US 20050165668A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2005/0165668 A1**

Hinkle (43) **Pub. Date: Jul. 28, 2005**

(54) **MULTI-PROCESSING FINANCIAL TRANSACTION PROCESSING SYSTEM**

(76) Inventor: **William H. Hinkle**, Denver, CO (US)

Correspondence Address:
Merchant & Gould P.C.
Attention: George C. Lewis
P.O. Box 2903
Minneapolis, MN 55402-0903 (US)

(21) Appl. No.: **10/928,463**

(22) Filed: **Aug. 26, 2004**

Related U.S. Application Data

(63) Continuation of application No. 10/085,596, filed on Feb. 26, 2002, now Pat. No. 6,904,411, which is a continuation of application No. 09/181,698, filed on Oct. 28, 1998, now Pat. No. 6,442,533.

(60) Provisional application No. 60/063,714, filed on Oct. 29, 1997.

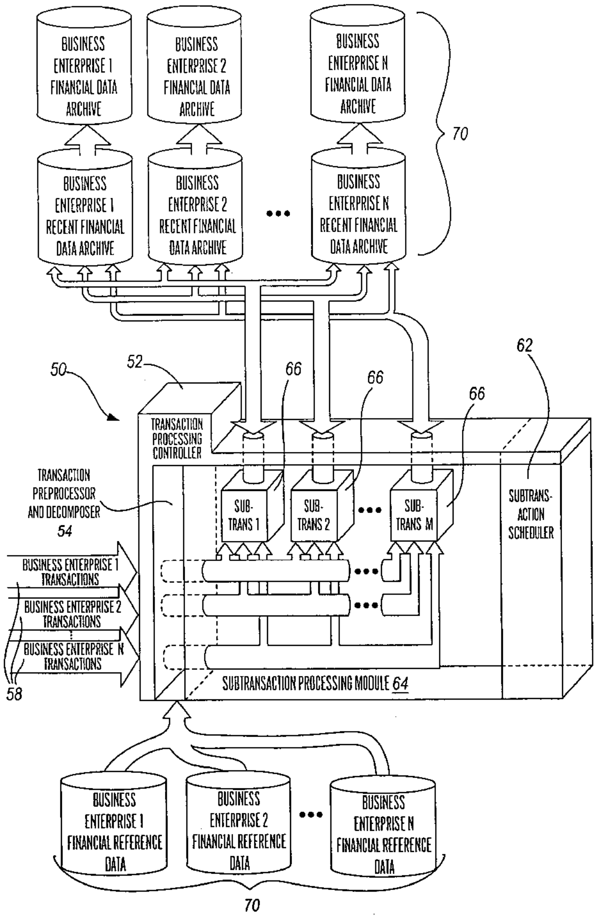
Publication Classification

(51) **Int. Cl.^{7}** **G06F 17/60**

(52) **U.S. Cl.** **705/37**

(57) **ABSTRACT**

A financial transaction processing system is disclosed, wherein substantial processing efficiencies are provided with, additionally, a substantial decrease in the size of the executable code. Each transaction processed by the transaction processing system is described by a transaction data descriptor that includes a series of subtransaction data descriptions of actions that can be performed independently of one another. Thus, complex transaction processing logic is substantially removed from the executable code, and instead such transaction data descriptors are processed interpretatively. Moreover, the independence of the subtransactions allows the subtransactions of a transaction to be processed in parallel when performed on a multiprocessor computer. Additionally, the transaction processing system provides account balancing enhancements in that there are control columns in various data tables that are automatically updated during transaction processing so that by comparing control column totals, an indication of the integrity of current financial records is provided. Additionally, the transaction processing system provides full auditability in that any changes to financial data can be traced for any effective period of time into the past so that auditors can periodically perform a full audit of the financial transaction data retained by the transaction processing system.


BUSINESS ENTERPRISE 1 FINANCIAL DATA ARCHIVE
BUSINESS ENTERPRISE 2 FINANCIAL DATA ARCHIVE
BUSINESS ENTERPRISE N FINANCIAL DATA ARCHIVE
70
BUSINESS ENTERPRISE 1 RECENT FINANCIAL DATA ARCHIVE
BUSINESS ENTERPRISE 2 RECENT FINANCIAL DATA ARCHIVE
•••
BUSINESS ENTERPRISE N RECENT FINANCIAL DATA ARCHIVE
62
50
52
66
66
66
TRANSACTION PROCESSING CONTROLLER
TRANSACTION PREPROCESSOR AND DECOMPOSER
54
SUB-TRANS 1
SUB-TRANS 2
•••
SUB-TRANS M
SUBTRANS-ACTION SCHEDULER
BUSINESS ENTERPRISE 1 TRANSACTIONS
BUSINESS ENTERPRISE 2 TRANSACTIONS
BUSINESS ENTERPRISE N TRANSACTIONS
58
SUBTRANSACTION PROCESSING MODULE 64
BUSINESS ENTERPRISE 1 FINANCIAL REFERENCE DATA
BUSINESS ENTERPRISE 2 FINANCIAL REFERENCE DATA
•••
BUSINESS ENTERPRISE N FINANCIAL REFERENCE DATA
70

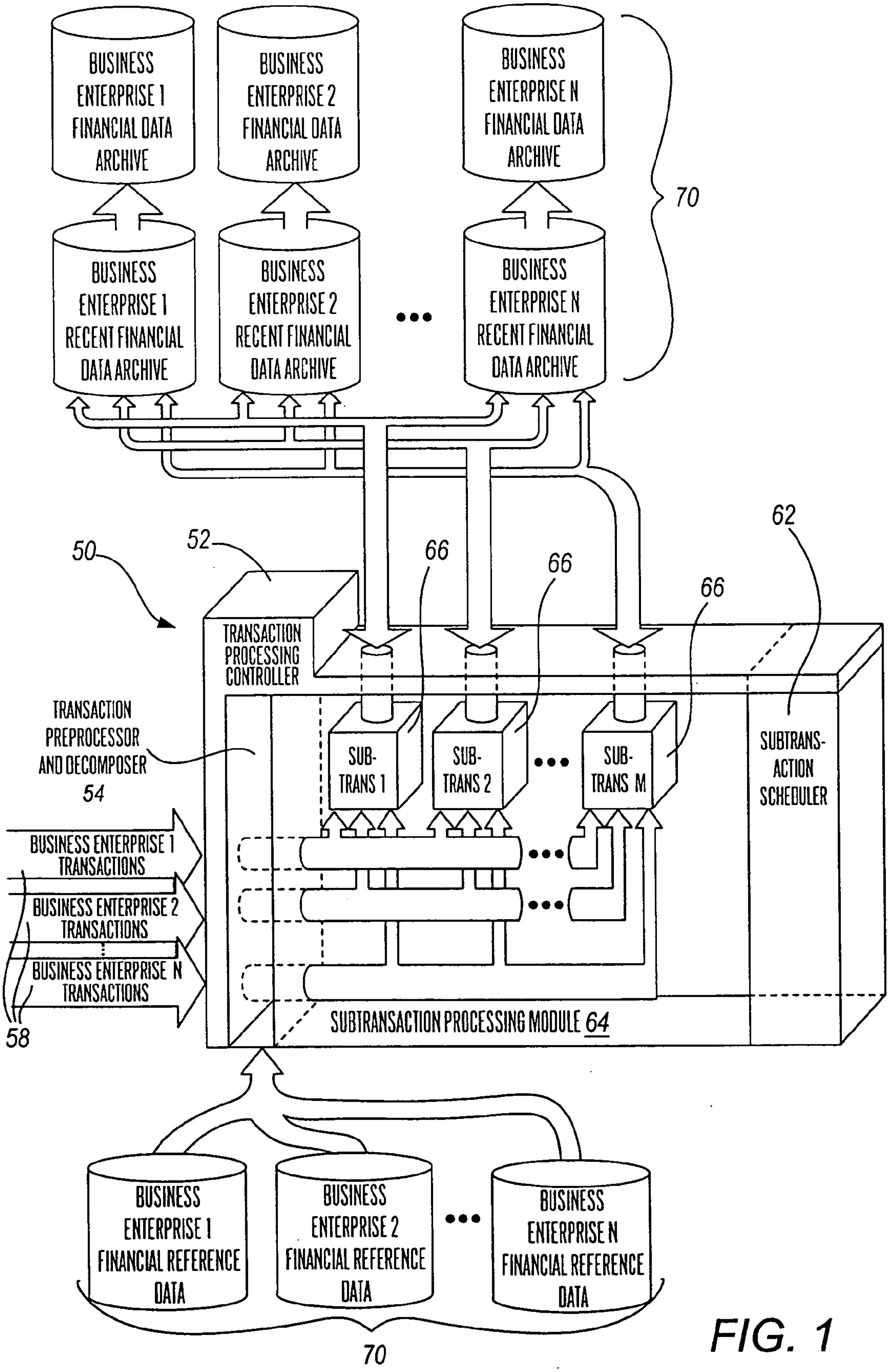

BUSINESS ENTERPRISE 1 FINANCIAL DATA ARCHIVE
BUSINESS ENTERPRISE 2 FINANCIAL DATA ARCHIVE
BUSINESS ENTERPRISE N FINANCIAL DATA ARCHIVE
70
BUSINESS ENTERPRISE 1 RECENT FINANCIAL DATA ARCHIVE
BUSINESS ENTERPRISE 2 RECENT FINANCIAL DATA ARCHIVE
•••
BUSINESS ENTERPRISE N RECENT FINANCIAL DATA ARCHIVE
62
50
52
66
66
66
TRANSACTION PROCESSING CONTROLLER
TRANSACTION PREPROCESSOR AND DECOMPOSER
54
SUB-TRANS 1
SUB-TRANS 2
•••
SUB-TRANS M
SUBTRANS-ACTION SCHEDULER
BUSINESS ENTERPRISE 1 TRANSACTIONS
BUSINESS ENTERPRISE 2 TRANSACTIONS
BUSINESS ENTERPRISE N TRANSACTIONS
58
SUBTRANSACTION PROCESSING MODULE 64
BUSINESS ENTERPRISE 1 FINANCIAL REFERENCE DATA
BUSINESS ENTERPRISE 2 FINANCIAL REFERENCE DATA
•••
BUSINESS ENTERPRISE N FINANCIAL REFERENCE DATA
70

FIG. 1

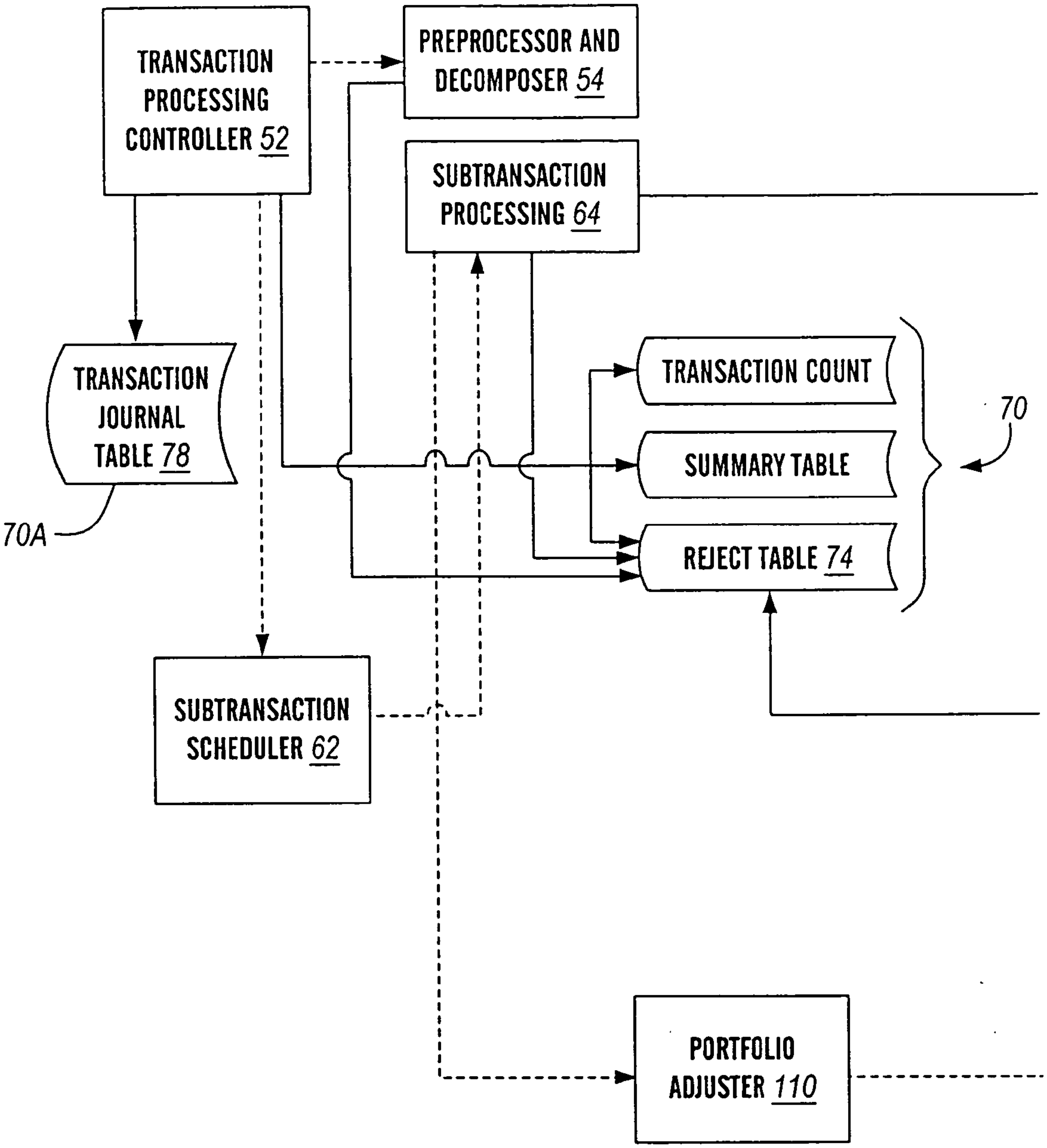
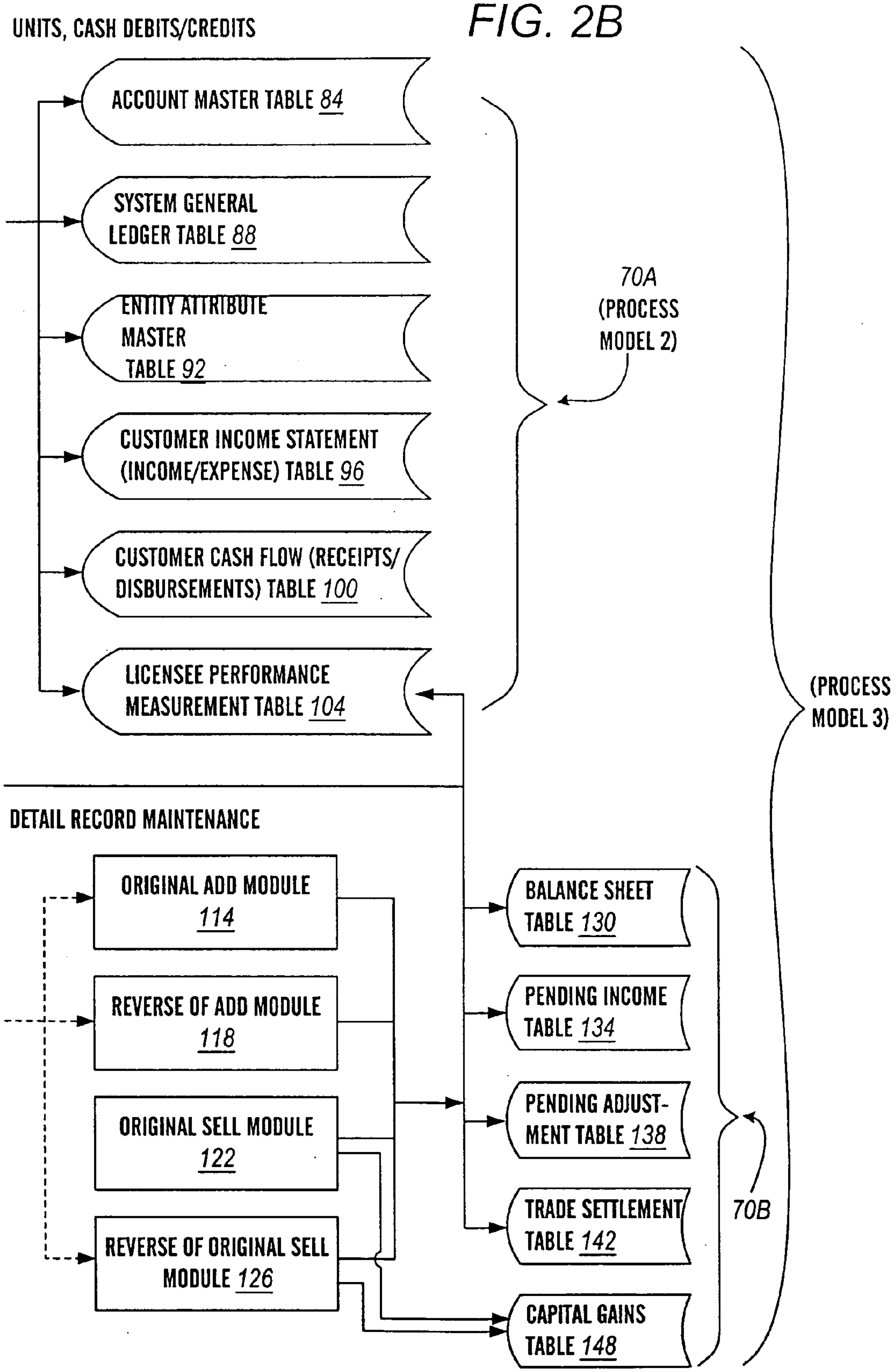

TRANSACTION PROCESSING CONTROLLER 52
PREPROCESSOR AND DECOMPOSER 54
SUBTRANSACTION PROCESSING 64
TRANSACTION JOURNAL TABLE 78
70A
TRANSACTION COUNT
SUMMARY TABLE
REJECT TABLE 74
70
SUBTRANSACTION SCHEDULER 62
PORTFOLIO ADJUSTER 110

