp();

    // Create a buffer of
    // o "GET %s HTTP/2.0\r\n\r\n" /* HTTP/2.0\r\n\r\nConnection: Keep-Alive\r\n\r\n"
    // o "/tpcc.dll?FORMID=8d5TERMIID=8d5SYNCD=8d5CMD=8s"
    // o 2
    // o Info->TermID,
    // o Info->SyncID,
    // o ".Delivery.",
    // and send it on the socket.
    sprintf(theUrl,
            MAIN_MENU_FMT,
            MAIN_MENU_FORM_ID,
            Info->TermID,
            Info->SyncID,
            ".Delivery.");
    sprintf(Request, GET_REQUEST, theUrl);
    if (Info->socket.write(Request) < 0)
    {
        perror("(Transactions.C) Delivery() failure on write to socket due to: ");
        return 30;
    }

    // Iteratively read up to 4096 bytes from the socket until we find "</HTML>".
    // Identify the transaction state and byte count (for debug).
    pSHM->User[Info->UserNum].transaction_state = Transaction_State1;
    pSHM->User[Info->UserNum].transaction_bytes = 0;
    totalBytes = 0;
    memset(Result, 0, sizeof(Result));
    while(totalBytes < sizeof(Result))
    {
        if ((bytes = Info->socket.read(buf, sizeof(buf)-1)) < 0) return 40;
        pSHM->User[Info->UserNum].transaction_bytes = totalBytes + bytes;
        if ((totalBytes + bytes) > sizeof(Result)) return 50;
        memcpy(sResult(totalBytes), buf, bytes);
        if (strstr(Result, "</HTML>")) break;
        totalBytes += bytes;
    }
    pSHM->User[Info->UserNum].transaction_state = Transaction_State2;

    // If the Result buffer does not contain "TPC-C Delivery", return.
    if (strstr(Result, DELIVERY_EXPECT_STRING) == NULL)
    {
        cout << "(Transactions.C) Delivery() received the following string: "
        << Result << endl;
        return 3;
    }

    // Sleep for the number of seconds specified for the DELIVERY option.
    Sleep(pSHM->menu[DELIVERY]*1000.0);
    pSHM->User[Info->UserNum].transaction_state = Transaction_State3;

    // After the socket read-wait and the sleep, locally mark the end of
    // Menu response time and the beginning of keying time.
    t2 = pSHM->timestamp();

    // Subtract the time that the last transaction was finished from
    // the time at the beginning of this routine to produce
    // the number of milliseconds of think time.
    Info->tdata.think_ms = t1 - Info->tdata.trans_finish_ms;

    // Copy the transaction data into the shared memory's output buffer (finished_trans),
    // but exclude initial transaction value and other non-transaction types.
    if ((Info->tdata.trans_type > 0) && (Info->tdata.trans_type < MAX_TRAN_TYPE))
        pSHM->transaction_done(&Info->tdata);

    // Calculate the time (in milliseconds) from the beginning of this routine
    // to (essentially) right now.
    Info->tdata.menu_ms = t2 - t1;

    // Re-initialize the "status".
    Info->tdata.status = 0;

    // Begin Key Time

    // Calculate data for delivery input screen by
    // producing a random number between 1 and 10, inclusive.
    // I.e. carrier # is 1 - 10.
    carrier = (char)uniform(1L, 10L, &Info->seedbuf);

    //Build Url to fill in Delivery form.
    // o "/tpcc.dll?PI=*FORMID=8d5TERMIID=8d5SYNCD=8d"
    // o 5
    // o Info->TermID,
    // o Info->SyncID,
    // o "4OCD=8d5CMD=Process" where 8d is 1 - 10.
    sprintf(theUrl,
            MAIN_FORM_FMT,
            DELIVERY_FORM_ID,
            Info->TermID,
            Info->SyncID);
    sprintf(theUrl+strlen(theUrl), "%OCD=8d5CMD=Process", carrier);

    // Fill in additional data
    Info->tdata.trans_type = DELIVERY;
    Info->tdata.data = 0;
    Info->tdata.wd = Info->Warehouse;
    Info->tdata.wd_id = Info->District;
    Info->tdata.d_id = 0;
    Info->tdata.c_id = 0;
    Info->tdata.o_id = 0;
    Info->tdata.orders = 0;
    Info->tdata.remote = 0;

    // Mark End of Keying time and beginning of delivery response time.
    // Determine how long to sleep to meet minimum keying time requirement
    // and take a short nap.
    // DELIVERY_KEYING_TIME = 2000 milliseconds
    // We are subtracting from DELIVERY_KEYING_TIME the variable time
    // that it took us to get from the last sleep to here.
    // Identify the transaction state (for debug).
    pSHM->User[Info->UserNum].transaction_state = Transaction_State4;
    unsigned int wait_time = DELIVERY_KEYING_TIME + t2 - pSHM->timestamp();
    if (wait_time > 0)
        Sleep((unsigned int)wait_time);

    // Determine the time at the beginning of the delivery response time.
    // Determine from that the amount of time past since the prior sleep.
    Info->tdata.trans_start_ms = pSHM->timestamp();
    Info->tdata.key_ms = Info->tdata.trans_start_ms - t2;

    sprintf(Request, GET_REQUEST, theUrl);
    if (Info->socket.write(Request) < 0)
    {
        perror("(Transactions.C) Delivery() failure on write to socket due to: ");
        return 31;
    }

    // Iteratively read up to 4096 bytes from the socket until we find "</HTML>".
    // Identify the transaction state and byte count (for debug).
    pSHM->User[Info->UserNum].transaction_state = Transaction_State5;
    pSHM->User[Info->UserNum].transaction_bytes = 0;
    totalBytes = 0;
    memset(Result, 0, sizeof(Result));
    while(totalBytes < sizeof(Result))
    {
        if ((bytes = Info->socket.read(buf, sizeof(buf)-1)) < 0) return 41;
        pSHM->User[Info->UserNum].transaction_bytes = totalBytes + bytes;
        if ((totalBytes + bytes) > sizeof(Result)) return 51;
        memcpy(sResult(totalBytes), buf, bytes);
        if (strstr(Result, "</HTML>")) break;
        totalBytes += bytes;
    }
    pSHM->User[Info->UserNum].transaction_state = Transaction_State6;

    // Search the Result string for "TPC-C Delivery".
    if (strstr(Result, DELIVERY_EXPECT_STRING) == NULL)
    {
        cout << "(Transaction.C) Delivery() received the following string: "
        << Result << endl;
        return 5;
    }

    // Scan Result string for "NAME=\"STATUSID\" VALUE=\"\"
    // and then point past it.
    // If the value at this location is >0,
    // use it as a status and copy the transaction data into
    // the shared memory's output buffer (finished_trans).
    if (pStr = strstr(Result, STATUS_ID_STRING))
    {
        pStr += strlen(STATUS_ID_STRING);
        if ((status = atoi(pStr)) > 0)
        {
            Info->tdata.status = status;
            pSHM->transaction_done(&Info->tdata);
        }
    }