FIG. 2A

FIG. 2B


UNITS, CASH DEBITS/CREDITS
ACCOUNT MASTER TABLE 84
SYSTEM GENERAL LEDGER TABLE 88
ENTITY ATTRIBUTE MASTER TABLE 92
CUSTOMER INCOME STATEMENT (INCOME/EXPENSE) TABLE 96
CUSTOMER CASH FLOW (RECEIPTS/ DISBURSEMENTS) TABLE 100
LICENSEE PERFORMANCE MEASUREMENT TABLE 104
70A (PROCESS MODEL 2)
(PROCESS MODEL 3)
DETAIL RECORD MAINTENANCE
ORIGINAL ADD MODULE 114
REVERSE OF ADD MODULE 118
ORIGINAL SELL MODULE 122
REVERSE OF ORIGINAL SELL MODULE 126
BALANCE SHEET TABLE 130
PENDING INCOME TABLE 134
PENDING ADJUST-MENT TABLE 138
TRADE SETTLEMENT TABLE 142
CAPITAL GAINS TABLE 148
70B

DATA FLOW INTO WORKING STORAGE DURING EXECUTION OF TRANSACTION
"PREPROCESSOR & DECOMPOSER 54"

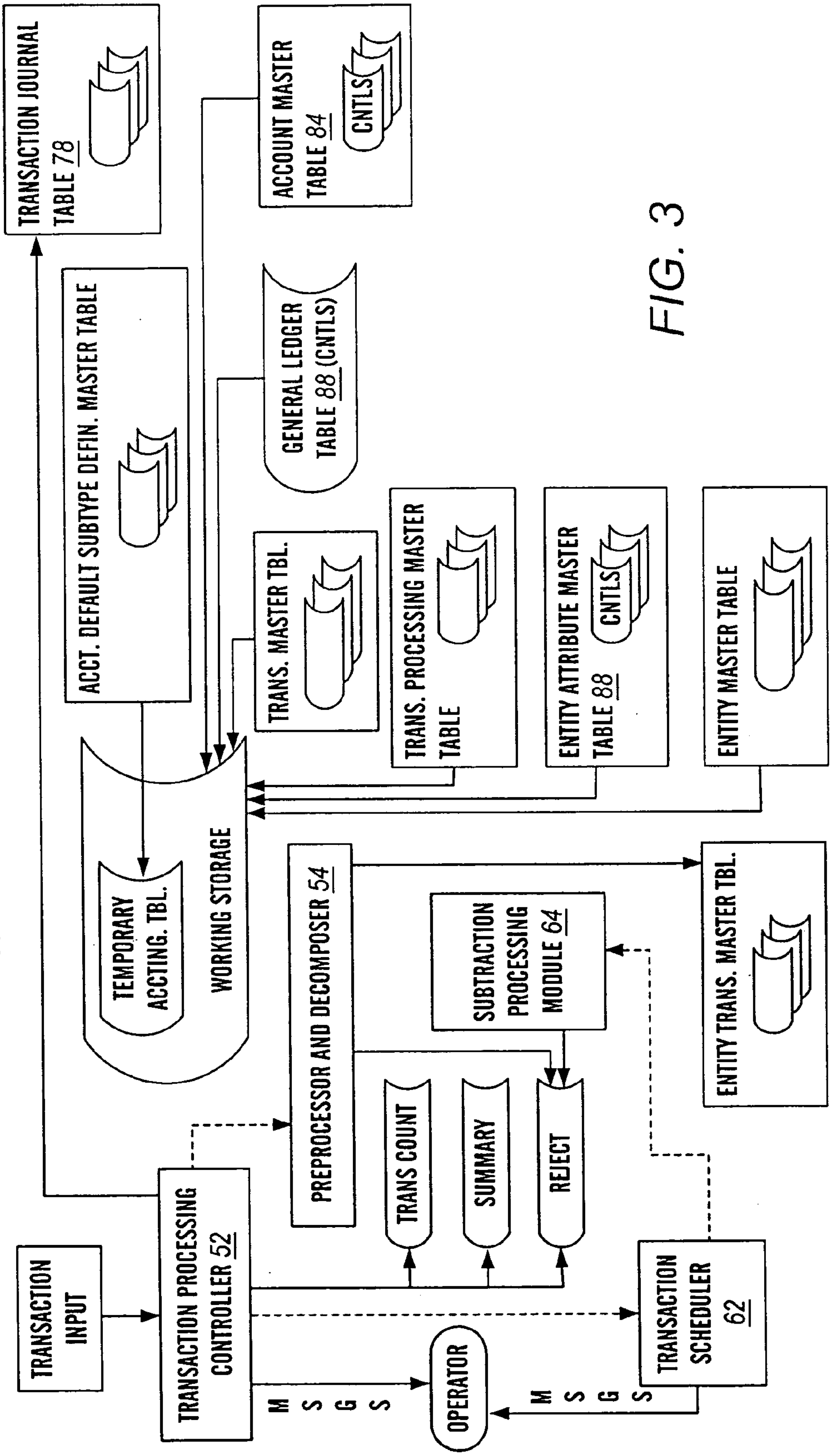
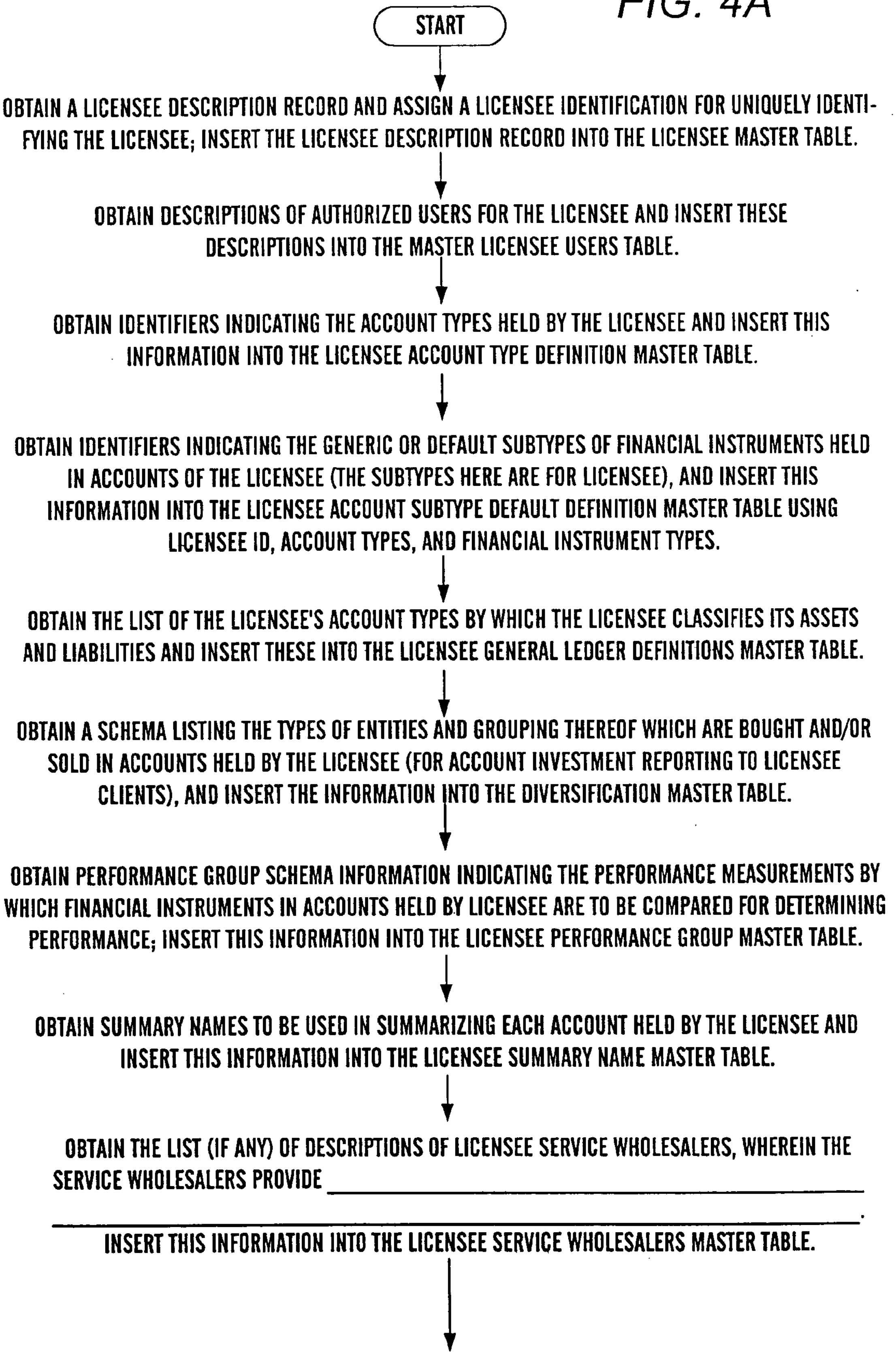

TRANSACTION JOURNAL TABLE 78
ACCOUNT MASTER TABLE 84
CNTLS
ACCT. DEFAULT SUBTYPE DEFIN. MASTER TABLE
GENERAL LEDGER TABLE 88 (CNTLS)
TRANS. MASTER TBL.
TRANS. PROCESSING MASTER TABLE
ENTITY ATTRIBUTE MASTER TABLE 88
CNTLS
ENTITY MASTER TABLE
TEMPORARY ACCTING. TBL.
WORKING STORAGE
PREPROCESSOR AND DECOMPOSER 54
SUBTRACTION PROCESSING MODULE 64
ENTITY TRANS. MASTER TBL.
TRANSACTION INPUT
TRANSACTION PROCESSING CONTROLLER 52
TRANS COUNT
SUMMARY
REJECT
TRANSACTION SCHEDULER 62
MSGS
OPERATOR
MSGS

FIG. 3

FIG. 4A



OBTAIN A LICENSEE DESCRIPTION RECORD AND ASSIGN A LICENSEE IDENTIFICATION FOR UNIQUELY IDENTIFYING THE LICENSEE; INSERT THE LICENSEE DESCRIPTION RECORD INTO THE LICENSEE MASTER TABLE.

OBTAIN DESCRIPTIONS OF AUTHORIZED USERS FOR THE LICENSEE AND INSERT THESE DESCRIPTIONS INTO THE MASTER LICENSEE USERS TABLE.

OBTAIN IDENTIFIERS INDICATING THE ACCOUNT TYPES HELD BY THE LICENSEE AND INSERT THIS INFORMATION INTO THE LICENSEE ACCOUNT TYPE DEFINITION MASTER TABLE.

OBTAIN IDENTIFIERS INDICATING THE GENERIC OR DEFAULT SUBTYPES OF FINANCIAL INSTRUMENTS HELD IN ACCOUNTS OF THE LICENSEE (THE SUBTYPES HERE ARE FOR LICENSEE), AND INSERT THIS INFORMATION INTO THE LICENSEE ACCOUNT SUBTYPE DEFAULT DEFINITION MASTER TABLE USING LICENSEE ID, ACCOUNT TYPES, AND FINANCIAL INSTRUMENT TYPES.

OBTAIN THE LIST OF THE LICENSEE'S ACCOUNT TYPES BY WHICH THE LICENSEE CLASSIFIES ITS ASSETS AND LIABILITIES AND INSERT THESE INTO THE LICENSEE GENERAL LEDGER DEFINITIONS MASTER TABLE.

OBTAIN A SCHEMA LISTING THE TYPES OF ENTITIES AND GROUPING THEREOF WHICH ARE BOUGHT AND/OR SOLD IN ACCOUNTS HELD BY THE LICENSEE (FOR ACCOUNT INVESTMENT REPORTING TO LICENSEE CLIENTS), AND INSERT THE INFORMATION INTO THE DIVERSIFICATION MASTER TABLE.

OBTAIN PERFORMANCE GROUP SCHEMA INFORMATION INDICATING THE PERFORMANCE MEASUREMENTS BY WHICH FINANCIAL INSTRUMENTS IN ACCOUNTS HELD BY LICENSEE ARE TO BE COMPARED FOR DETERMINING PERFORMANCE; INSERT THIS INFORMATION INTO THE LICENSEE PERFORMANCE GROUP MASTER TABLE.

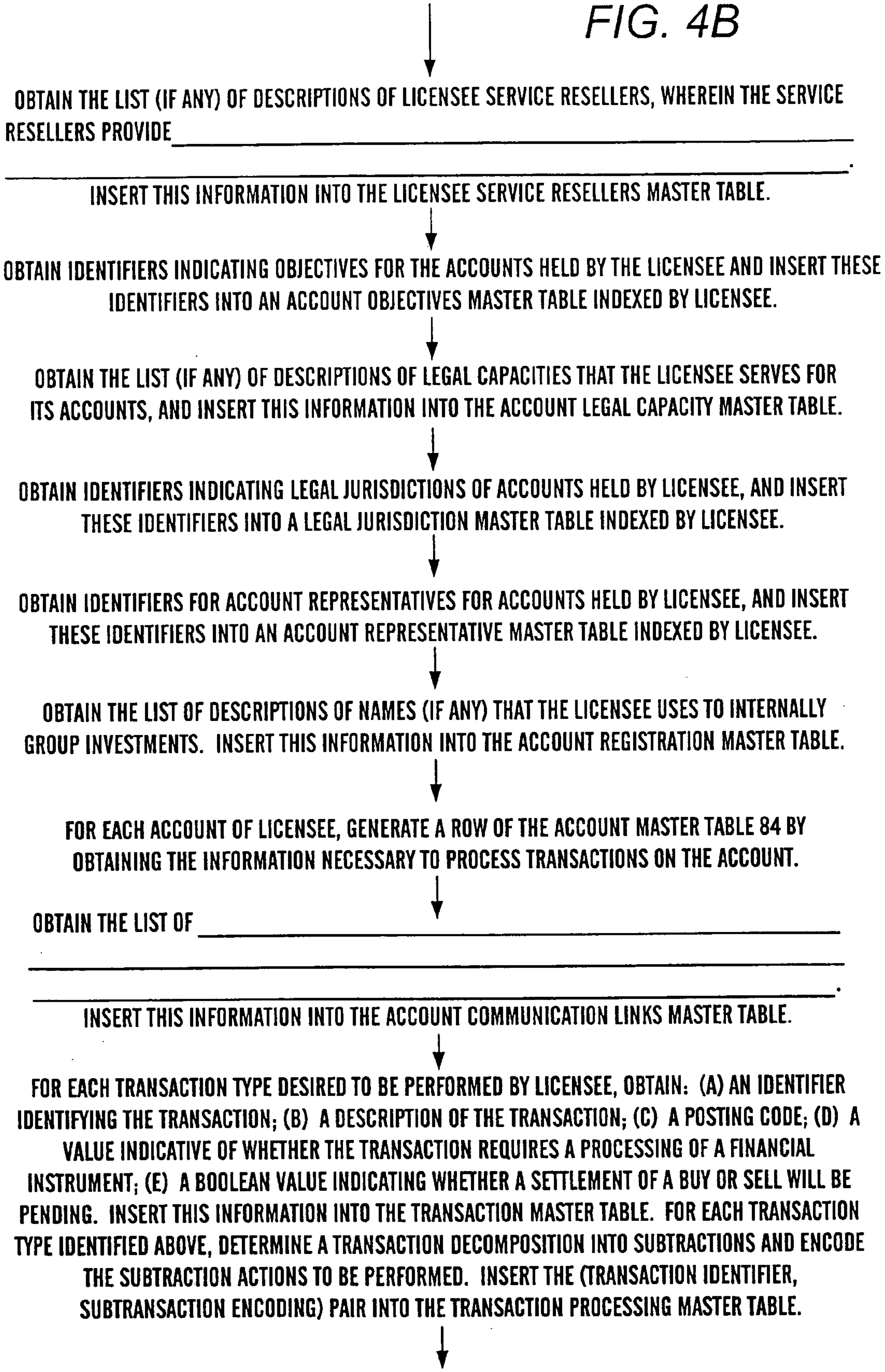
OBTAIN SUMMARY NAMES TO BE USED IN SUMMARIZING EACH ACCOUNT HELD BY THE LICENSEE AND INSERT THIS INFORMATION INTO THE LICENSEE SUMMARY NAME MASTER TABLE.

OBTAIN THE LIST (IF ANY) OF DESCRIPTIONS OF LICENSEE SERVICE WHOLESALERS, WHEREIN THE SERVICE WHOLESALERS PROVIDE ______________________________

____.

INSERT THIS INFORMATION INTO THE LICENSEE SERVICE WHOLESALERS MASTER TABLE.

FIG. 4B



OBTAIN THE LIST (IF ANY) OF DESCRIPTIONS OF LICENSEE SERVICE RESELLERS, WHEREIN THE SERVICE RESELLERS PROVIDE ____ ____. INSERT THIS INFORMATION INTO THE LICENSEE SERVICE RESELLERS MASTER TABLE.

OBTAIN IDENTIFIERS INDICATING OBJECTIVES FOR THE ACCOUNTS HELD BY THE LICENSEE AND INSERT THESE IDENTIFIERS INTO AN ACCOUNT OBJECTIVES MASTER TABLE INDEXED BY LICENSEE.

OBTAIN THE LIST (IF ANY) OF DESCRIPTIONS OF LEGAL CAPACITIES THAT THE LICENSEE SERVES FOR ITS ACCOUNTS, AND INSERT THIS INFORMATION INTO THE ACCOUNT LEGAL CAPACITY MASTER TABLE.

OBTAIN IDENTIFIERS INDICATING LEGAL JURISDICTIONS OF ACCOUNTS HELD BY LICENSEE, AND INSERT THESE IDENTIFIERS INTO A LEGAL JURISDICTION MASTER TABLE INDEXED BY LICENSEE.

OBTAIN IDENTIFIERS FOR ACCOUNT REPRESENTATIVES FOR ACCOUNTS HELD BY LICENSEE, AND INSERT THESE IDENTIFIERS INTO AN ACCOUNT REPRESENTATIVE MASTER TABLE INDEXED BY LICENSEE.