    // Transaction response time includes delay to display data on screen.
    // Identify the transaction state (for debug).
    Sleep(pSHM->response[DELIVERY]*1000.0);
    pSHM->User[Info->UserNum].transaction_state = Transaction_State7;

    // Mark End of Transaction response time, begin Think time
    Info->tdata.trans_finish_ms = pSHM->timestamp();
    Info->tdata.trans_ms = Info->tdata.trans_finish_ms - Info->tdata.trans_start_ms;
    double think_time = neg_exp_4(pSHM->think[DELIVERY], &Info->seedbuf);
    if ((think_time*1000.0) > 0)
        Sleep((unsigned) think_time*1000.0);

    return 1;
}

/*-----*/
/* New Order Transaction */
/*-----*/
int NewOrder(UserInfo *Info)
{
    char Result[4096];
    char theUrl[1024];
}

```

```

char Request[256];
char buf[4097];
buf[sizeof(buf)-1] = 0;

neworder_struct neworder;
int RBtrans, remoteflag, i;
int t1, t2;
int status;
int whses;
int low_whse = 1;
int rollback = 0;

int bytes, totalBytes;
char* pStr;

// Identify the transaction ID and state (for debug).
pSHM->User[Info->UserNum].transaction_ID = NewOrder_ID;
pSHM->User[Info->UserNum].transaction_state = Transaction_State0;

// Start NewOrder Menu Response Time
// Locally mark the beginning of this routine by getting
// the time relative to when RTE was started.
cout << "(Transactions.C) NewOrder() invoked by " << Info->UserNum << endl;
t1 = pSHM->timestamp();

// Build Url to Request NewOrder Menu
// Create a buffer of
// o "GET /s HTTP/2.0\r\nAccept: */*\r\nConnection: Keep-Alive\r\n\r\n"
// o "/tpcc.dll?FORMID=8d45TERMINID=8d45SYNCRID=8d45CMD=8s"
// o 2
// o Info->TermID,
// o Info->SyncID,
// o ".NewOrder.."
// and send it on the socket.
sprintf(theUrl,
        MAIN_MENU_FMT,
        MAIN_MENU_FORM_ID,
        Info->TermID,
        Info->SyncID,
        ".NewOrder.");
sprintf(Request, GET_REQUEST, theUrl);
if (Info->socket.write(Request) < 0)
{
    perror("(Transactions.C) NewOrder() failure on write to socket due to: ");
    return 32;
}

// Iteratively read up to 4096 bytes from the socket until we find "</HTML>".
// Identify the transaction state and byte count (for debug).
pSHM->User[Info->UserNum].transaction_state = Transaction_State1;
pSHM->User[Info->UserNum].transaction_bytes = 0;
totalBytes = 0;
memset(Result, 0, sizeof(Result));
while(totalBytes < sizeof(Result))
{
    if ((bytes = Info->socket.read(buf, sizeof(buf)-1)) < 0) return 42;
    pSHM->User[Info->UserNum].transaction_bytes = totalBytes + bytes;
    if ((totalBytes + bytes) > sizeof(Result)) return 52;
    memcpy(Result[totalBytes], buf, bytes);
    if (strstr(Result, "</HTML>")) break;
    totalBytes += bytes;
}
pSHM->User[Info->UserNum].transaction_state = Transaction_State2;
/* cout << "(Transaction.C) NewOrder() received the following string of length: "
   << totalBytes << endl; */

// If the Result buffer does not contain "TPC-C New Order", return.
if (strstr(Result, NEW_ORDER_EXPECT_STRING) == 0)
{
    cout << "(Transaction.C) NewOrder() received the following string: "
         << Result << endl;
    return 7;
}

// Sleep for the number of seconds specified for the NEWORDER option.
Sleep(pSHM->menu[NEWORDER]*1000.0);
pSHM->User[Info->UserNum].transaction_state = Transaction_State3;

// Mark End of Menu response time, beginning of keying time
// After the socket read-wait and the sleep, locally mark the end of
// Menu response time and the beginning of keying time.
t2 = pSHM->timestamp();

// Subtract the time that the last transaction was finished from
// the time at the beginning of this routine to produce
// the number of milliseconds of think time.
Info->tdata.think_ms = t1 - Info->tdata.trans_finish_ms;

// Copy the transaction data into the shared memory's output buffer (finished_trans),
// but exclude initial transaction value and other non-transaction types.
if ((Info->tdata.trans_type > 0) && (Info->tdata.trans_type < MAX_TRAN_TYPE))
    pSHM->transaction_done(&Info->tdata);

// Calculate the time (in milliseconds) from the beginning of this routine
// to (essentially) right now.
Info->tdata.menu_ms = t2 - t1;

// Re-initialize the "status".
Info->tdata.status = 0;

// If the Result string contains "NAME=\"STATUSID\" VALUE=\"",
// and if the value after this string is >0,
// use it as a status and copy the transaction data into
// the shared memory's output buffer (finished_trans).
if (pStr = strstr(Result, STATUS_ID_STRING))
{
    pStr += strlen(STATUS_ID_STRING);
    if ((status = atoi(pStr)) > 0)
    {
        Info->tdata.status = status;
        pSHM->transaction_done(&Info->tdata);
    }
}

// Reinitialize portions of the TransData.
Info->tdata.data = 0;
Info->tdata.orders = 0;
Info->tdata.remote = 0;

// Indicate that we want a ROLLBACK TRANSACTION for 1 of NewOrders.
RBtrans = 0;
if (uniform(1L, 5000L, &Info->seedbuf) <= 50)
{
    RBtrans = 1;
    Info->tdata.data = ROLLBACK;
}

// Generate a random District number between 1 and 10, inclusive.
neworder.did = uniform(1L, 10L, &Info->seedbuf);

// Generate a random Customer number less than 3000.
neworder.cid = NURand(1023, 1, 3000, CUSTC, &Info->seedbuf);

//Build Url to fill into NewOrder form.
// o "/tpcc.dll?PI=8FORMID=8d45TERMINID=8d45SYNCRID=8d"
// o 3
// o Info->TermID,
// o Info->SyncID,
// o "6DID=8d" where 8d is 1 - 10.
// o "8CID=8d" where 8d is 1 - 2999.
sprintf(theUrl,
        MAIN_FORM_FMT,
        NEW_ORDER_FORM_ID,
        Info->TermID,
        Info->SyncID);
sprintf(theUrl+strlen(theUrl), "6DID=8d", neworder.did);
sprintf(theUrl+strlen(theUrl), "8CID=8d", neworder.cid);

remoteflag = 0; // for remote orders
neworder.olremote = 0; // find total number of remote order-lines
whses = pSHM->End_WH;