OBTAIN THE LIST OF DESCRIPTIONS OF NAMES (IF ANY) THAT THE LICENSEE USES TO INTERNALLY GROUP INVESTMENTS. INSERT THIS INFORMATION INTO THE ACCOUNT REGISTRATION MASTER TABLE.

FOR EACH ACCOUNT OF LICENSEE, GENERATE A ROW OF THE ACCOUNT MASTER TABLE 84 BY OBTAINING THE INFORMATION NECESSARY TO PROCESS TRANSACTIONS ON THE ACCOUNT.

OBTAIN THE LIST OF ____ ____ ____. INSERT THIS INFORMATION INTO THE ACCOUNT COMMUNICATION LINKS MASTER TABLE.

FOR EACH TRANSACTION TYPE DESIRED TO BE PERFORMED BY LICENSEE, OBTAIN: (A) AN IDENTIFIER IDENTIFYING THE TRANSACTION; (B) A DESCRIPTION OF THE TRANSACTION; (C) A POSTING CODE; (D) A VALUE INDICATIVE OF WHETHER THE TRANSACTION REQUIRES A PROCESSING OF A FINANCIAL INSTRUMENT; (E) A BOOLEAN VALUE INDICATING WHETHER A SETTLEMENT OF A BUY OR SELL WILL BE PENDING. INSERT THIS INFORMATION INTO THE TRANSACTION MASTER TABLE. FOR EACH TRANSACTION TYPE IDENTIFIED ABOVE, DETERMINE A TRANSACTION DECOMPOSITION INTO SUBTRACTIONS AND ENCODE THE SUBTRACTION ACTIONS TO BE PERFORMED. INSERT THE (TRANSACTION IDENTIFIER, SUBTRANSACTION ENCODING) PAIR INTO THE TRANSACTION PROCESSING MASTER TABLE.

FIG. 4C

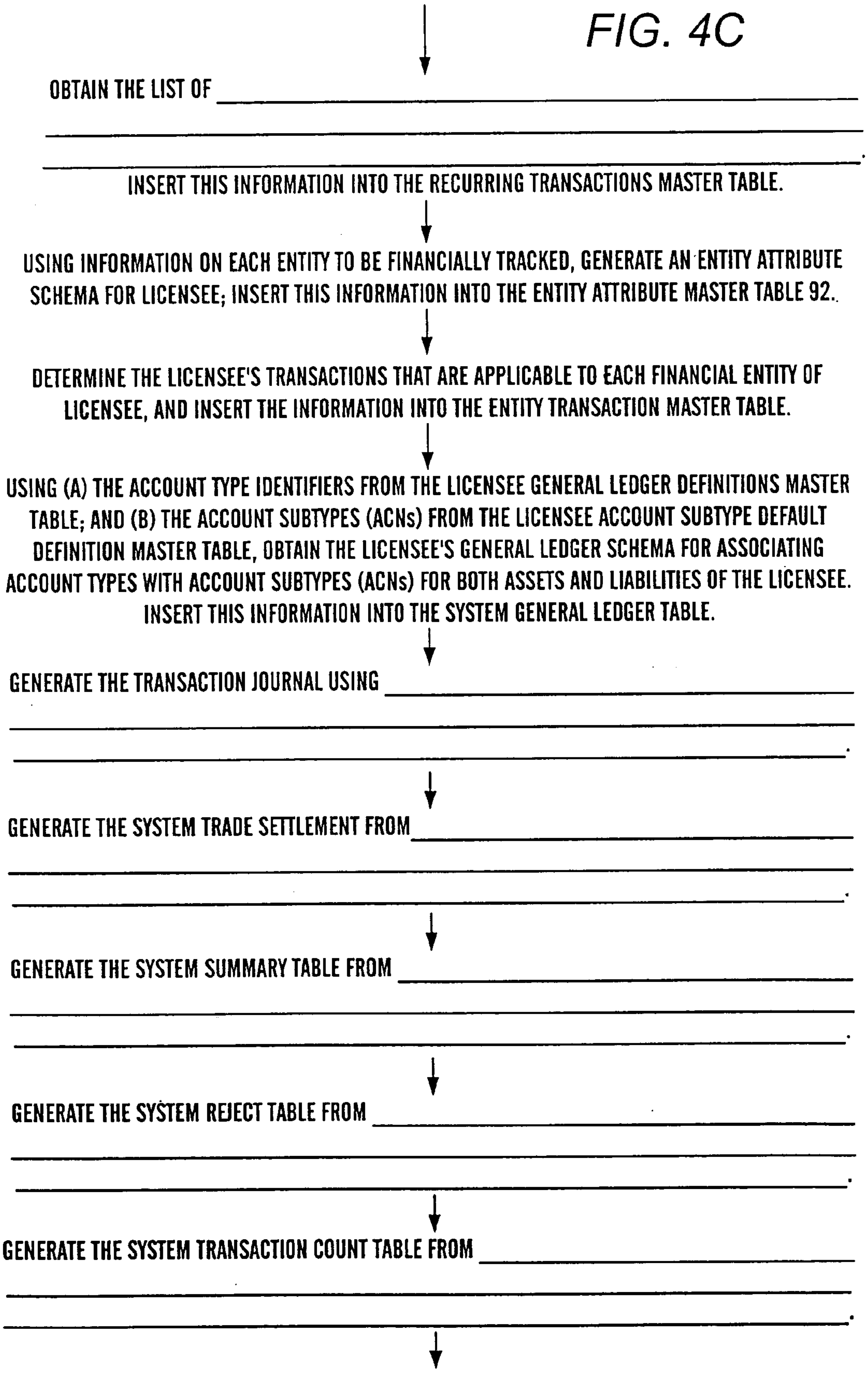
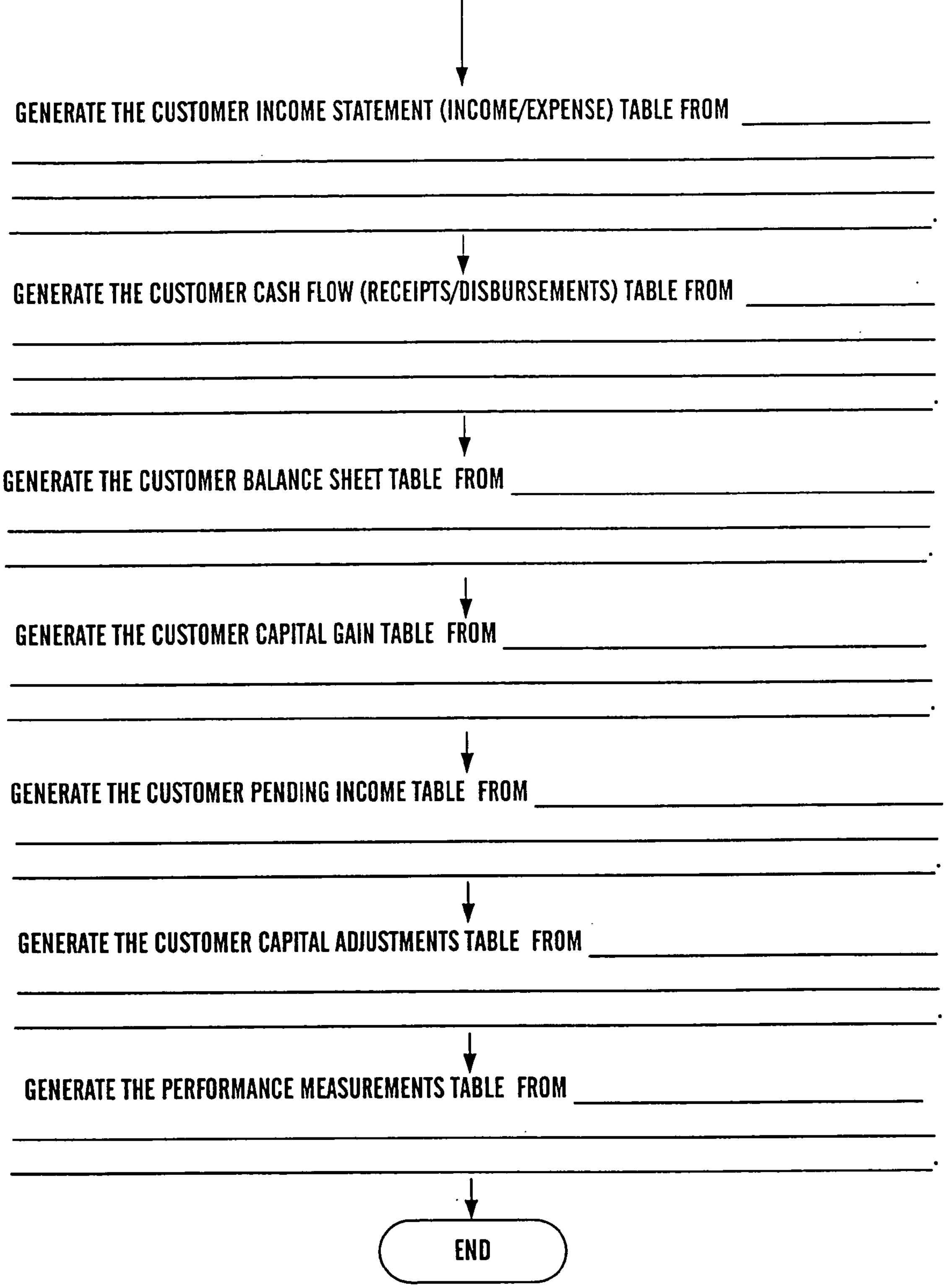

OBTAIN THE LIST OF ____________________ ____________________ ____________________.
INSERT THIS INFORMATION INTO THE RECURRING TRANSACTIONS MASTER TABLE.
USING INFORMATION ON EACH ENTITY TO BE FINANCIALLY TRACKED, GENERATE AN ENTITY ATTRIBUTE SCHEMA FOR LICENSEE; INSERT THIS INFORMATION INTO THE ENTITY ATTRIBUTE MASTER TABLE 92..
DETERMINE THE LICENSEE'S TRANSACTIONS THAT ARE APPLICABLE TO EACH FINANCIAL ENTITY OF LICENSEE, AND INSERT THE INFORMATION INTO THE ENTITY TRANSACTION MASTER TABLE.
USING (A) THE ACCOUNT TYPE IDENTIFIERS FROM THE LICENSEE GENERAL LEDGER DEFINITIONS MASTER TABLE; AND (B) THE ACCOUNT SUBTYPES (ACNs) FROM THE LICENSEE ACCOUNT SUBTYPE DEFAULT DEFINITION MASTER TABLE, OBTAIN THE LICENSEE'S GENERAL LEDGER SCHEMA FOR ASSOCIATING ACCOUNT TYPES WITH ACCOUNT SUBTYPES (ACNs) FOR BOTH ASSETS AND LIABILITIES OF THE LICENSEE. INSERT THIS INFORMATION INTO THE SYSTEM GENERAL LEDGER TABLE.
GENERATE THE TRANSACTION JOURNAL USING ____________________ ____________________ ____________________.
GENERATE THE SYSTEM TRADE SETTLEMENT FROM ____________________ ____________________ ____________________.
GENERATE THE SYSTEM SUMMARY TABLE FROM ____________________ ____________________ ____________________.
GENERATE THE SYSTEM REJECT TABLE FROM ____________________ ____________________ ____________________.
GENERATE THE SYSTEM TRANSACTION COUNT TABLE FROM ____________________ ____________________ ____________________.

FIG. 4D


GENERATE THE CUSTOMER INCOME STATEMENT (INCOME/EXPENSE) TABLE FROM
GENERATE THE CUSTOMER CASH FLOW (RECEIPTS/DISBURSEMENTS) TABLE FROM
GENERATE THE CUSTOMER BALANCE SHEET TABLE FROM
GENERATE THE CUSTOMER CAPITAL GAIN TABLE FROM
GENERATE THE CUSTOMER PENDING INCOME TABLE FROM
GENERATE THE CUSTOMER CAPITAL ADJUSTMENTS TABLE FROM
GENERATE THE PERFORMANCE MEASUREMENTS TABLE FROM
END

PROCESSING MODEL #1

FOR ALL "MASTER" TABLES THE FOLLOWING CONFIGURATION IS PROVIDED:

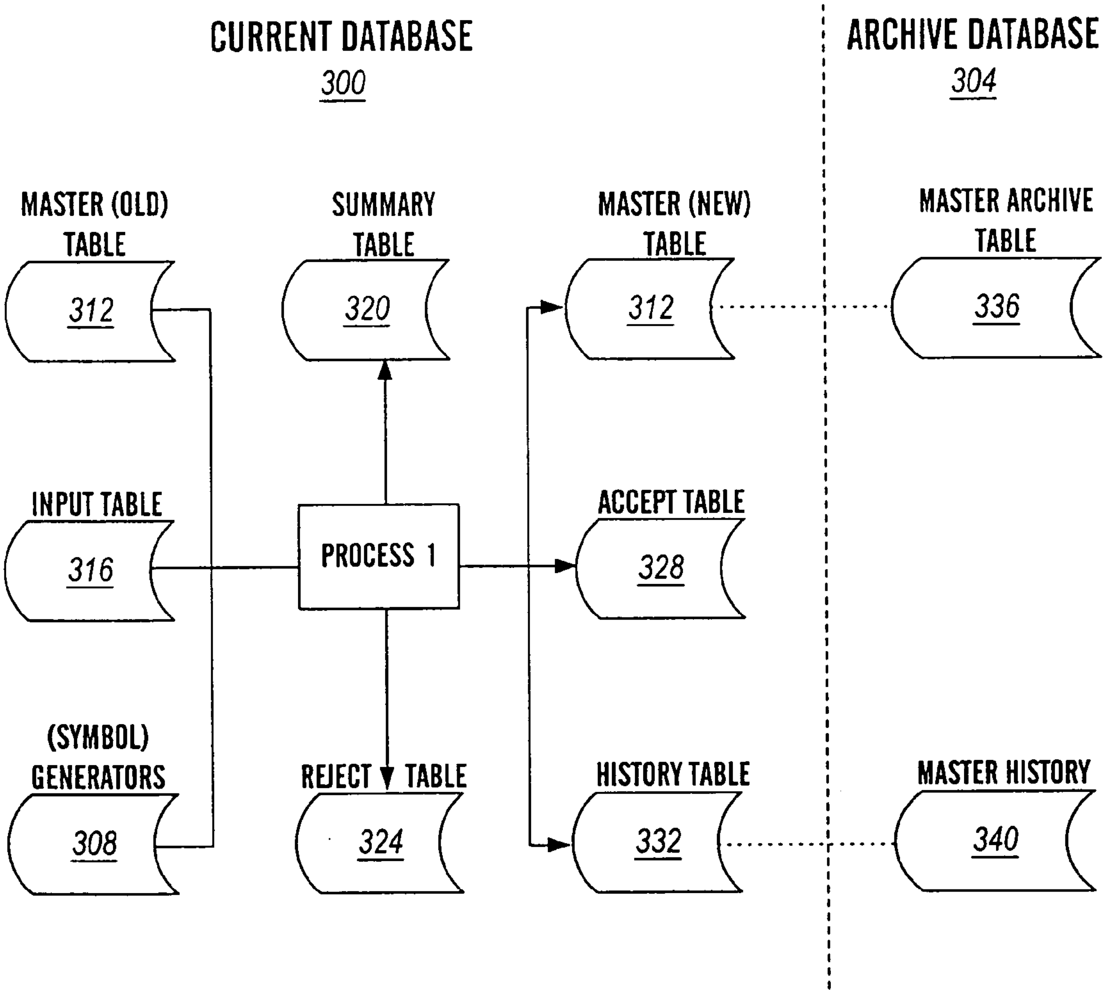


FIG. 5

TRANSACTION_CONTROLLER

FIG. 6

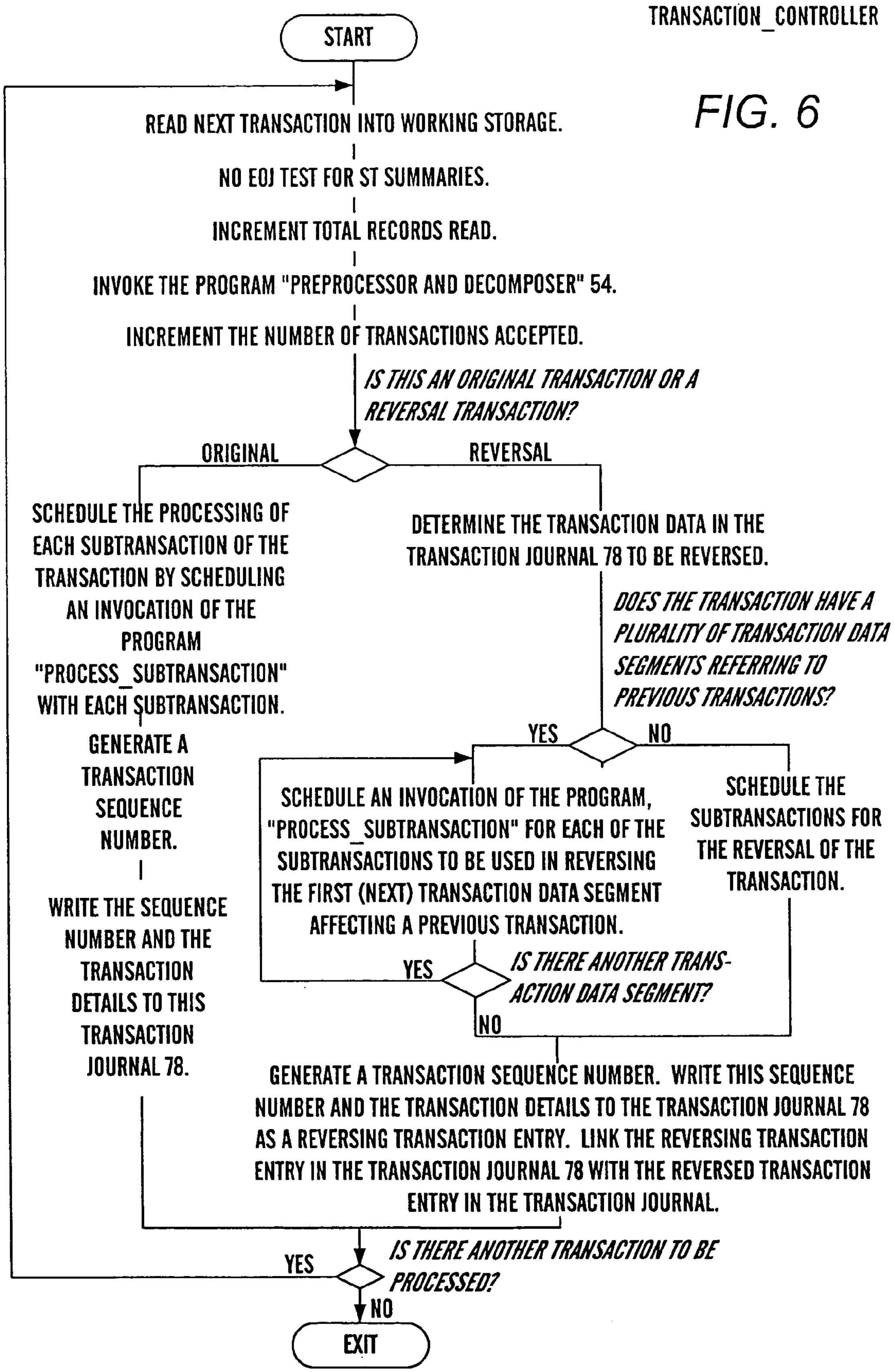
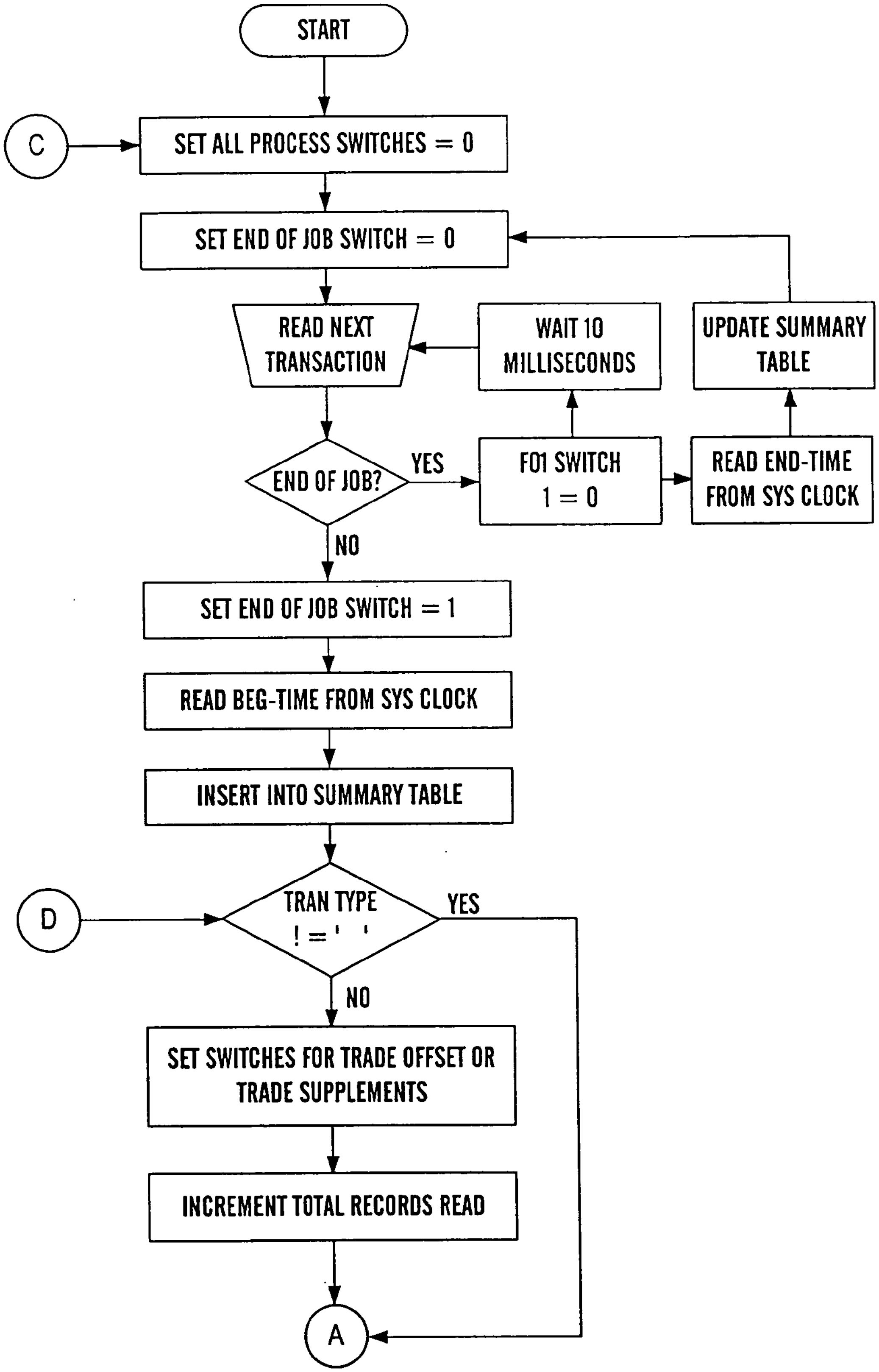

START
READ NEXT TRANSACTION INTO WORKING STORAGE.
NO EOJ TEST FOR ST SUMMARIES.
INCREMENT TOTAL RECORDS READ.
INVOKE THE PROGRAM "PREPROCESSOR AND DECOMPOSER" 54.
INCREMENT THE NUMBER OF TRANSACTIONS ACCEPTED.
IS THIS AN ORIGINAL TRANSACTION OR A REVERSAL TRANSACTION?
ORIGINAL
REVERSAL
SCHEDULE THE PROCESSING OF EACH SUBTRANSACTION OF THE TRANSACTION BY SCHEDULING AN INVOCATION OF THE PROGRAM "PROCESS_SUBTRANSACTION" WITH EACH SUBTRANSACTION.
GENERATE A TRANSACTION SEQUENCE NUMBER.
WRITE THE SEQUENCE NUMBER AND THE TRANSACTION DETAILS TO THIS TRANSACTION JOURNAL 78.
DETERMINE THE TRANSACTION DATA IN THE TRANSACTION JOURNAL 78 TO BE REVERSED.
DOES THE TRANSACTION HAVE A PLURALITY OF TRANSACTION DATA SEGMENTS REFERRING TO PREVIOUS TRANSACTIONS?
YES
NO
SCHEDULE AN INVOCATION OF THE PROGRAM, "PROCESS_SUBTRANSACTION" FOR EACH OF THE SUBTRANSACTIONS TO BE USED IN REVERSING THE FIRST (NEXT) TRANSACTION DATA SEGMENT AFFECTING A PREVIOUS TRANSACTION.
SCHEDULE THE SUBTRANSACTIONS FOR THE REVERSAL OF THE TRANSACTION.
YES
IS THERE ANOTHER TRANSACTION DATA SEGMENT?
NO
GENERATE A TRANSACTION SEQUENCE NUMBER. WRITE THIS SEQUENCE NUMBER AND THE TRANSACTION DETAILS TO THE TRANSACTION JOURNAL 78 AS A REVERSING TRANSACTION ENTRY. LINK THE REVERSING TRANSACTION ENTRY IN THE TRANSACTION JOURNAL 78 WITH THE REVERSED TRANSACTION ENTRY IN THE TRANSACTION JOURNAL.
YES
IS THERE ANOTHER TRANSACTION TO BE PROCESSED?
NO
EXIT

FIG. 6A

TRANSACTION CONTROLLER ("COMMAND")


START
C
SET ALL PROCESS SWITCHES = 0
SET END OF JOB SWITCH = 0
READ NEXT TRANSACTION
WAIT 10 MILLISECONDS
UPDATE SUMMARY TABLE
END OF JOB?
YES
F01 SWITCH 1 = 0
READ END-TIME FROM SYS CLOCK
NO
SET END OF JOB SWITCH = 1
READ BEG-TIME FROM SYS CLOCK
INSERT INTO SUMMARY TABLE
D
TRAN TYPE != ' '
YES
NO
SET SWITCHES FOR TRADE OFFSET OR TRADE SUPPLEMENTS
INCREMENT TOTAL RECORDS READ
A

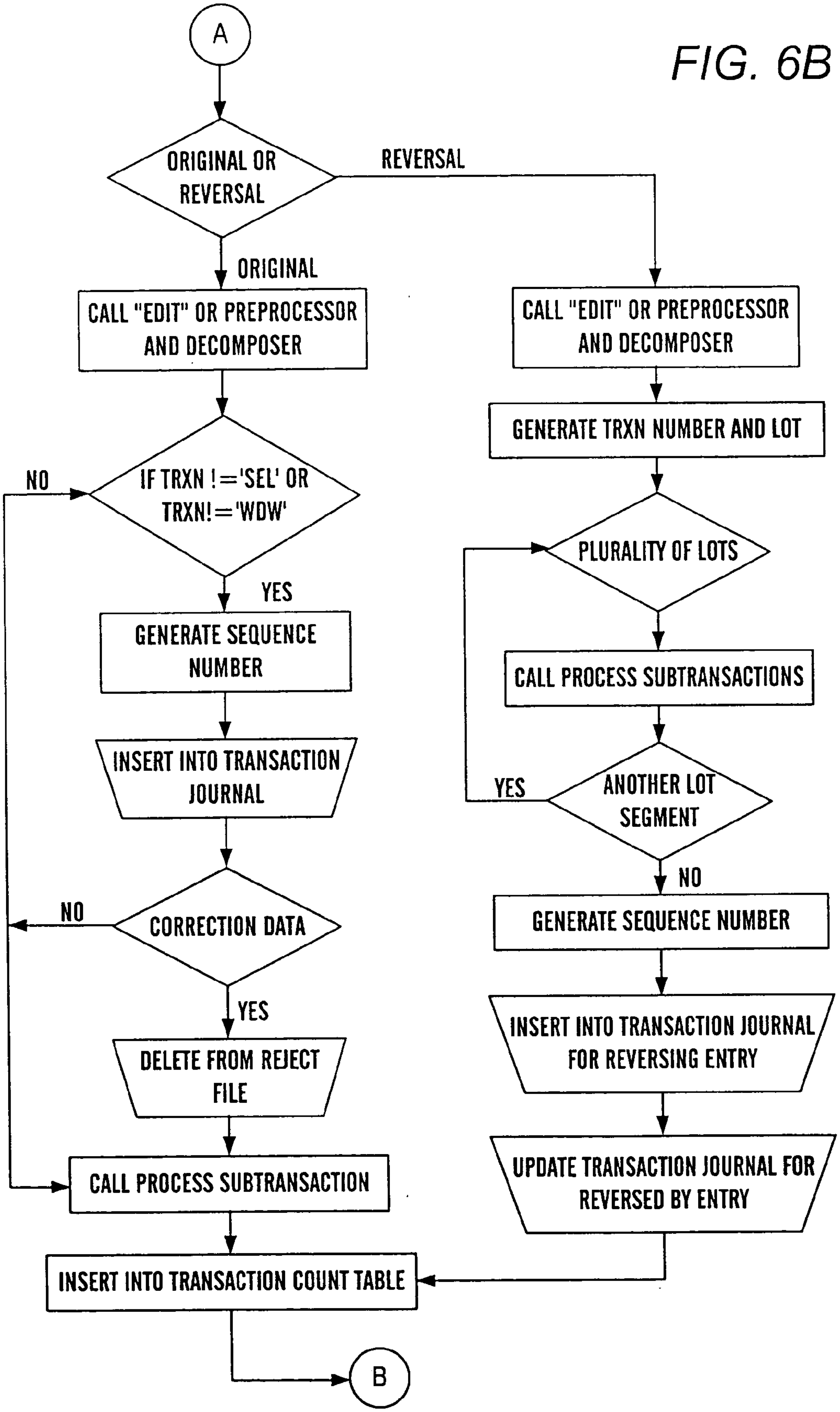

A
ORIGINAL OR REVERSAL
REVERSAL
ORIGINAL
CALL "EDIT" OR PREPROCESSOR AND DECOMPOSER
CALL "EDIT" OR PREPROCESSOR AND DECOMPOSER
GENERATE TRXN NUMBER AND LOT
NO
IF TRXN !='SEL' OR TRXN!='WDW'
YES
PLURALITY OF LOTS
GENERATE SEQUENCE NUMBER
CALL PROCESS SUBTRANSACTIONS
INSERT INTO TRANSACTION JOURNAL
YES
ANOTHER LOT SEGMENT
NO
GENERATE SEQUENCE NUMBER
NO
CORRECTION DATA
YES
INSERT INTO TRANSACTION JOURNAL FOR REVERSING ENTRY
DELETE FROM REJECT FILE
CALL PROCESS SUBTRANSACTION
UPDATE TRANSACTION JOURNAL FOR REVERSED BY ENTRY
INSERT INTO TRANSACTION COUNT TABLE
B

FIG. 6B

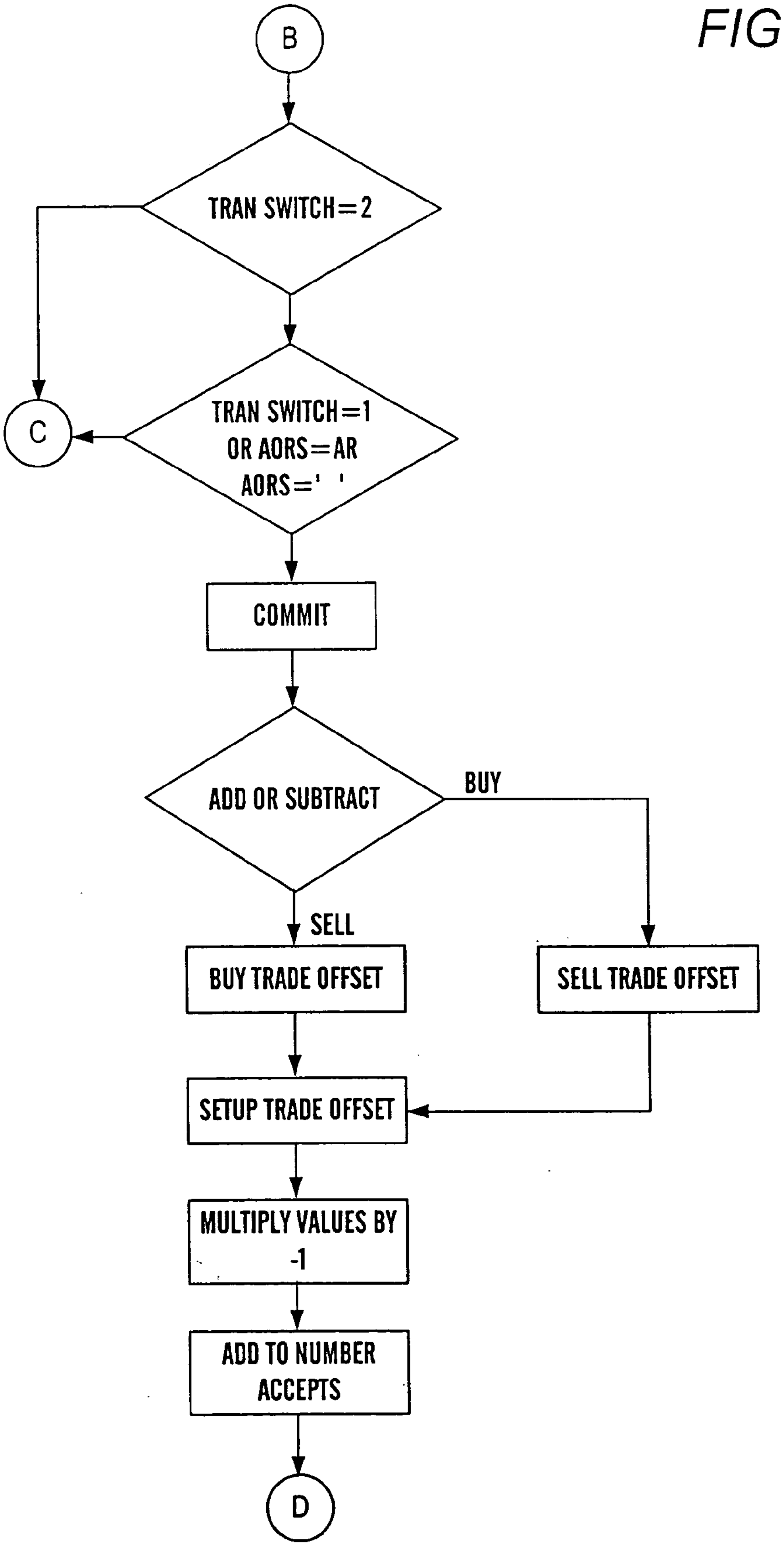

B
TRAN SWITCH=2
C
TRAN SWITCH=1
OR AORS=AR
AORS=' '
COMMIT
ADD OR SUBTRACT
BUY
SELL
BUY TRADE OFFSET
SELL TRADE OFFSET
SETUP TRADE OFFSET
MULTIPLY VALUES BY
-1
ADD TO NUMBER
ACCEPTS
D

FIG. 6C

PREPROCESSOR AND DECOMPOSER (TRANSACTION)