// Generate a random number between 5 and 15, inclusive,
// as a loop counter.
neworder.nloop = uniform(5L, 15L, &Info->seedbuf);
for (i = 0; i < neworder.nloop; i++)
{
    neworder.item[i].olswid = Info->Warehouse;
    if (whses > 1 && (uniform(0.0, 100.0, &Info->seedbuf) < 1.0)) {
        while (neworder.item[i].olswid == Info->Warehouse)
        {
            neworder.item[i].olswid =
                (long) uniform((long) low_whse, (long)whses, &Info->seedbuf);
        }
        printf("**** Remote NEWORDER *** Local W_ID: %d, Remote W_ID:
            %d\n", Info->Warehouse, neworder.item[i].olswid);
        neworder.olremote++; // find total number of remote order-lines //
        Info->tdata.remote++;
        remoteflag = 1;
    }
    sprintf(theUrl+strlen(theUrl), "%SP402d=8d", i, neworder.item[i].olswid);
    Info->tdata.orders++;

    // If last iteration of loop and a RollBack transaction.
    if (neworder.nloop == i+1 && RBtrans)
        neworder.item[i].oliid = 999999;
    else
    {
        // Item value less than 100,000.
        neworder.item[i].oliid = NURand(8191, 1, 100000, ITEMCI, &Info->seedbuf);
    }
    sprintf(theUrl+strlen(theUrl), "%IID402d=8d", i, neworder.item[i].oliid);
    // Quantity is a random value between 1 and 10, inclusive.
    neworder.item[i].olquantity = uniform(1L, 10L, &Info->seedbuf);
    sprintf(theUrl+strlen(theUrl), "%Qy402d=8d", i, neworder.item[i].olquantity);
} // end of for loop

// If this is a remote order, ...
if (remoteflag == 1)
    neworder.oremote = 1;
else
    neworder.oremote = 0;

// Complete the URL by specifying remaining Supplier Warehouse #,
// Item Id and Quantity fields as null.
for (i=1; i++) {
    sprintf(theUrl+strlen(theUrl), "%SP402d=8IID402d=8Qy402d=8", i, i, i);
}
sprintf(theUrl+strlen(theUrl), "%CMD=Process");

// Fill in additional data
Info->tdata.trans_type = NEWORDER;
Info->tdata.wh = Info->Warehouse;
Info->tdata.w_d_id = Info->District;
Info->tdata.d_id = (char)neworder.did;
Info->tdata.c_id = (unsigned short)neworder.cid;

// Determine how long to sleep to meet minimum Keying time requirement and
// take a short nap.
// Mark End of Keying time and beginning of delivery response time.
// Determine how long to sleep to meet minimum keying time requirement
// and take a short nap.
// NEW ORDER KEYING TIME = 18000 milliseconds
// We are subtracting from NEW ORDER KEYING TIME the variable time
// that it took us to get from the last sleep to here.
unsigned int wait_time = NEW_ORDER_KEYING_TIME + t2 - pSHM->timestamp();
if (wait_time > 0)
    Sleep((unsigned int)wait_time);
pSHM->User[Info->UserNum].transaction_state = Transaction_State4;

// Log Key Time
// Determine the time at the beginning of the delivery response time.
// Determine from that the amount of time past since the prior sleep.
Info->tdata.trans_start_ms = pSHM->timestamp();
Info->tdata.key_ms = Info->tdata.trans_start_ms - t2;

sprintf(Request, GET_REQUEST, theUrl);
if (Info->socket.write(Request) < 0)
{
    perror("(Transactions.C) NewOrder() failure on write to socket due to: ");
    return 33;
}

// Iteratively read up to 4096 bytes from the socket until we find "</HTML>".
pSHM->User[Info->UserNum].transaction_state = Transaction_State5;
pSHM->User[Info->UserNum].transaction_bytes = 0;
totalBytes = 0;
memset(Result, 0, sizeof(Result));
while(totalBytes < sizeof(Result))
{
    if ((bytes = Info->socket.read(buf, sizeof(buf)-1)) < 0) return 43;
    pSHM->User[Info->UserNum].transaction_bytes = totalBytes + bytes;
    if ((totalBytes + bytes) > sizeof(Result)) return 53;
    memcpy(Result[totalBytes], buf, bytes);
    if (strstr(Result, "</HTML>")) break;
    totalBytes += bytes;
}
pSHM->User[Info->UserNum].transaction_state = Transaction_State6;

// Search the Result string for "TPC-C New Order".
if (strstr(Result, NEW_ORDER_EXPECT_STRING) == 0)
{
    cout << "(Transaction.C) NewOrder() received the following string: "
         << Result << endl;
    return 9;
}

// Transaction response time includes delay to display data on screen.
Sleep(pSHM->response[NEWORDER]*1000.0);
pSHM->User[Info->UserNum].transaction_state = Transaction_State7;

// Mark End of NewOrder response time, beginning of Think Time
Info->tdata.trans_finish_ms = pSHM->timestamp();
Info->tdata.trans_ms = Info->tdata.trans_finish_ms - Info->tdata.trans_start_ms;

if (pStr = strstr(Result, STATUS_ID_STRING))
{
    pStr += strlen(STATUS_ID_STRING);
    status = atoi(pStr);

    if (status > 0)
    {
        Info->tdata.status = status;
        if (pStr = strstr(Result, ORDER_NUMBER_STRING))
    }
}

```

```

    {
        pStr += strlen(ORDER_NUMBER_STRING);
        neworder.oid = atoi(pStr);
        if (pStr = strstr(Result, ROLLBACK_MSG_STRING)) {
            neworder.rollback = 1;
            if ( neworder.item[neworder.nloop-1].oliid == 999999 ) {
                }
            }
        }
    }
    else
    {
        neworder.oid = -1;
        neworder.invalid = 1;
        PSHM->transaction_done(&Info->tdata);
        return 22;
    }
}
// routine will work for TPCKIT version 3.002
else
{
    if (!(pStr = strstr(Result, ORDER_NUMBER_STRING)))
        neworder.oid = -1;
    else
    {
        pStr += strlen(ORDER_NUMBER_STRING);
        neworder.oid = atoi(pStr);
        if (pStr = strstr(Result, ROLLBACK_MSG_STRING)) {
            neworder.rollback = 1;
            if ( neworder.item[neworder.nloop-1].oliid == 999999 ) {
                }
            }
        }
    }
}
Info->tdata.o_id = (unsigned short)neworder.oid;

double think_time = neg_exp_4(PSHM->think[NWORDER], &Info->seedbuf);
if ((think_time*1000.0) > 0) {
    Sleep((unsigned)(think_time*1000.0));
}

return 1;
}

/*..... Order Status .....*/
int OrderStatus(UserInfo *Info)
{
    char Result[4096];
    char theUrl[1024];
    char Request[256];
    char * pStr;

    char buf[4097];
    buf[sizeof(buf)-1] = 0;

    ordatat_struct      ordatat;

    int      t1, t2;
    int      status;
    int      bytes, totalBytes;