FIG. 7A

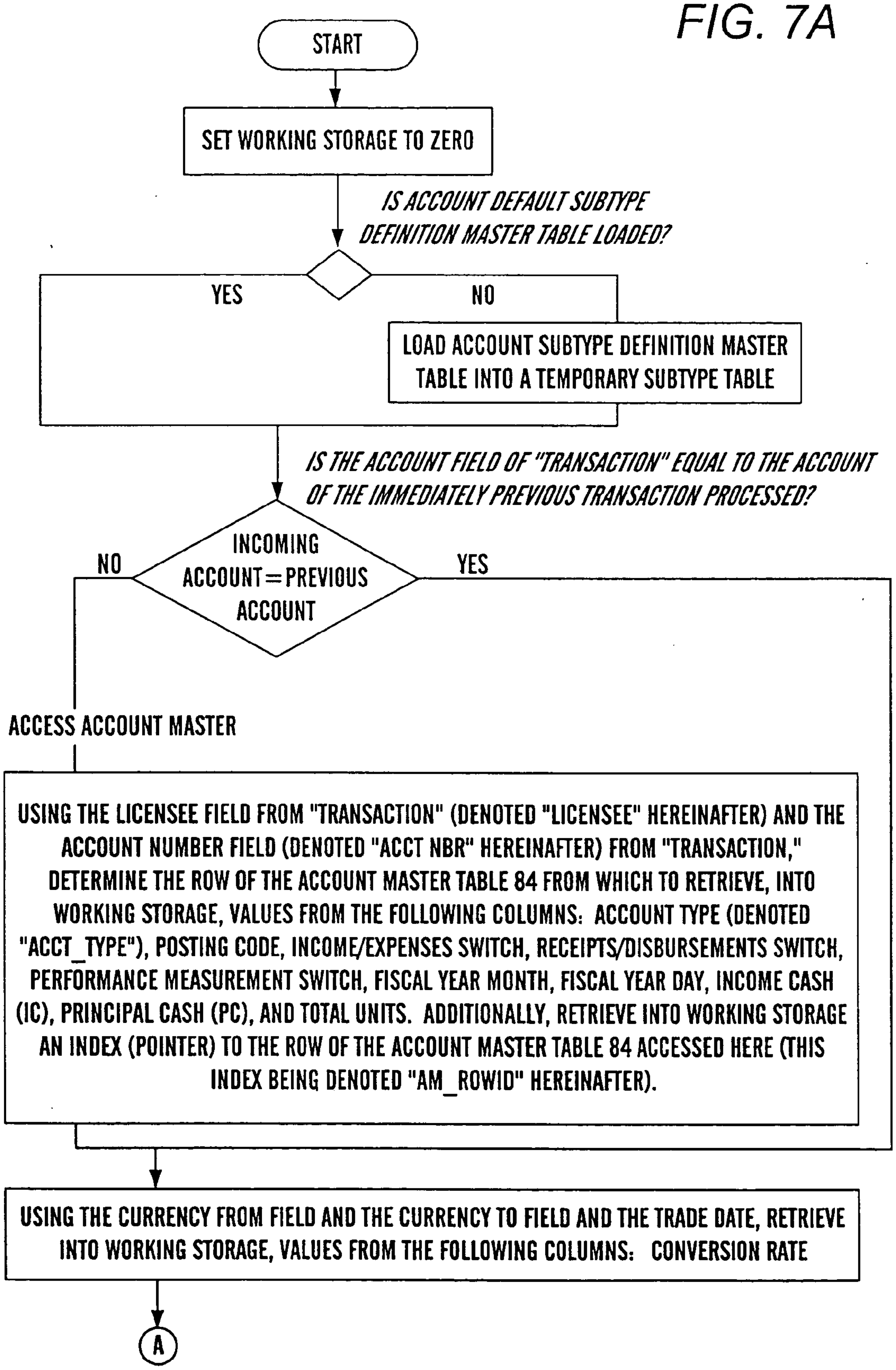
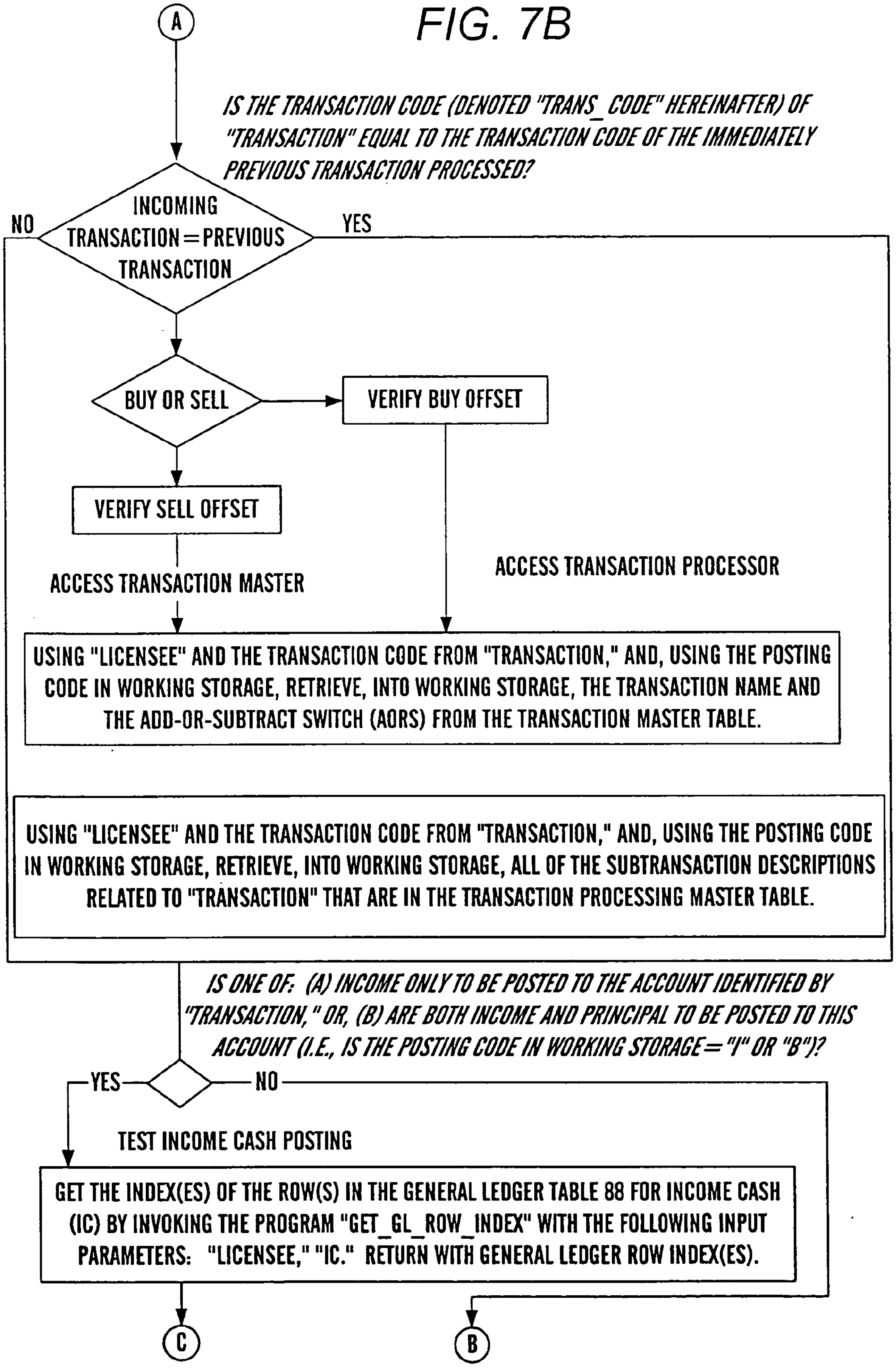

START
SET WORKING STORAGE TO ZERO
IS ACCOUNT DEFAULT SUBTYPE DEFINITION MASTER TABLE LOADED?
YES
NO
LOAD ACCOUNT SUBTYPE DEFINITION MASTER TABLE INTO A TEMPORARY SUBTYPE TABLE
IS THE ACCOUNT FIELD OF "TRANSACTION" EQUAL TO THE ACCOUNT OF THE IMMEDIATELY PREVIOUS TRANSACTION PROCESSED?
INCOMING ACCOUNT=PREVIOUS ACCOUNT
NO
YES
ACCESS ACCOUNT MASTER
USING THE LICENSEE FIELD FROM "TRANSACTION" (DENOTED "LICENSEE" HEREINAFTER) AND THE ACCOUNT NUMBER FIELD (DENOTED "ACCT NBR" HEREINAFTER) FROM "TRANSACTION," DETERMINE THE ROW OF THE ACCOUNT MASTER TABLE 84 FROM WHICH TO RETRIEVE, INTO WORKING STORAGE, VALUES FROM THE FOLLOWING COLUMNS: ACCOUNT TYPE (DENOTED "ACCT_TYPE"), POSTING CODE, INCOME/EXPENSES SWITCH, RECEIPTS/DISBURSEMENTS SWITCH, PERFORMANCE MEASUREMENT SWITCH, FISCAL YEAR MONTH, FISCAL YEAR DAY, INCOME CASH (IC), PRINCIPAL CASH (PC), AND TOTAL UNITS. ADDITIONALLY, RETRIEVE INTO WORKING STORAGE AN INDEX (POINTER) TO THE ROW OF THE ACCOUNT MASTER TABLE 84 ACCESSED HERE (THIS INDEX BEING DENOTED "AM_ROWID" HEREINAFTER).
USING THE CURRENCY FROM FIELD AND THE CURRENCY TO FIELD AND THE TRADE DATE, RETRIEVE INTO WORKING STORAGE, VALUES FROM THE FOLLOWING COLUMNS: CONVERSION RATE
A

FIG. 7B


A
IS THE TRANSACTION CODE (DENOTED "TRANS_CODE" HEREINAFTER) OF "TRANSACTION" EQUAL TO THE TRANSACTION CODE OF THE IMMEDIATELY PREVIOUS TRANSACTION PROCESSED?
INCOMING TRANSACTION = PREVIOUS TRANSACTION
NO
YES
BUY OR SELL
VERIFY BUY OFFSET
VERIFY SELL OFFSET
ACCESS TRANSACTION MASTER
ACCESS TRANSACTION PROCESSOR
USING "LICENSEE" AND THE TRANSACTION CODE FROM "TRANSACTION," AND, USING THE POSTING CODE IN WORKING STORAGE, RETRIEVE, INTO WORKING STORAGE, THE TRANSACTION NAME AND THE ADD-OR-SUBTRACT SWITCH (AORS) FROM THE TRANSACTION MASTER TABLE.
USING "LICENSEE" AND THE TRANSACTION CODE FROM "TRANSACTION," AND, USING THE POSTING CODE IN WORKING STORAGE, RETRIEVE, INTO WORKING STORAGE, ALL OF THE SUBTRANSACTION DESCRIPTIONS RELATED TO "TRANSACTION" THAT ARE IN THE TRANSACTION PROCESSING MASTER TABLE.
IS ONE OF: (A) INCOME ONLY TO BE POSTED TO THE ACCOUNT IDENTIFIED BY "TRANSACTION," OR, (B) ARE BOTH INCOME AND PRINCIPAL TO BE POSTED TO THIS ACCOUNT (I.E., IS THE POSTING CODE IN WORKING STORAGE = "I" OR "B")?
YES
NO
TEST INCOME CASH POSTING
GET THE INDEX(ES) OF THE ROW(S) IN THE GENERAL LEDGER TABLE 88 FOR INCOME CASH (IC) BY INVOKING THE PROGRAM "GET_GL_ROW_INDEX" WITH THE FOLLOWING INPUT PARAMETERS: "LICENSEE," "IC." RETURN WITH GENERAL LEDGER ROW INDEX(ES).
C
B

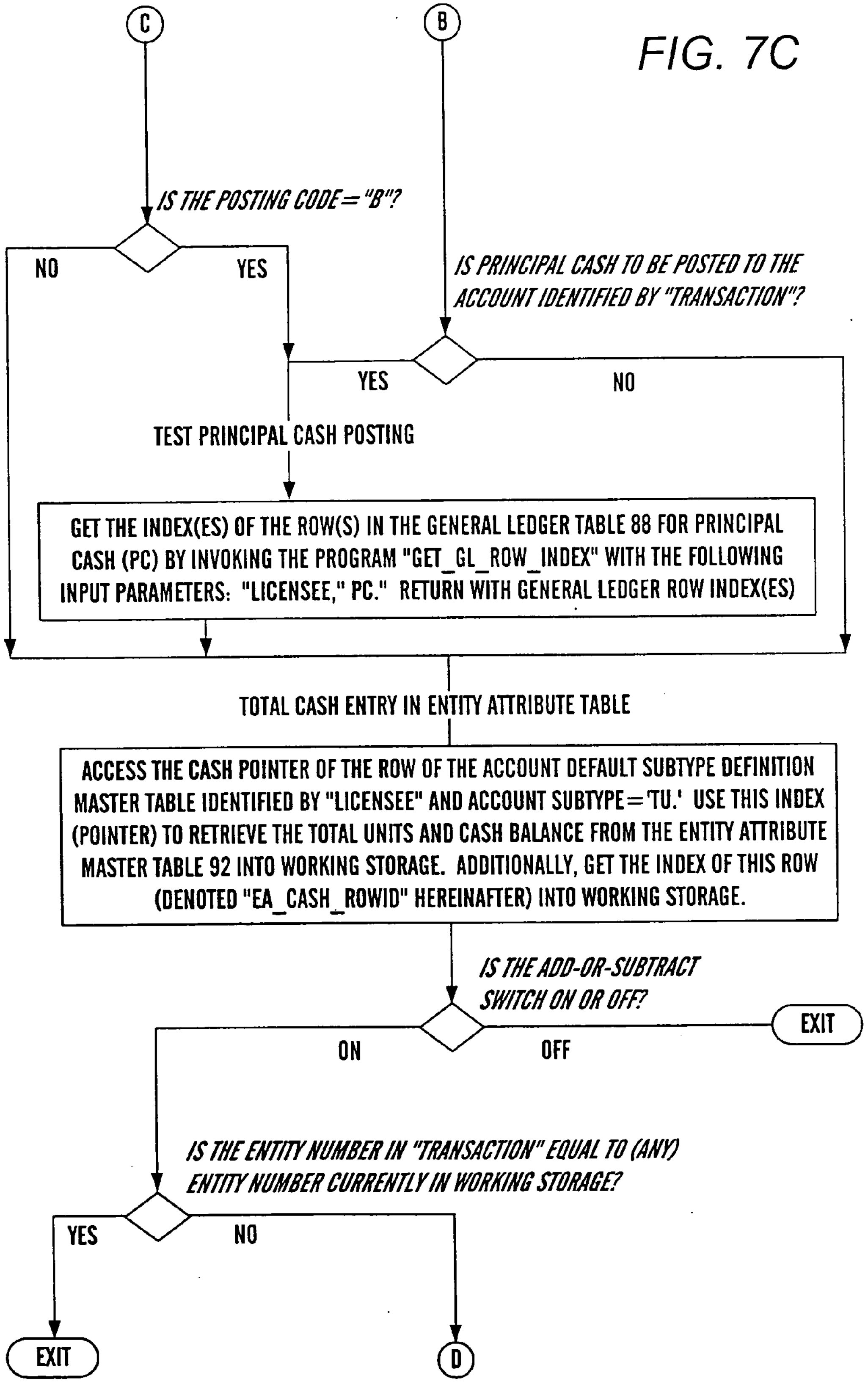
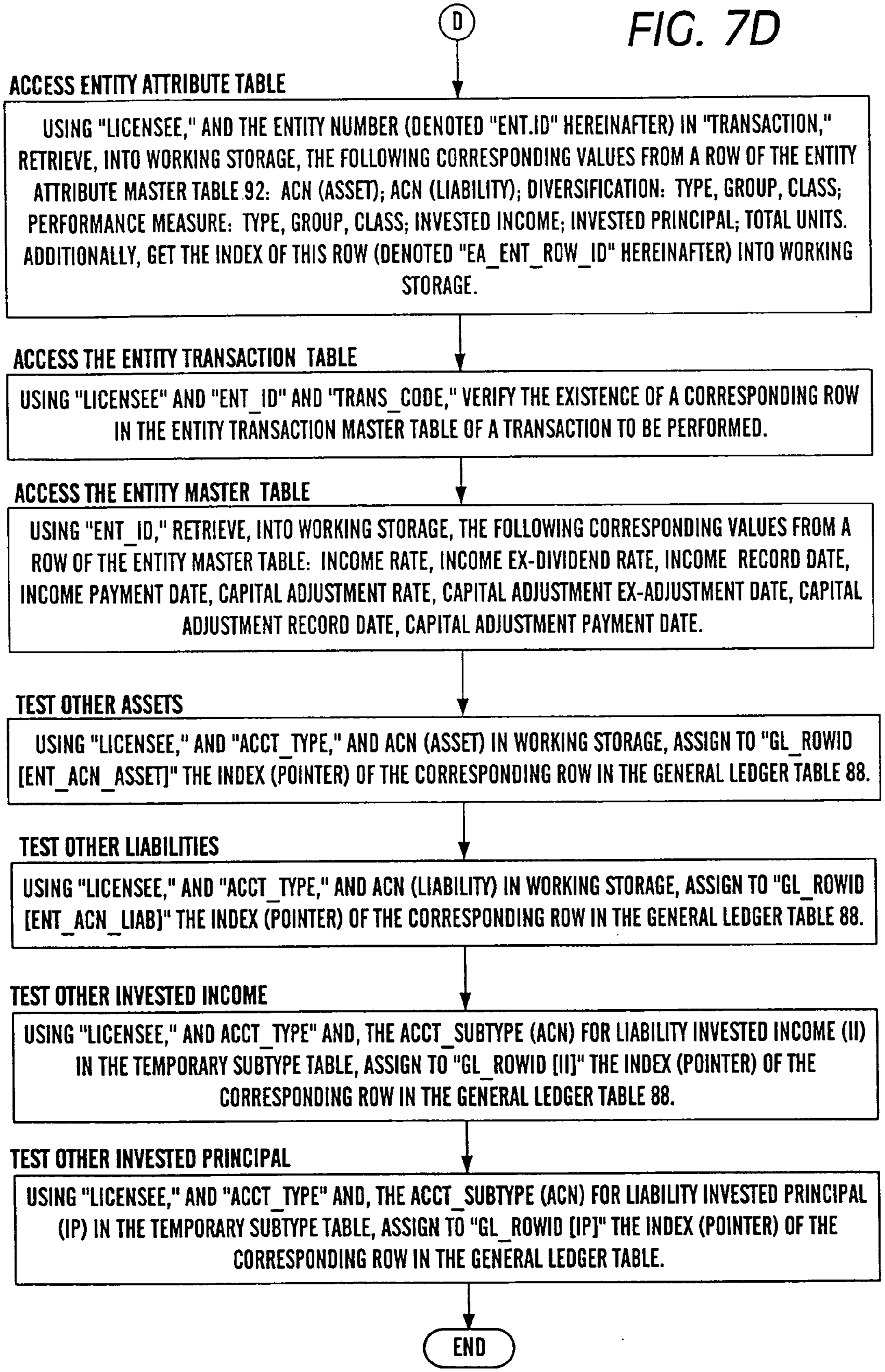

C
B
IS THE POSTING CODE="B"?
NO
YES
IS PRINCIPAL CASH TO BE POSTED TO THE ACCOUNT IDENTIFIED BY "TRANSACTION"?
YES
NO
TEST PRINCIPAL CASH POSTING
GET THE INDEX(ES) OF THE ROW(S) IN THE GENERAL LEDGER TABLE 88 FOR PRINCIPAL CASH (PC) BY INVOKING THE PROGRAM "GET_GL_ROW_INDEX" WITH THE FOLLOWING INPUT PARAMETERS: "LICENSEE," PC." RETURN WITH GENERAL LEDGER ROW INDEX(ES)
TOTAL CASH ENTRY IN ENTITY ATTRIBUTE TABLE
ACCESS THE CASH POINTER OF THE ROW OF THE ACCOUNT DEFAULT SUBTYPE DEFINITION MASTER TABLE IDENTIFIED BY "LICENSEE" AND ACCOUNT SUBTYPE='TU.' USE THIS INDEX (POINTER) TO RETRIEVE THE TOTAL UNITS AND CASH BALANCE FROM THE ENTITY ATTRIBUTE MASTER TABLE 92 INTO WORKING STORAGE. ADDITIONALLY, GET THE INDEX OF THIS ROW (DENOTED "EA_CASH_ROWID" HEREINAFTER) INTO WORKING STORAGE.
IS THE ADD-OR-SUBTRACT SWITCH ON OR OFF?
ON
OFF
EXIT
IS THE ENTITY NUMBER IN "TRANSACTION" EQUAL TO (ANY) ENTITY NUMBER CURRENTLY IN WORKING STORAGE?
YES
NO
EXIT
D

FIG. 7C

FIG. 7D



ACCESS ENTITY ATTRIBUTE TABLE

USING "LICENSEE," AND THE ENTITY NUMBER (DENOTED "ENT.ID" HEREINAFTER) IN "TRANSACTION," RETRIEVE, INTO WORKING STORAGE, THE FOLLOWING CORRESPONDING VALUES FROM A ROW OF THE ENTITY ATTRIBUTE MASTER TABLE 92: ACN (ASSET); ACN (LIABILITY); DIVERSIFICATION: TYPE, GROUP, CLASS; PERFORMANCE MEASURE: TYPE, GROUP, CLASS; INVESTED INCOME; INVESTED PRINCIPAL; TOTAL UNITS. ADDITIONALLY, GET THE INDEX OF THIS ROW (DENOTED "EA_ENT_ROW_ID" HEREINAFTER) INTO WORKING STORAGE.

ACCESS THE ENTITY TRANSACTION TABLE

USING "LICENSEE" AND "ENT_ID" AND "TRANS_CODE," VERIFY THE EXISTENCE OF A CORRESPONDING ROW IN THE ENTITY TRANSACTION MASTER TABLE OF A TRANSACTION TO BE PERFORMED.

ACCESS THE ENTITY MASTER TABLE

USING "ENT_ID," RETRIEVE, INTO WORKING STORAGE, THE FOLLOWING CORRESPONDING VALUES FROM A ROW OF THE ENTITY MASTER TABLE: INCOME RATE, INCOME EX-DIVIDEND RATE, INCOME RECORD DATE, INCOME PAYMENT DATE, CAPITAL ADJUSTMENT RATE, CAPITAL ADJUSTMENT EX-ADJUSTMENT DATE, CAPITAL ADJUSTMENT RECORD DATE, CAPITAL ADJUSTMENT PAYMENT DATE.

TEST OTHER ASSETS

USING "LICENSEE," AND "ACCT_TYPE," AND ACN (ASSET) IN WORKING STORAGE, ASSIGN TO "GL_ROWID [ENT_ACN_ASSET]" THE INDEX (POINTER) OF THE CORRESPONDING ROW IN THE GENERAL LEDGER TABLE 88.

TEST OTHER LIABILITIES

USING "LICENSEE," AND "ACCT_TYPE," AND ACN (LIABILITY) IN WORKING STORAGE, ASSIGN TO "GL_ROWID [ENT_ACN_LIAB]" THE INDEX (POINTER) OF THE CORRESPONDING ROW IN THE GENERAL LEDGER TABLE 88.

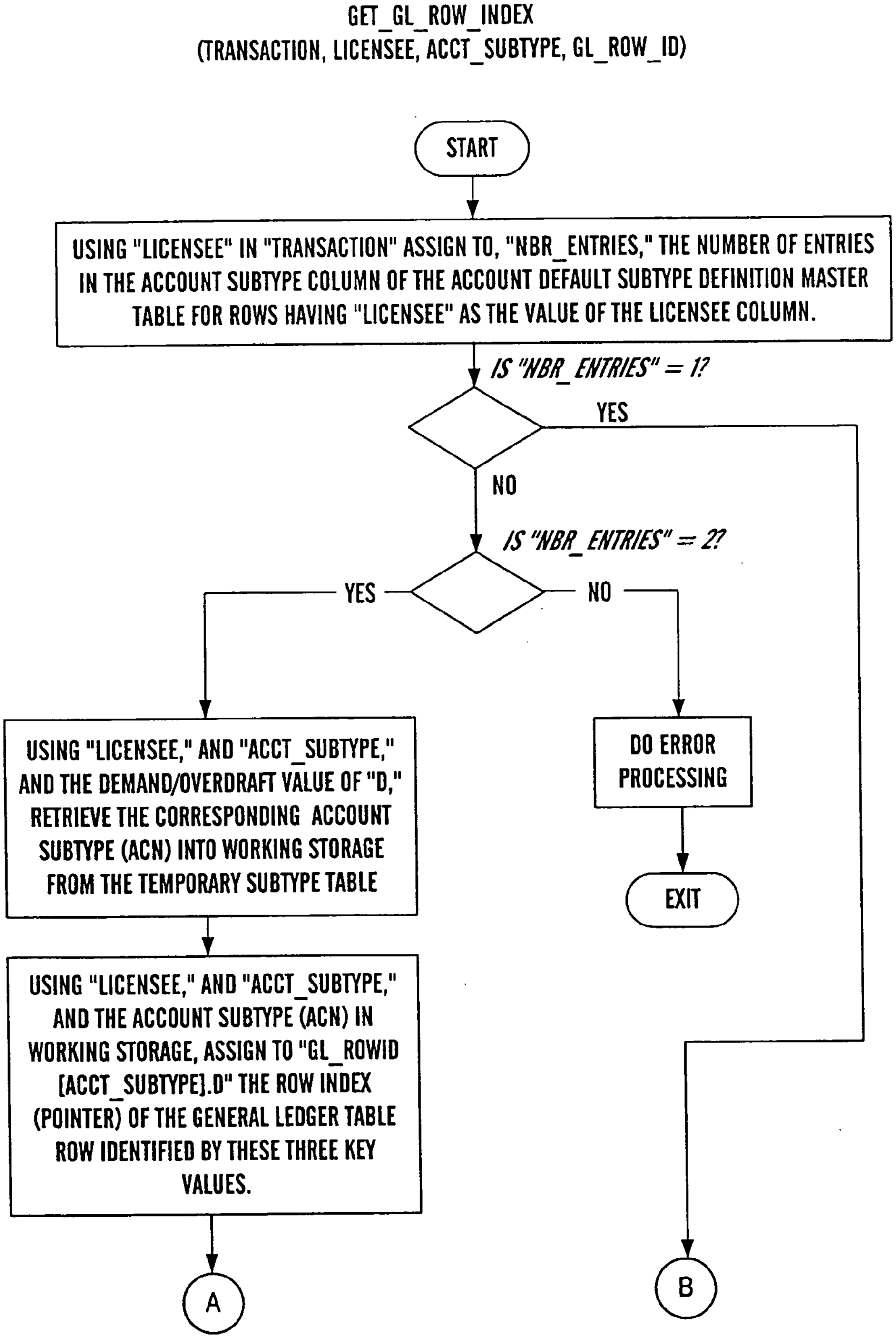
TEST OTHER INVESTED INCOME

USING "LICENSEE," AND ACCT_TYPE" AND, THE ACCT_SUBTYPE (ACN) FOR LIABILITY INVESTED INCOME (II) IN THE TEMPORARY SUBTYPE TABLE, ASSIGN TO "GL_ROWID [II]" THE INDEX (POINTER) OF THE CORRESPONDING ROW IN THE GENERAL LEDGER TABLE 88.

TEST OTHER INVESTED PRINCIPAL

USING "LICENSEE," AND "ACCT_TYPE" AND, THE ACCT_SUBTYPE (ACN) FOR LIABILITY INVESTED PRINCIPAL (IP) IN THE TEMPORARY SUBTYPE TABLE, ASSIGN TO "GL_ROWID [IP]" THE INDEX (POINTER) OF THE CORRESPONDING ROW IN THE GENERAL LEDGER TABLE.

FIG. 8A


GET_GL_ROW_INDEX
(TRANSACTION, LICENSEE, ACCT_SUBTYPE, GL_ROW_ID)
START
USING "LICENSEE" IN "TRANSACTION" ASSIGN TO, "NBR_ENTRIES," THE NUMBER OF ENTRIES IN THE ACCOUNT SUBTYPE COLUMN OF THE ACCOUNT DEFAULT SUBTYPE DEFINITION MASTER TABLE FOR ROWS HAVING "LICENSEE" AS THE VALUE OF THE LICENSEE COLUMN.
IS "NBR_ENTRIES" = 1?
YES
NO
IS "NBR_ENTRIES" = 2?
YES
NO
USING "LICENSEE," AND "ACCT_SUBTYPE," AND THE DEMAND/OVERDRAFT VALUE OF "D," RETRIEVE THE CORRESPONDING ACCOUNT SUBTYPE (ACN) INTO WORKING STORAGE FROM THE TEMPORARY SUBTYPE TABLE
DO ERROR PROCESSING
EXIT
USING "LICENSEE," AND "ACCT_SUBTYPE," AND THE ACCOUNT SUBTYPE (ACN) IN WORKING STORAGE, ASSIGN TO "GL_ROWID [ACCT_SUBTYPE].D" THE ROW INDEX (POINTER) OF THE GENERAL LEDGER TABLE ROW IDENTIFIED BY THESE THREE KEY VALUES.
A
B

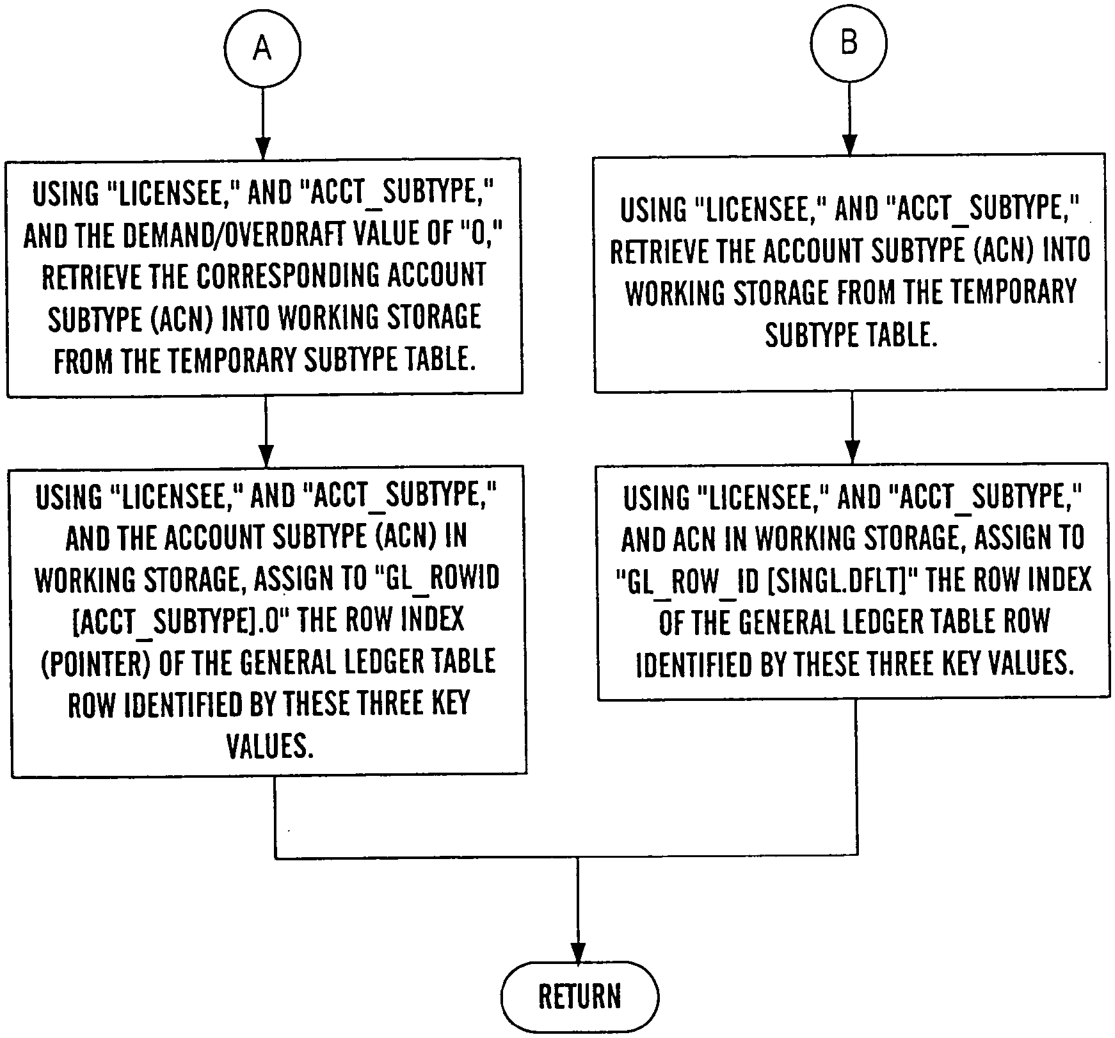

A
USING "LICENSEE," AND "ACCT_SUBTYPE," AND THE DEMAND/OVERDRAFT VALUE OF "0," RETRIEVE THE CORRESPONDING ACCOUNT SUBTYPE (ACN) INTO WORKING STORAGE FROM THE TEMPORARY SUBTYPE TABLE.
USING "LICENSEE," AND "ACCT_SUBTYPE," AND THE ACCOUNT SUBTYPE (ACN) IN WORKING STORAGE, ASSIGN TO "GL_ROWID [ACCT_SUBTYPE].0" THE ROW INDEX (POINTER) OF THE GENERAL LEDGER TABLE ROW IDENTIFIED BY THESE THREE KEY VALUES.
B
USING "LICENSEE," AND "ACCT_SUBTYPE," RETRIEVE THE ACCOUNT SUBTYPE (ACN) INTO WORKING STORAGE FROM THE TEMPORARY SUBTYPE TABLE.
USING "LICENSEE," AND "ACCT_SUBTYPE," AND ACN IN WORKING STORAGE, ASSIGN TO "GL_ROW_ID [SINGL.DFLT]" THE ROW INDEX OF THE GENERAL LEDGER TABLE ROW IDENTIFIED BY THESE THREE KEY VALUES.
RETURN

FIG. 8B

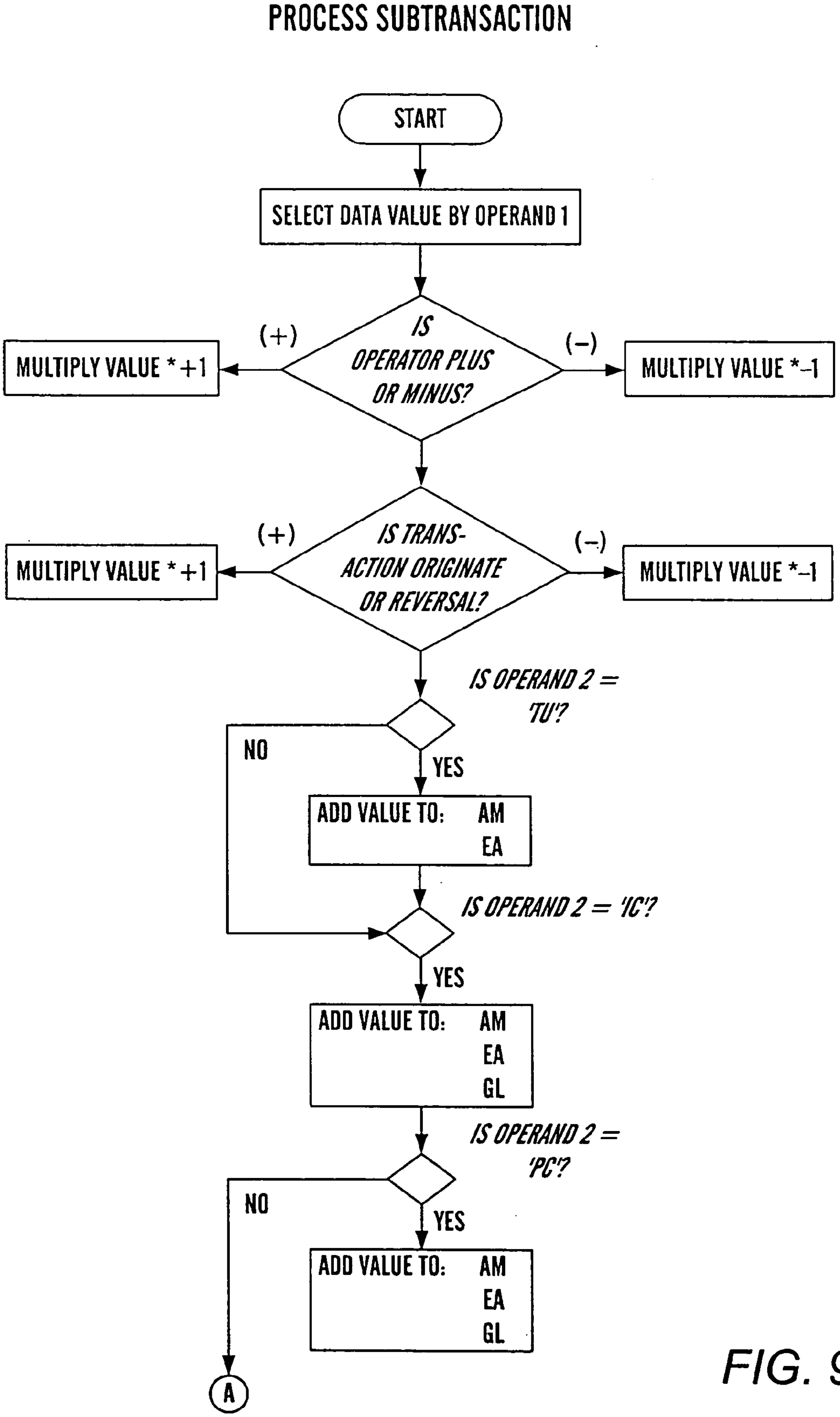

PROCESS SUBTRANSACTION
START
SELECT DATA VALUE BY OPERAND 1
IS OPERATOR PLUS OR MINUS?
(+)
MULTIPLY VALUE * +1
(–)
MULTIPLY VALUE *–1
IS TRANSACTION ORIGINATE OR REVERSAL?
(+)
MULTIPLY VALUE * +1
(–)
MULTIPLY VALUE *–1
IS OPERAND 2 = 'TU'?
NO
YES
ADD VALUE TO: AM EA
IS OPERAND 2 = 'IC'?
YES
ADD VALUE TO: AM EA GL
IS OPERAND 2 = 'PC'?
NO
YES
ADD VALUE TO: AM EA GL
A