    // Identify the transaction ID and state (for debug).
    PSHM->User[Info->UserNum].transaction_ID = OrderStatus_ID;
    PSHM->User[Info->UserNum].transaction_state = Transaction_State0;

    // Build Url to request Order Status Input screen

    // Locally mark the beginning of this routine by getting
    // the time relative to when RTE was started.
    cout << "(Transactions.C) OrderStatus() invoked by " << Info->UserNum << endl;
    t1 = PSHM->timestamp();

    // Create a buffer of
    // o "GET /s HTTP/2.0\r\nAccept: /* HTTP/2.0\r\nConnection: Keep-Alive\r\n\r\n"
    // o "/tpcc.d11?FORMID=4d5TERMINID=4d5SYNCD=4d5CMD=1s"
    // o 2
    // o Info->TermID,
    // o Info->SyncID,
    // o ".Order-Status.."
    // and send it on the socket.
    sprintf(theUrl,
        MAIN_MENU_FMT,
        MAIN_MENU_FORM_ID,
        Info->TermID,
        Info->SyncID,
        ".Order-Status..");
    sprintf(Request, GET_REQUEST, theUrl);
    if (Info->socket.write(Request < 0))
    {
        perror ("(Transactions.C) OrderStatus() failure on write to socket due to: ");
        return 34;
    }

    // Iteratively read up to 4096 bytes from the socket until we find "</HTML>".
    PSHM->User[Info->UserNum].transaction_state = Transaction_State1;
    PSHM->User[Info->UserNum].transaction_bytes = 0;
    totalBytes = 0;
    memset(Result, 0, sizeof(Result));
    while(totalBytes < sizeof(Result))
    {
        if ((bytes = Info->socket.read(buf, sizeof(buf)-1) < 0) return 44;
        PSHM->User[Info->UserNum].transaction_bytes = totalBytes + bytes;
        if ((totalBytes + bytes) > sizeof(Result)) return 54;
        memcpy(&Result[totalBytes], buf, bytes);
        if (strstr(Result, "</HTML>")) break;
        totalBytes += bytes;
    }
    PSHM->User[Info->UserNum].transaction_state = Transaction_State2;

    // If the Result buffer does not contain "TPC-C Order-Status", return.
    if (strstr(Result, ORDER_STATUS_EXPECT_STRING) == 0)
    {
        cout << "(Transaction.C) OrderStatus() received the following string:\n"
            << Result << endl;
        return 11;
    }

    // Sleep for the number of seconds specified for the ORDSTAT option.
    Sleep(PSHM->menu[ORDSTAT]*1000.0);
    PSHM->User[Info->UserNum].transaction_state = Transaction_State3;

    // After the socket read-wait and the sleep, locally mark the end of
    // Menu response time and the beginning of keying time.
    t2 = PSHM->timestamp();

    // Subtract the time that the last transaction was finished from
    // the time at the beginning of this routine to produce
    // the number of milliseconds of think time.
    Info->tdata.think_ms = t1 - Info->tdata.trans_finish_ms;

    // Copy the transaction data into the shared memory's output buffer (finished_trans),
    // but exclude initial transaction value and other non-transaction types.
    if ((Info->tdata.trans_type > 0) && (Info->tdata.trans_type < MAX_TRAN_TYPE))
        PSHM->transaction_done(&Info->tdata);

    // Calculate the time (in milliseconds) from the beginning of this routine
    // to (essentially) right now.
    Info->tdata.menu_ms = t2 - t1;

    // Re-initialize the "status".
    Info->tdata.status = 0;

    // Scan Result string for "NAME=\STATUSID\ VALUE=\""
    // and then point past it.
    // If the value at this location is >0,
    // use it as a status and copy the transaction data into
    // the shared memory's output buffer (finished_trans).
    if (pStr = strstr(Result, STATUS_ID_STRING))
    {
        pStr += strlen(STATUS_ID_STRING);
        if ((status = atoi(pStr)) > 0)
        {
            Info->tdata.status = status;
            PSHM->transaction_done(&Info->tdata);
        }
    }

    Info->tdata.trans_type = ORDSTAT;
    Info->tdata.data = 0;
    Info->tdata.d_id = 0;
    Info->tdata.orders = 0;
    Info->tdata.remote = 0;
    Info->tdata.o_id = 0;

    // Set District number per a random number between 1 and 10, inclusively.
    ordstat.did = uniform(1L, 10L, &Info->seedbuf);

    //Build Url to fill in OrderStatus form
    // o "/tpcc.d11?PI=*FORMID=4d5TERMINID=4d5SYNCD=4d5"
    // o 6
    // o Info->TermID,
    // o Info->SyncID,
    // o "4DID=4d" where d is 1 - 10.
    sprintf(theUrl,
        MAIN_FORM_FMT,
        ORDER_STATUS_FORM_ID,
        Info->TermID,
        Info->SyncID);
    sprintf(theUrl+strlen(theUrl), "%DID=%d", ordstat.did);

    // For 60% of the transactions, ...
    if (uniform(1L, 10L, &Info->seedbuf) <= 60)
    {
        // Glom onto a 3-part randomly generated customer last name and
        // update the buffer.
        char getnameBuf[20];
        getname(getnameBuf, &Info->seedbuf);
        strcpy(ordstat.clast, getnameBuf);
        sprintf(theUrl+strlen(theUrl), "%CID=*4CLT=%s", ordstat.clast);
        ordstat.byname = 1;
        ordstat.cid = 0;
        Info->tdata.data |= BY_NAME;
    }
    else // For the remaining 40% of transactions...
    {
        // Generate a Customer ID less than or equal to 3000 and
        // update the buffer.
        ordstat.cid = NURand(1023, 1, 3000, CUSTC, &Info->seedbuf);
    }
#ifdef AUDIT_TRANS
    if (dnum == OS_DNUM) ordstat.cid = AUDIT_C_ID;
#endif
    sprintf(theUrl+strlen(theUrl), "%CID=*4dCLT=*", ordstat.cid);
    ordstat.byname = 0;
    ordstat.clast[0] = (char) NULL;
}
strcat(theUrl, "%CMD=Process");

// Fill in additional data
Info->tdata.wh = Info->Warehouse;
Info->tdata.w_d_id = Info->District;
Info->tdata.c_id = (unsigned short)ordstat.cid;

// Determine how long to sleep to meet minimum Keying time requirement and
// take a short nap
unsigned int wait_time = ORDER_STATUS_KEYING_TIME + t2 - PSHM->timestamp();
if (wait_time > 0)
    Sleep((unsigned int)wait_time);
PSHM->User[Info->UserNum].transaction_state = Transaction_State4;

// Mark End of Keying time, beginning of OrderStatus response time.
Info->tdata.trans_start_ms = PSHM->timestamp();
Info->tdata.key_ms = Info->tdata.trans_start_ms - t2;

sprintf(Request, GET_REQUEST, theUrl);
if (Info->socket.write(Request) < 0)
{
    perror ("(Transactions.C) OrderStatus() failure on write to socket due to: ");
    return 35;
}