FIG. 9A

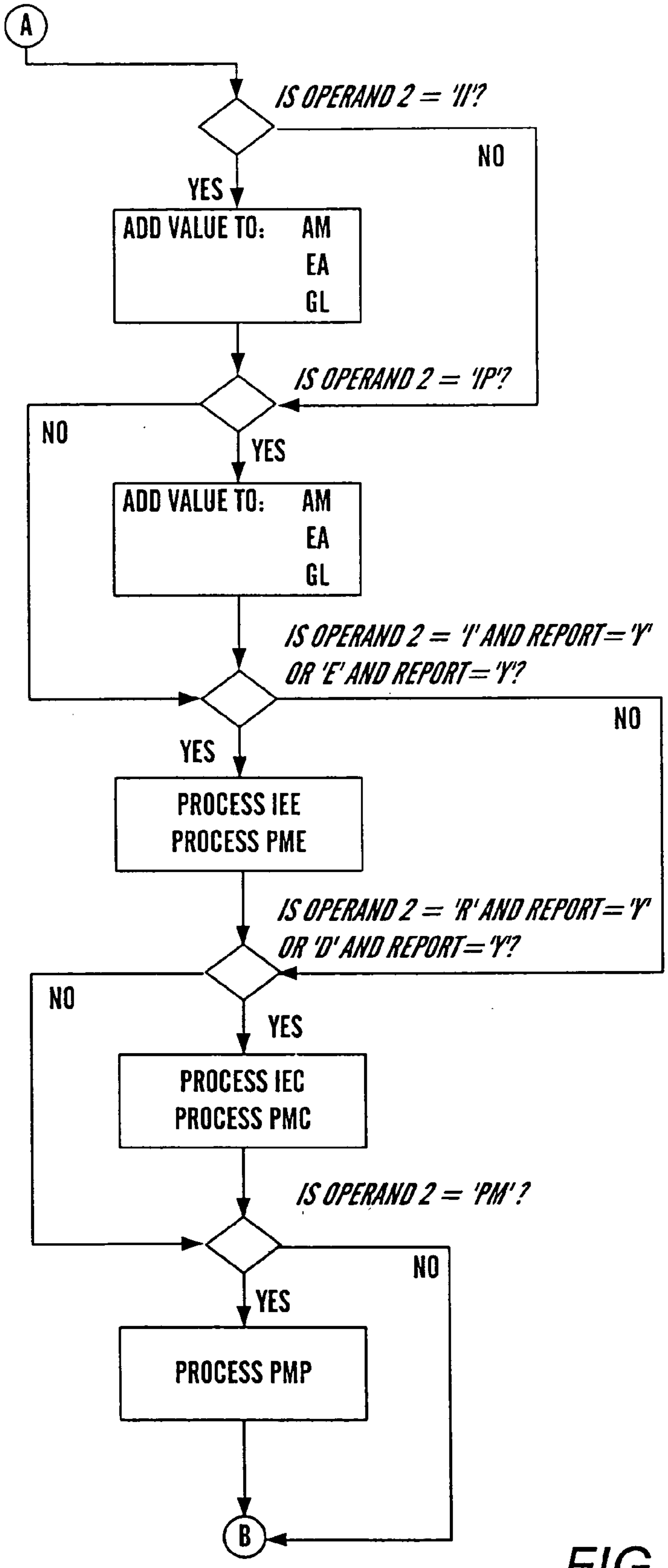

A
IS OPERAND 2 = 'II'?
NO
YES
ADD VALUE TO: AM
EA
GL
IS OPERAND 2 = 'IP'?
NO
YES
ADD VALUE TO: AM
EA
GL
IS OPERAND 2 = 'I' AND REPORT='Y'
OR 'E' AND REPORT='Y'?
NO
YES
PROCESS IEE
PROCESS PME
IS OPERAND 2 = 'R' AND REPORT='Y'
OR 'D' AND REPORT='Y'?
NO
YES
PROCESS IEC
PROCESS PMC
IS OPERAND 2 = 'PM' ?
NO
YES
PROCESS PMP
B

FIG. 9B

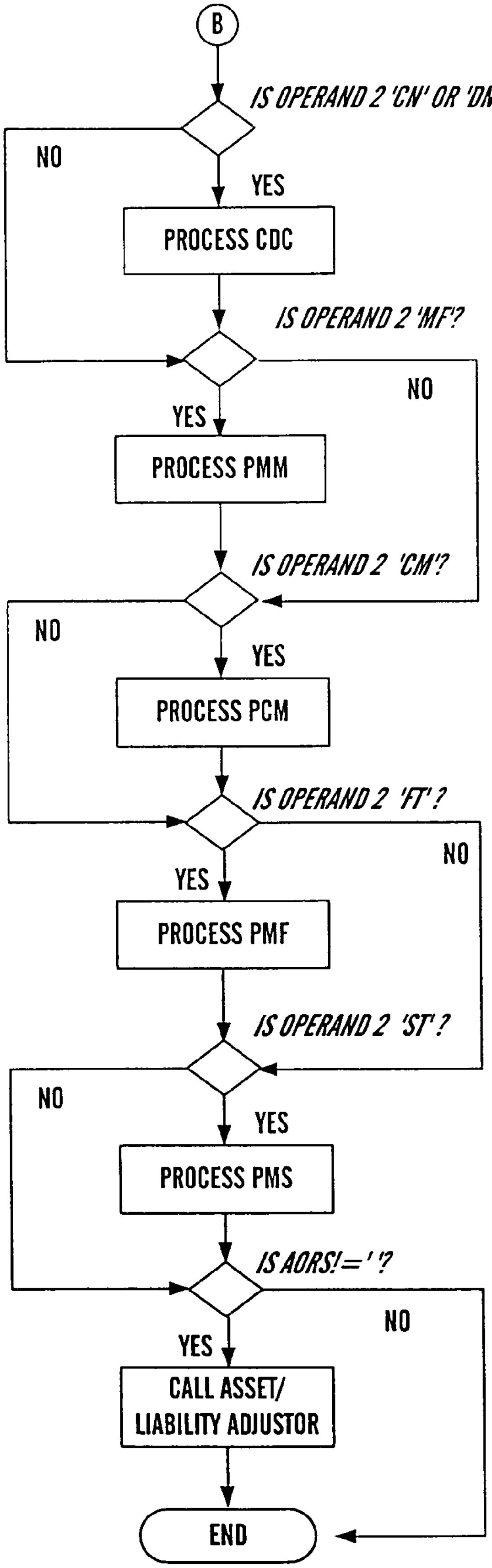
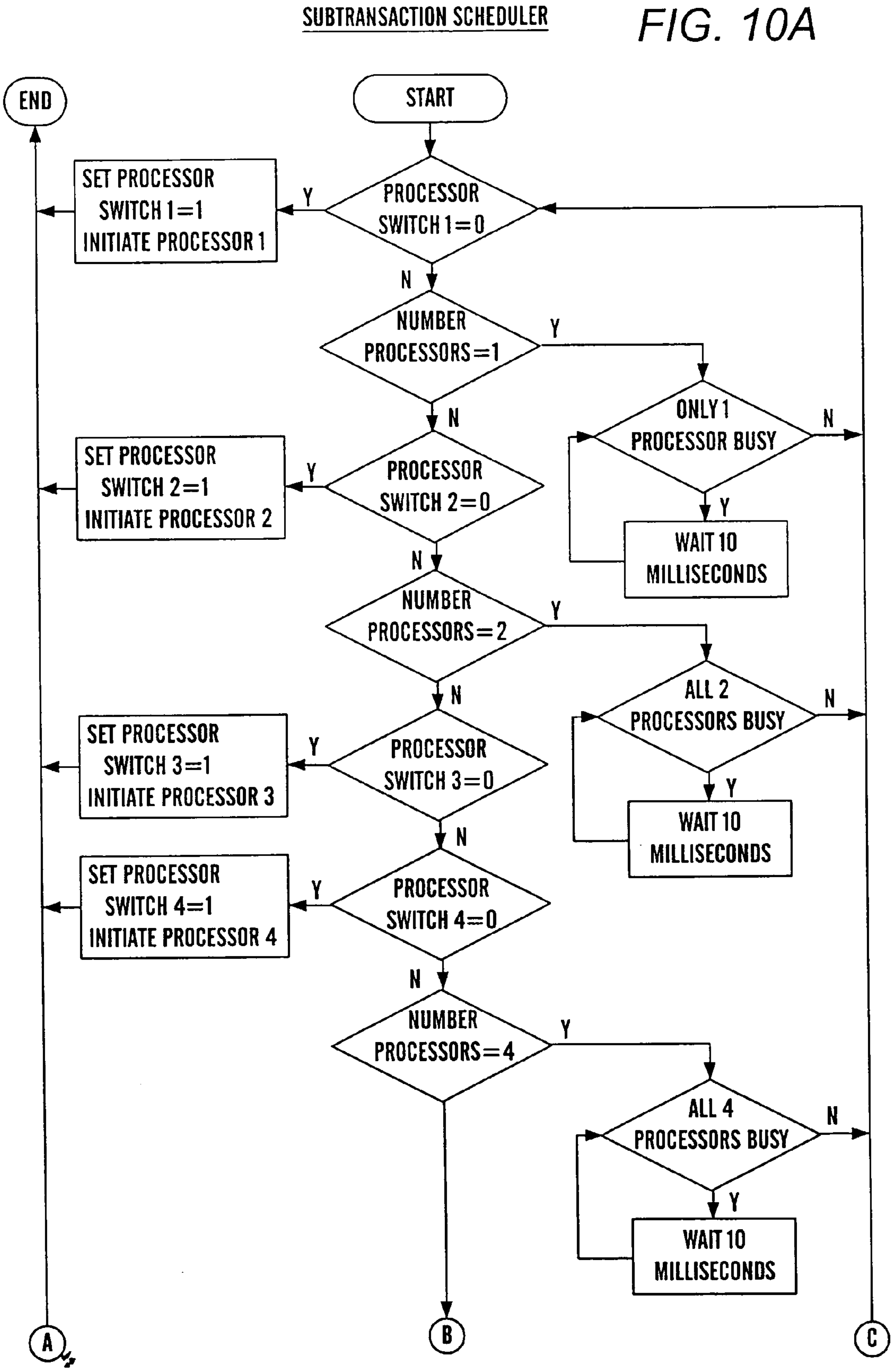

B
IS OPERAND 2 'CN' OR 'DN'?
NO
YES
PROCESS CDC
IS OPERAND 2 'MF'?
NO
YES
PROCESS PMM
IS OPERAND 2 'CM'?
NO
YES
PROCESS PCM
IS OPERAND 2 'FT'?
NO
YES
PROCESS PMF
IS OPERAND 2 'ST'?
NO
YES
PROCESS PMS
IS AORS!=' '?
NO
YES
CALL ASSET/
LIABILITY ADJUSTOR
END

FIG. 9C

FIG. 10A


SUBTRANSACTION SCHEDULER
START
END
PROCESSOR SWITCH 1=0
Y
SET PROCESSOR SWITCH 1=1 INITIATE PROCESSOR 1
N
NUMBER PROCESSORS=1
Y
ONLY 1 PROCESSOR BUSY
N
Y
WAIT 10 MILLISECONDS
N
PROCESSOR SWITCH 2=0
Y
SET PROCESSOR SWITCH 2=1 INITIATE PROCESSOR 2
N
NUMBER PROCESSORS=2
Y
ALL 2 PROCESSORS BUSY
N
Y
WAIT 10 MILLISECONDS
N
PROCESSOR SWITCH 3=0
Y
SET PROCESSOR SWITCH 3=1 INITIATE PROCESSOR 3
N
PROCESSOR SWITCH 4=0
Y
SET PROCESSOR SWITCH 4=1 INITIATE PROCESSOR 4
N
NUMBER PROCESSORS=4
Y
ALL 4 PROCESSORS BUSY
N
Y
WAIT 10 MILLISECONDS
A
B
C

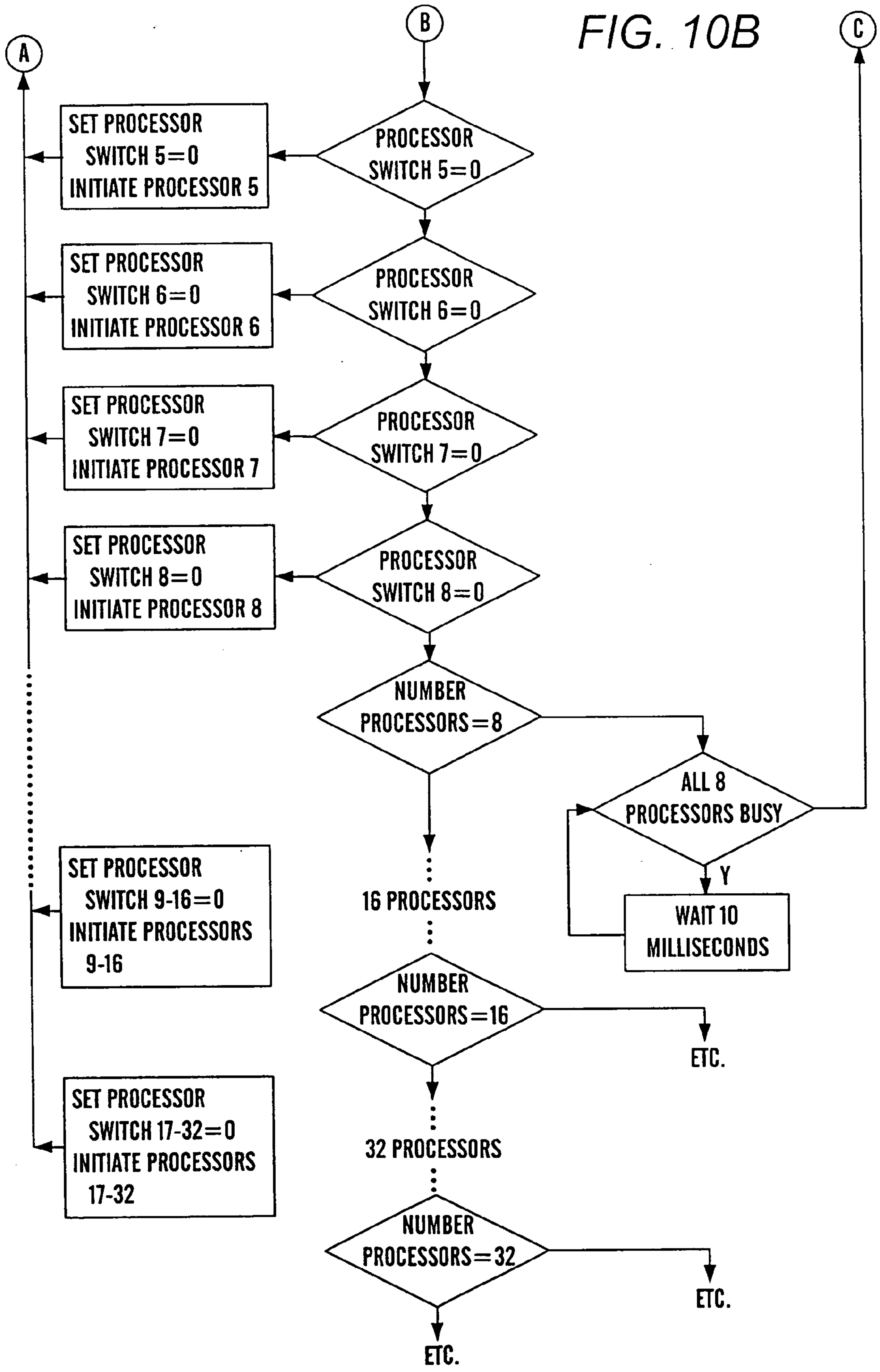
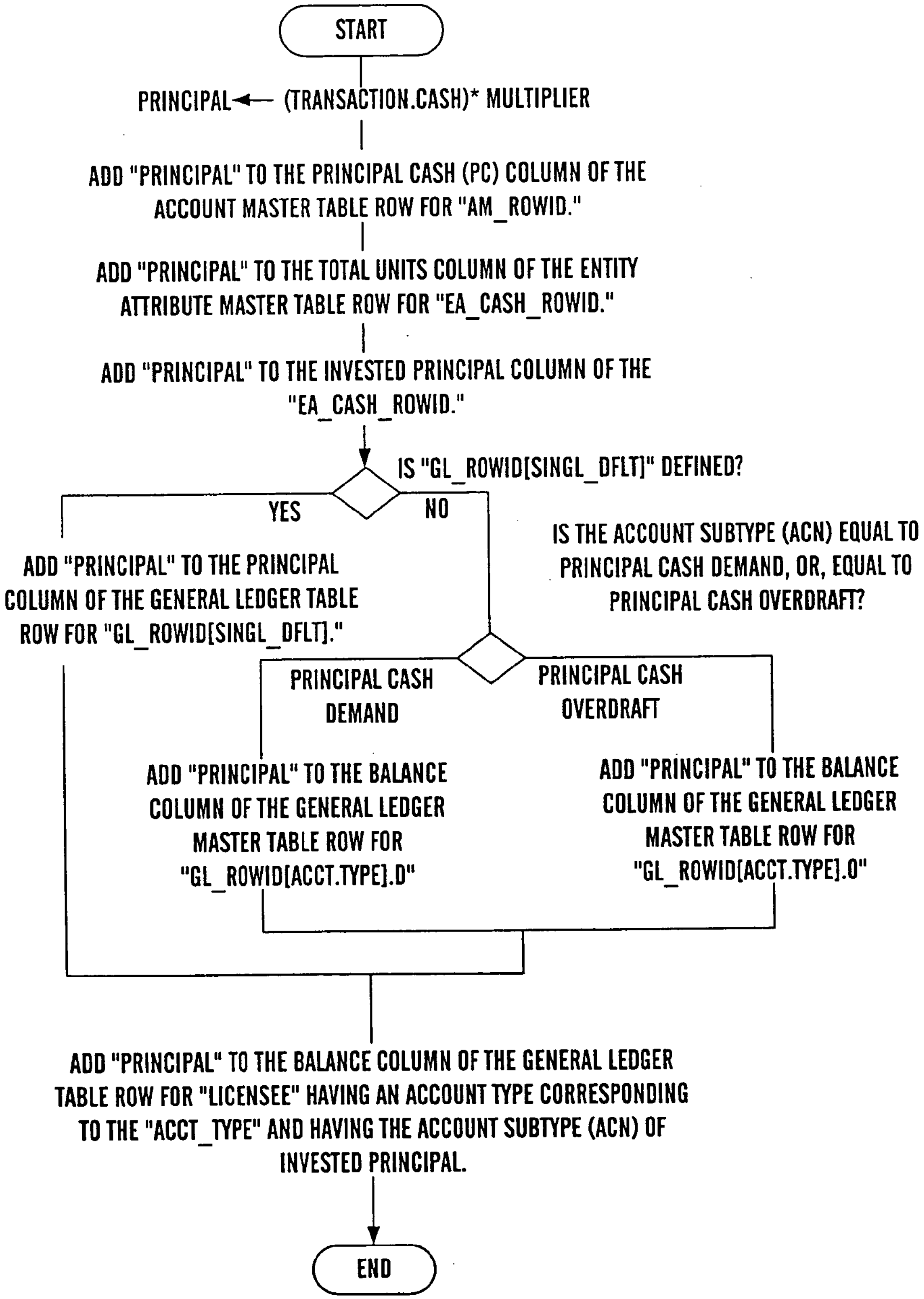