// Iteratively read up to 4096 bytes from the socket until we find "</HTML>".
PSHM->User[Info->UserNum].transaction_state = Transaction_State5;
PSHM->User[Info->UserNum].transaction_bytes = 0;
totalBytes = 0;
memset(Result, 0, sizeof(Result));
while(totalBytes < sizeof(Result))
{
    if ((bytes = Info->socket.read(buf, sizeof(buf)-1) < 0) return 45;
    PSHM->User[Info->UserNum].transaction_bytes = totalBytes + bytes;
    if ((totalBytes + bytes) > sizeof(Result)) return 55;
    memcpy(&Result[totalBytes], buf, bytes);
    if (strstr(Result, "</HTML>")) break;
    totalBytes += bytes;
}
PSHM->User[Info->UserNum].transaction_state = Transaction_State6;

// Search the Result string for "TPC-C Order-Status".
if (strstr(Result, ORDER_STATUS_EXPECT_STRING) == 0)
{
    cout << "(Transaction.C) OrderStatus() received the following string: "
        << Result << endl;
    return 13;
}

// Transaction response time includes delay to display data on screen.
Sleep((PSHM->response[ORDSTAT]*1000.0));
PSHM->User[Info->UserNum].transaction_state = Transaction_State7;

// Mark End of OrderStatus response time, beginning of Think Time
Info->tdata.trans_finish_ms = PSHM->timestamp();
Info->tdata.trans_ms = Info->tdata.trans_finish_ms - Info->tdata.trans_start_ms;

// Scan Result string for "NAME=\STATUSID\ VALUE=\""
// and then point past it.
// If the value at this location is >0,
// use it as a status and copy the transaction data into
// the shared memory's output buffer (finished_trans).
if (pStr = strstr(Result, STATUS_ID_STRING))
{
    pStr += strlen(STATUS_ID_STRING);
    if ((status = atoi(pStr)) > 0)
    {
        Info->tdata.status = status;
        PSHM->transaction_done(&Info->tdata);
    }
}

double think_time = neg_exp_4(PSHM->think[ORDSTAT], &Info->seedbuf);
if ((think_time*1000.0) > 0) {
    Sleep((unsigned)(think_time*1000.0));
}

```

```

return 1;
}

// For 60% of the transactions, ...
if (uniform(1L, 100L, &Info->seedbuf) <= 60)
{
    // Randomly generate a 3-part Customer name.
    char getnameBuf[20];
    getname(getnameBuf, &Info->seedbuf);
    strcpy(payment.clast, getnameBuf);
    payment.byname = 1;
    payment.cid = 0;
    Info->tdata.data |= BY_NAME;
}
else
{
    // For the remaining 40%, ...
    // Choose a Customer ID between 1 and 3000.
    payment.cid = NURand(1023, 1, 3000, CUSTC, &Info->seedbuf);
}
#ifdef AUDIT_TRANS
    if (dnum == PAY_DNUM)
        payment.cid = AUDIT_C_ID;
#endif
    payment.byname = 0;
    payment.clast[0] = (char) NULL;
}

// For 85% of the transactions OR
// if there are fewer than 2 warehouses,
// use the one warehouse in Info->Warehouse.
whses = pSHM->End WH;
// cout << "Total warehouses = " << whses << " hom whs = " << Info->Warehouse << endl;
if (whses < 2 || uniform(1L, 100L, &Info->seedbuf) <= 85)
{
    payment.cid = Info->Warehouse;
    payment.cidid = payment.did;
    payment.remote = 0;
}
else
{
    // For the remaining 15 % of transactions,
    // randomly choose a warehouse ID other than the one in Info->Warehouse.
    while (1)
    {
        payment.cidid = (long)uniform((long)low whses, (long)whses, &Info->seedbuf);
        if (payment.cidid != Info->Warehouse) break;
    }
    payment.remote = 1;
    payment.cidid = uniform(1L, 10L, &Info->seedbuf) // district 1 to 10
    Info->tdata.data |= REMOTE;
}

// cout << "seed = " << &Info->seedbuf << " rem dat = " << payment.cidid << endl;
// cout << "hom whs = " << Info->Warehouse << " rem whs = " << payment.cidid << endl;
// cout << "cid = " << payment.cid << " clast = " << payment.clast << endl;
}
if (payment.byname)
{
    sprintf(theUrl+strlen(theUrl), "%CID*%CWI*%d%CDI*%d%CLT*%s",
            payment.cidid, payment.cidid, payment.clast);
}
else
{
    sprintf(theUrl+strlen(theUrl), "%CID*%d%CWI*%d%CDI*%d%CLT*%s",
            payment.cid, payment.cidid, payment.cidid);
}

// Set a dollar amount between 1 and 5000, inclusive, and
// a cents amount between 0 and 99, inclusive.
dollars = uniform(1L, 5000L, &Info->seedbuf);
if (dollars == 5000)
    cents = 0;
else
    cents = uniform(0L, 99L, &Info->seedbuf);
payment.amount = ((double) dollars) + ((double) cents) / 100.0;
sprintf(theUrl+strlen(theUrl), "%$M*%d.%02d",
        dollars, cents);
strcat(theUrl, "%CMD=Process");

// Fill in additional data
Info->tdata.wh = Info->Warehouse;
Info->tdata.w_d_id = Info->District;
Info->tdata.d_id = (char)payment.cidid;
Info->tdata.c_id = (unsigned short)payment.cid;

// Determine how long to sleep to meet minimum Keying time requirement and
// take a short nap
unsigned int wait_time = PAYMENT_KEYING_TIME + t2 - pSHM->timestamp();
if (wait_time > 0)
    Sleep((unsigned int)wait_time);
pSHM->User[Info->UserNum].transaction_state = Transaction_State4;

// Mark End of Keying time, beginning of Payment response time
Info->tdata.trans_start_ms = pSHM->timestamp();
Info->tdata.key_ms = Info->tdata.trans_start_ms - t2;

sprintf(Request, GET_REQUEST, theUrl);
if ((Info->socket.write(Request)) < 0)
{
    perror ("(Transactions.C) Payment() failure on write to socket due to: ");
    return 37;
}

// Iteratively read up to 4096 bytes from the socket until we find "</HTML>".
pSHM->User[Info->UserNum].transaction_state = Transaction_State5;
pSHM->User[Info->UserNum].transaction_bytes = 0;
memset(Result, 0, sizeof(Result));
while(totalBytes < sizeof(Result))
{
    if ((bytes = Info->socket.read(buf, sizeof(buf)-1)) < 0) return 47;
    pSHM->User[Info->UserNum].transaction_bytes = totalBytes + bytes;
    if ((totalBytes + bytes) > sizeof(Result)) return 57;
    memcpy(&Result[totalBytes], buf, bytes);
    if (strstr(Result, "</HTML>")) break;
    totalBytes += bytes;
}
pSHM->User[Info->UserNum].transaction_state = Transaction_State6;

// If the Result string does not contain "TPC-C Payment", return.
if (strstr(Result, PAYMENT_EXPECT_STRING) == 0) return 17;

// Sleep for the number of seconds specified for the PAYMENT option.
Sleep(pSHM->response[PAYMENT]*1000.0);
pSHM->User[Info->UserNum].transaction_state = Transaction_State7;

// Mark End of Payment response time, beginning of Think Time
Info->tdata.trans_finish_ms = pSHM->timestamp();
Info->tdata.trans_ms = Info->tdata.trans_finish_ms - Info->tdata.trans_start_ms;

// Scan Result string for "NAME=\\"STATUSID\\" VALUE=\\"
// and then point past it.
// If the value at this location is >0,
// use it as a status and copy the transaction data into
// the shared memory's output buffer (finished_trans).
if (pStr = strstr(Result, STATUS_ID_STRING))
{
    pStr += strlen(STATUS_ID_STRING);
    if ((status = atoi(pStr)) > 0)
    {
        Info->tdata.status = status;
        pSHM->transaction_done(&Info->tdata);
    }
}

Info->tdata.trans_type = PAYMENT;
Info->tdata.data = 0;
Info->tdata.orders = 0;
Info->tdata.remote = 0;
Info->tdata.c_id = 0;
Info->tdata.o_id = 0;

// Set the District number per a random number between 1 and 10, inclusively.
payment.did = uniform(1L, 10L, &Info->seedbuf);

//Build Url to fill in Payment form.
// o "tpcc.dll?PI*%FORMID=%d&TERMD=%d&SYNCID=%d"
// o 4
// o Info->TermID (originally from AS/400 at login)
// o Info->SyncID (originally from AS/400 at login)
// o "%CID*%d" where %d is 1 - 10.
sprintf(theUrl,
        MAIN_FORM_FMT,
        PAYMENT_FORM_ID,
        Info->TermID,
        Info->SyncID);
sprintf(theUrl+strlen(theUrl), "%DID*%d", payment.did);

```

```

        pSHM->transaction_done(&Info->tdata);
    }
}

double think_time = neg_exp_4(pSHM->think(PAYMENT), &Info->seedbuf);
if ((think_time*1000.0) > 0)
    Sleep((unsigned)(think_time*1000.0));
return 1;
}

/***** Stock Level *****/
int StockLevel(UserInfo *Info)
{
    char Result[4096];
    char theUrl[1024];
    char Request[256];
    char* pStr;

    char buf[4097];
    buf[sizeof(buf)-1] = 0;

    stocklev_struct      stocklevel;

    int      bytes, totalBytes;
    int      t1, t2;
    int      status;

    // Identify the transaction ID and state (for debug).
    pSHM->User[Info->UserNum].transaction_ID = StockLevel_ID;
    pSHM->User[Info->UserNum].transaction_state = Transaction_State0;

    // Locally mark the beginning of this routine by getting
    // the time relative to when RTE was started.
    cout << "(Transactions.C) StockLevel() invoked by " << Info->UserNum << endl;
    t1 = pSHM->timestamp();

    // Create a buffer of
    // o "GET /s HTTP/2.0\r\nAccept: */*\ HTTP/2.0\r\nConnection: Keep-Alive\r\n\r\n"
    // o "/tpcc.d112FORMID=4&STERMID=4&SYNCID=4&CMD=4s"
    // o 2
    // o Info->TermID,
    // o Info->SyncID,
    // o ".Stock-Level.."
    // and send it on the socket.
    sprintf(theUrl,
        MAIN_MENU_FMT,
        MAIN_MENU_FORM_ID,
        Info->TermID,
        Info->SyncID,
        ".Stock-Level..");
    sprintf(Request, GET_REQUEST, theUrl);
    if ((Info->socket.write(Request)) < 0)
    {
        perror("(Transactions.C) StockLevel() failure on write to socket due to: ");
        return 38;
    }

    // Iteratively read up to 4096 bytes from the socket until we find "</HTML>".
    pSHM->User[Info->UserNum].transaction_state = Transaction_State1;
    pSHM->User[Info->UserNum].transaction_bytes = 0;
    totalBytes = 0;
    memset(Result, 0, sizeof(Result));
    while(totalBytes < sizeof(Result))
    {
        if ((bytes = Info->socket.read(buf, sizeof(buf)-1)) < 0) return 48;
        pSHM->User[Info->UserNum].transaction_bytes = totalBytes + bytes;
        if ((totalBytes + bytes) > sizeof(Result)) return 58;
        memcpy(&Result[totalBytes], buf, bytes);
        if (strstr(Result, "</HTML>")) break;
        totalBytes += bytes;
    }
    pSHM->User[Info->UserNum].transaction_state = Transaction_State2;

    // If the Result buffer does not contain "TPC-C Stock Level", return.
    if (strstr(Result, STOCK_LEVEL_EXPECT_STRING) == 0)
    {
        cout << "(Transaction.C) StockLevel() received the following string: "
            << Result << endl;
        return 19;
    }

    // Sleep for the number of seconds specified for the STOCKLEV option.
    Sleep(pSHM->menu[STOCKLEV]*1000.0);
    pSHM->User[Info->UserNum].transaction_state = Transaction_State3;

    // After the socket read-wait and the sleep, locally mark the end of
    // Menu response time and the beginning of keying time.
    t2 = pSHM->timestamp();

    // Subtract the time that the last transaction was finished from
    // the time at the beginning of this routine to produce
    // the number of milliseconds of think time.
    Info->tdata.think_ms = t1 - Info->tdata.trans_finish_ms;

    // Copy the transaction data into the shared memory's output buffer (finished_trans),
    // but exclude initial transaction value and other non-transaction types.
    if (Info->tdata.trans_type > 0 && Info->tdata.trans_type < MAX_TRAN_TYPE)
        pSHM->transaction_done(&Info->tdata);

    // Calculate the time (in milliseconds) from the beginning of this routine
    // to (essentially) right now.
    Info->tdata.menu_ms = t2 - t1;

    // Re-initialize the "status".
    Info->tdata.status = 0;

    // Scan Result string for "NAME=\"STATUSID\" VALUE=\""
    // and then point past it.
    // If the value at this location is >0,
    // use it as a status and copy the transaction data into
    // the shared memory's output buffer (finished_trans).
    if (pStr = strstr(Result, STATUS_ID_STRING))
    {
        pStr += strlen(STATUS_ID_STRING);
        if ((status = atoi(pStr)) > 0)
        {
            Info->tdata.status = status;
            pSHM->transaction_done(&Info->tdata);
        }
    }

    // Generate a random threshold value between 10 and 20, inclusively.
    stocklevel.threshold = uniform(10L, 20L, &Info->seedbuf);

    //Build Url to fill in Stock-Level form.
    // o "/tpcc.d112P1*4FORMID=4&STERMID=4&SYNCID=4d"
    // o 7
    // o Info->TermID (originally from AS/400 at login)
    // o Info->SyncID (originally from AS/400 at login)
    // o "4TT*4d" where 4d is 10 - 20.
    // o "4CMD=Process"
    sprintf(theUrl,
        MAIN_MENU_FMT,
        STOCK_LEVEL_FORM_ID,
        Info->TermID,
        Info->SyncID);
    sprintf(theUrl, &strcat(theUrl, "4TT*4d", stocklevel.threshold);
    strcat(theUrl, "4CMD=Process");

    // Fill in additional data
    Info->tdata.trans_type = STOCKLEV;
    Info->tdata.data = 0;
    Info->tdata.orders = 0;
    Info->tdata.remote = 0;
    Info->tdata.c_id = 0;
    Info->tdata.d_id = 0;
    Info->tdata.wR = Info->Warehouse;
    Info->tdata.d_id = Info->District;

    // Determine how long to sleep to meet minimum Keying time requirement and
    // take a short nap
    unsigned int wait_time = STOCK_LEVEL_KEYING_TIME + t2 - pSHM->timestamp();
    if (wait_time > 0)
        Sleep((unsigned int)wait_time);
    pSHM->User[Info->UserNum].transaction_state = Transaction_State4;

    // Mark End of Keying time, beginning of Stock-Level response time
    Info->tdata.trans_start_ms = pSHM->timestamp();
    Info->tdata.key_ms = Info->tdata.trans_start_ms - t2;

    // Send the Stock-Level form on the socket.
    sprintf(Request, GET_REQUEST, theUrl);
    if ((Info->socket.write(Request)) < 0)
    {
        perror("(Transactions.C) StockLevel() failure on write to socket due to: ");
        return 39;
    }

    // Iteratively read up to 4096 bytes from the socket until we find "</HTML>".
    pSHM->User[Info->UserNum].transaction_state = Transaction_State5;
    pSHM->User[Info->UserNum].transaction_bytes = 0;
    totalBytes = 0;
    memset(Result, 0, sizeof(Result));
    while(totalBytes < sizeof(Result))
    {
        if ((bytes = Info->socket.read(buf, sizeof(buf)-1)) < 0) return 49;
        pSHM->User[Info->UserNum].transaction_bytes = totalBytes + bytes;
        if ((totalBytes + bytes) > sizeof(Result)) return 59;
        memcpy(&Result[totalBytes], buf, bytes);
        if (strstr(Result, "</HTML>")) break;
        totalBytes += bytes;
    }
    pSHM->User[Info->UserNum].transaction_state = Transaction_State6;

    // If the Result buffer does not contain "TPC-C Stock Level", return.
    if (strstr(Result, STOCK_LEVEL_EXPECT_STRING) == 0)
    {
        cout << "(Transaction.C) StockLevel() received the following string: "
            << Result << endl;
        return 21;
    }

    // Transaction response time includes delay to display data on screen.
    Sleep(pSHM->response[STOCKLEV]*1000.0);
    pSHM->User[Info->UserNum].transaction_state = Transaction_State7;

    // Mark End of Stock-Level response time, beginning of Think Time
    Info->tdata.trans_finish_ms = pSHM->timestamp();
    Info->tdata.trans_ms = Info->tdata.trans_finish_ms - Info->tdata.trans_start_ms;

    // Scan Result string for "NAME=\"STATUSID\" VALUE=\""
    // and then point past it.
    // If the value at this location is >0,
    // use it as a status and copy the transaction data into
    // the shared memory's output buffer (finished_trans).
    if (pStr = strstr(Result, STATUS_ID_STRING))
    {
        pStr += strlen(STATUS_ID_STRING);
        if ((status = atoi(pStr)) > 0)
        {
            Info->tdata.status = status;
            pSHM->transaction_done(&Info->tdata);
        }
    }

    double think_time = neg_exp_4(pSHM->think(STOCKLEV), &Info->seedbuf);
    if ((think_time*1000.0) > 0)
        Sleep((unsigned)(think_time*1000.0));
    return 1;
}
}

```



# Appendix F. 180-Day DASD Requirements

The appendix documents the information required to calculate the 180-Day DASD requirements for the TPC Benchmark C measurements on the IBM ~ iSeries 400 Model 840-2420 system. Also included is the calculation used to determine the amount of DASD required for the 8 hours of journal space.

Section “IBM ~ iSeries 400 Model 840-2420 -- 180-Day DASD Requirements” shows the values in the calculation of the 180-Day Space. Also used were the formulas provided in Clause 4.2.3 of the TPC Benchmark C Standard Specification. These formulas are:

$$\begin{aligned}
 \text{Daily-Growth} &= \{\text{dynamic-space}/(\text{W}*62.5)\} * \text{tpmC} \\
 \text{Daily-Spread} &= \text{MAX}(0, \text{Free-Space} - 1.5*\text{Daily-Growth}) \\
 \text{180-Day-Space} &= \text{Static-Space} + 180*(\text{Daily-Growth}+\text{Daily-Spread})
 \end{aligned}$$

## F.1 IBM ^ iSeries 400 Model 840-2420 -- 180-Day DASD Requirements

Static Files	Number of Warehouses	12200	tpmC	152346	
File name	Customer	District	Item	New Order	Stock Warehouse
Number of Records	366000000	122000	100000	39600000	122000000 12100
Data Space	244490420224	12599296	9445376	1207877632	374541230080 1196032
Index Space	11050446848	2105344	1843200	2004901888	38053912576 229376
Add'l Index	25718587392			12288	
Add'l Index	16384				
<b>Dynamic Files</b>					
File name	History	Orderline	Orders		
Initial Records	366000000	3660140702	366000000		
Size Per Record	49.0016641166	75.7402476648	26.0009640801		
Initial Data Space	17934609067	277219963258	9516352853		
Initial Index Space	24576	98620739584	12024115200		
Add'l Index		16384	12172947456		
Add'l Index					
Configured Space	21521530880	332651167744	11419623424		
Free Space	3586921813	55431204486	1903270571		
Static Space	854767383757				
Dynamic Space	304670925178				
Free Space	60921396870				
Daily Growth	60872651498				
Daily Spread	0				
180 Day Space	11811844653371				
System Software	1941358080				
SubTotal	11813786011451				
Journal Space	1055366564646				
Total DASD Required	12869152576097				
DASD Priced	12990690623488				
Difference	121538047391				

## **F.2 IBM ^ iSeries 400 Model 840-2420 -- Journal DASD Requirements**

To determine the amount of DASD required for the 8 hours of journal, a series of tests were run for an extended period. Through these tests it was determined that an average of 6,927,432 bytes of DASD are required per tpmC.

# Appendix G. Third Party Quotes



THE ECOMMERCE TRANSACTION PLATFORM

September 18, 2000

Mr. Vinay Kulkarni  
Bldg. 006-2  
3605 Highway 52 North  
Rochester, MN 55901  
Phone: 507 253-2077  
FAX: 507 253-7743

Dear Mr. Kulkarni:

Per your request I am enclosing the pricing information regarding TUXEDO that you requested. This pricing applies to Tuxedo 6.1, 6.2, 6.3, 6.4, 6.5 and 7.1. Please note that Tuxedo 7.1 is our most recent version of Tuxedo but that all 6.x releases are generally available. Core functionality services pricing is appropriate for your activities. As per the table below, server systems are classified in one of 5 tiers based on CPU type and capacity. *This quote is valid for 90 days from the date of issue of this letter.*

## ***Tuxedo Core Functionality Services (CFS) Program Product Pricing and Description***

TUX-CFS provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations.

TUX-CFS prices are server only and are based on the overall performance characteristics of the server and uses the same five-tier computer classification as TUXEDO 6.x and 7.1. Prices range from \$3,000 for Tier 1 to \$250,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS at the user site must have a TUXEDO license installed and pay the appropriate per server license fees.

Very Truly Yours,

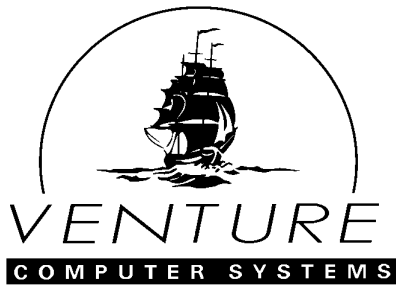
A handwritten signature in black ink that reads "Robert J. Buringer". The signature is written in a cursive, flowing style.

Robert J. Gieringer  
Worldwide Pricing Manager

**BEA Tux/CFS Unlimited User License Fees Per Server**

Unlimited User License fees per server	Number of Users	Dollar Amount	Maintenance (5 x 8) per year	Maintenance (7 x 24) per year
Tier 1 -- PC Servers with 1 or 2 CPUs, entry level RISC Uni-processor workstations and servers	Unlimited	\$3,000.00	\$480.00	\$690.00
Tier 2 - PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations with up to 2 CPUs	Unlimited	\$12,000.00	\$1,920.00	\$2,760.00
Tier 3 - Midrange Multiprocessors, up to 8 CPUs per system capacity	Unlimited	\$30,000.00	\$4,800.00	\$6,900.00
Tier 4 - Large (more than 8, less than 32 CPUs)	Unlimited	\$100,000.00	\$16,000.00	\$23,000.00
Tier 5 - Massively Parallel Systems, > 32 processors	Unlimited	\$250,000.00	\$40,000.00	\$57,500.00

	<i>Tier 1</i>	<i>Tier 2</i>	<i>Tier 3</i>	<i>Tier 3</i>	<i>Tier 4</i>	<i>Tier 5</i>
<i>Platform</i>	<i>Class 2</i>	<i>Class 3</i>	<i>Class 4</i>	<i>Class 5</i>	<i>Class 6</i>	<i>Class 7</i>
<i>IBM AS400</i>	<i>Model 170 (All except 2388)</i>  <i>Model 270 (1 way – 2248,2250, 2252)</i>	Models 20s, 30s, 40s, 200, 300, 400, 600, S10, 150 170 - 2388 150 all models  <i>Model 270 (2 Way)</i>	Models 310, 320, 500, 510, 520, 620, S20, 50s Model 530, 640, S30, S40, 53s 720 all models	730 all models 740-2069 s40-sb1-2 310 s40-sb1-2 311	Model 650 740-2070	



September 28, 2000

IBM Corporation  
Attn. Vinay Kulkarni  
Systems Performance I  
Rochester, MN 55901-7829

Dear Vinay:

Thank you for giving Venture Computer Systems the opportunity to bid for IBM's switch needs. Here are the prices we discussed:

Quantity	Description	Price	Extended
22,315	10BT HUB 8 RJ45 ENET MINI 1 UPLINK EXT POWER CREAM	\$28.55	\$637,093.25
20	3COM SUPERSTACK II SWITCH 3300 XM 24 PORT 10/100	\$975.24	\$19,504.80
TOTAL			\$656,598.05

**Bid Notes:**

- Hub Manufacturer: Hawking Technologies - [www.hawkingtech.com](http://www.hawkingtech.com)
- Price Quote Good for 90 Days
- Quote Number 30045

If you have any questions or comments, call me at (507) 252-5944 or E-Mail at [eckhoff@venturecs.com](mailto:eckhoff@venturecs.com).

Sincerely,

Jon Eckhoff  
Business Consultant  
Venture Computer Systems

# Appendix H. Auditor Letter



Benchmark Sponsor: Douglas D. Jans  
Advisory Software Engineer  
IBM Corp Dept HP4A:  
3605 Highway 52 North  
Rochester, MN 55901

October 3, 2000

I verified the TPC Benchmark™ C performance of the following Client Server configuration:

Platform: **IBM eServer iSeries 400 Model 840-2420 c/s**  
Operating system: **OS/400 Version 4 Release 5**  
Database Manager: **DB2 for AS/400 Version 4 Release 5**  
Transaction Manager: **BEA TUXEDO 6.4**

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
<b>Server: IBM eServer iSeries 400 Model 840-2420</b>				
24 x Power PC 2420 (500 MHz)	128 GB Main 8 MB cache/cpu	769 x 17.6 GB	0.5 Seconds	<b>152,346.25</b>
Sixteen (16) Clients: IBM eServer iSeries 400 Model 270-2252 (Specification for each)				
1 x Power PC 2252 (450 MHz)	8 GB Main 2 MB cache/cpu	6 x 8.6 GB	n/a	n/a

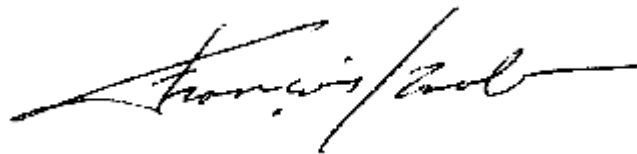
In my opinion, these performance results were produced in compliance with the TPC's requirements for the benchmark. The following verification items were given special attention:

- The database records were the proper size
- The database was properly scaled and populated
- The required ACID properties were met
- The transactions were correctly implemented
- Input data was generated according to the specified percentages
- The transaction cycle times included the required keying and think times
- The reported response times were correctly measured.
- All 90% response times were under the specified maximums
- At least 90% of all delivery transactions met the 80 Second completion time limit
- The reported measurement interval was 20 minutes (1200 seconds)
- The reported measurement interval was representative of steady state conditions
- At least five synch-points were taken during the reported measurement interval
- The repeatability of the measured performance was verified
- The 180 day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

None.

Respectfully Yours,

A handwritten signature in black ink, appearing to read "François Raab". The signature is fluid and cursive, with a long horizontal stroke extending to the right.

François Raab  
President