FIG. 10B
B
A
C
PROCESSOR SWITCH 5=0
SET PROCESSOR SWITCH 5=0 INITIATE PROCESSOR 5
PROCESSOR SWITCH 6=0
SET PROCESSOR SWITCH 6=0 INITIATE PROCESSOR 6
PROCESSOR SWITCH 7=0
SET PROCESSOR SWITCH 7=0 INITIATE PROCESSOR 7
PROCESSOR SWITCH 8=0
SET PROCESSOR SWITCH 8=0 INITIATE PROCESSOR 8
NUMBER PROCESSORS=8
ALL 8 PROCESSORS BUSY
Y
WAIT 10 MILLISECONDS
16 PROCESSORS
SET PROCESSOR SWITCH 9-16=0 INITIATE PROCESSORS 9-16
NUMBER PROCESSORS=16
ETC.
32 PROCESSORS
SET PROCESSOR SWITCH 17-32=0 INITIATE PROCESSORS 17-32
NUMBER PROCESSORS=32
ETC.
ETC.

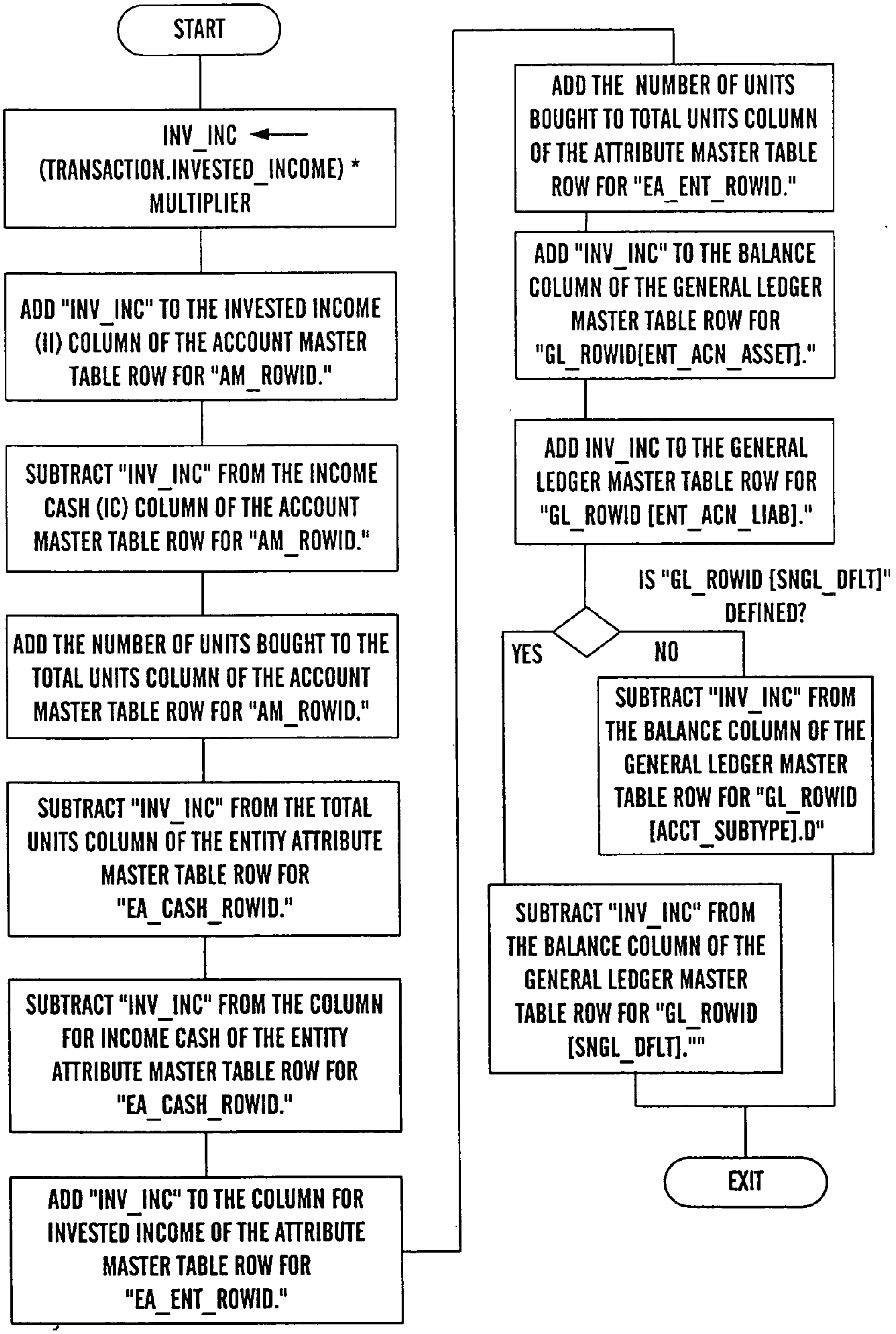
FIG. 11

PROCESS_PRINCIPAL_CASH


START
PRINCIPAL◄— (TRANSACTION.CASH)* MULTIPLIER
ADD "PRINCIPAL" TO THE PRINCIPAL CASH (PC) COLUMN OF THE ACCOUNT MASTER TABLE ROW FOR "AM_ROWID."
ADD "PRINCIPAL" TO THE TOTAL UNITS COLUMN OF THE ENTITY ATTRIBUTE MASTER TABLE ROW FOR "EA_CASH_ROWID."
ADD "PRINCIPAL" TO THE INVESTED PRINCIPAL COLUMN OF THE "EA_CASH_ROWID."
IS "GL_ROWID[SINGL_DFLT]" DEFINED?
YES
NO
ADD "PRINCIPAL" TO THE PRINCIPAL COLUMN OF THE GENERAL LEDGER TABLE ROW FOR "GL_ROWID[SINGL_DFLT]."
IS THE ACCOUNT SUBTYPE (ACN) EQUAL TO PRINCIPAL CASH DEMAND, OR, EQUAL TO PRINCIPAL CASH OVERDRAFT?
PRINCIPAL CASH DEMAND
PRINCIPAL CASH OVERDRAFT
ADD "PRINCIPAL" TO THE BALANCE COLUMN OF THE GENERAL LEDGER MASTER TABLE ROW FOR "GL_ROWID[ACCT.TYPE].D"
ADD "PRINCIPAL" TO THE BALANCE COLUMN OF THE GENERAL LEDGER MASTER TABLE ROW FOR "GL_ROWID[ACCT.TYPE].O"
ADD "PRINCIPAL" TO THE BALANCE COLUMN OF THE GENERAL LEDGER TABLE ROW FOR "LICENSEE" HAVING AN ACCOUNT TYPE CORRESPONDING TO THE "ACCT_TYPE" AND HAVING THE ACCOUNT SUBTYPE (ACN) OF INVESTED PRINCIPAL.
END

PROCESS_INVESTED_INCOME

FIG. 12


START
INV_INC ← (TRANSACTION.INVESTED_INCOME) * MULTIPLIER
ADD "INV_INC" TO THE INVESTED INCOME (II) COLUMN OF THE ACCOUNT MASTER TABLE ROW FOR "AM_ROWID."
SUBTRACT "INV_INC" FROM THE INCOME CASH (IC) COLUMN OF THE ACCOUNT MASTER TABLE ROW FOR "AM_ROWID."
ADD THE NUMBER OF UNITS BOUGHT TO THE TOTAL UNITS COLUMN OF THE ACCOUNT MASTER TABLE ROW FOR "AM_ROWID."
SUBTRACT "INV_INC" FROM THE TOTAL UNITS COLUMN OF THE ENTITY ATTRIBUTE MASTER TABLE ROW FOR "EA_CASH_ROWID."
SUBTRACT "INV_INC" FROM THE COLUMN FOR INCOME CASH OF THE ENTITY ATTRIBUTE MASTER TABLE ROW FOR "EA_CASH_ROWID."
ADD "INV_INC" TO THE COLUMN FOR INVESTED INCOME OF THE ATTRIBUTE MASTER TABLE ROW FOR "EA_ENT_ROWID."
ADD THE NUMBER OF UNITS BOUGHT TO TOTAL UNITS COLUMN OF THE ATTRIBUTE MASTER TABLE ROW FOR "EA_ENT_ROWID."
ADD "INV_INC" TO THE BALANCE COLUMN OF THE GENERAL LEDGER MASTER TABLE ROW FOR "GL_ROWID[ENT_ACN_ASSET]."
ADD INV_INC TO THE GENERAL LEDGER MASTER TABLE ROW FOR "GL_ROWID [ENT_ACN_LIAB]."
IS "GL_ROWID [SNGL_DFLT]" DEFINED?
YES
NO
SUBTRACT "INV_INC" FROM THE BALANCE COLUMN OF THE GENERAL LEDGER MASTER TABLE ROW FOR "GL_ROWID [ACCT_SUBTYPE].D"
SUBTRACT "INV_INC" FROM THE BALANCE COLUMN OF THE GENERAL LEDGER MASTER TABLE ROW FOR "GL_ROWID [SNGL_DFLT].""
EXIT

PROCESS_INVESTED_PRINCIPAL

FIG. 13

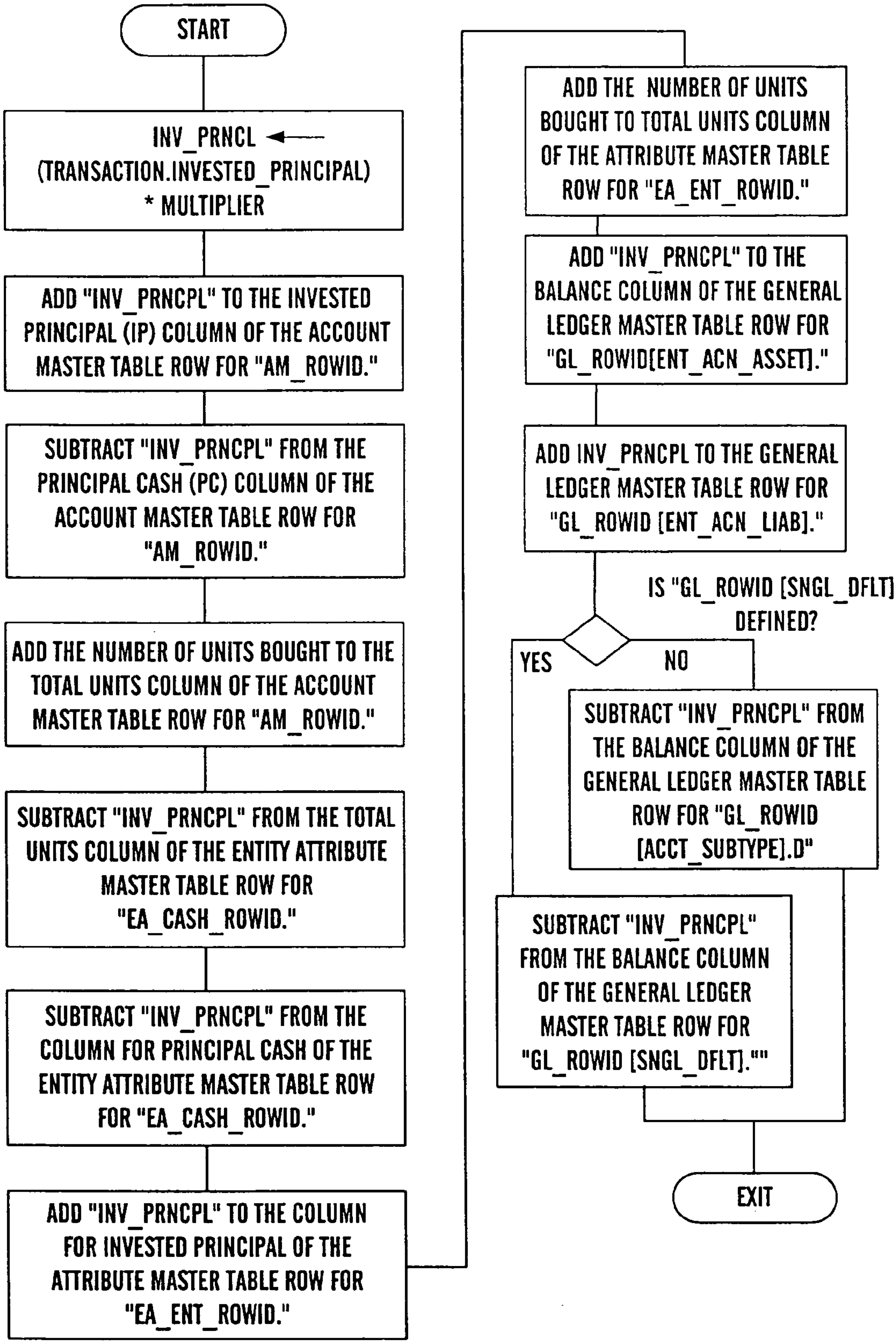

START
INV_PRNCL ← (TRANSACTION.INVESTED_PRINCIPAL) * MULTIPLIER
ADD "INV_PRNCPL" TO THE INVESTED PRINCIPAL (IP) COLUMN OF THE ACCOUNT MASTER TABLE ROW FOR "AM_ROWID."
SUBTRACT "INV_PRNCPL" FROM THE PRINCIPAL CASH (PC) COLUMN OF THE ACCOUNT MASTER TABLE ROW FOR "AM_ROWID."
ADD THE NUMBER OF UNITS BOUGHT TO THE TOTAL UNITS COLUMN OF THE ACCOUNT MASTER TABLE ROW FOR "AM_ROWID."
SUBTRACT "INV_PRNCPL" FROM THE TOTAL UNITS COLUMN OF THE ENTITY ATTRIBUTE MASTER TABLE ROW FOR "EA_CASH_ROWID."
SUBTRACT "INV_PRNCPL" FROM THE COLUMN FOR PRINCIPAL CASH OF THE ENTITY ATTRIBUTE MASTER TABLE ROW FOR "EA_CASH_ROWID."
ADD "INV_PRNCPL" TO THE COLUMN FOR INVESTED PRINCIPAL OF THE ATTRIBUTE MASTER TABLE ROW FOR "EA_ENT_ROWID."
ADD THE NUMBER OF UNITS BOUGHT TO TOTAL UNITS COLUMN OF THE ATTRIBUTE MASTER TABLE ROW FOR "EA_ENT_ROWID."
ADD "INV_PRNCPL" TO THE BALANCE COLUMN OF THE GENERAL LEDGER MASTER TABLE ROW FOR "GL_ROWID[ENT_ACN_ASSET]."
ADD INV_PRNCPL TO THE GENERAL LEDGER MASTER TABLE ROW FOR "GL_ROWID [ENT_ACN_LIAB]."
IS "GL_ROWID [SNGL_DFLT]" DEFINED?
YES
NO
SUBTRACT "INV_PRNCPL" FROM THE BALANCE COLUMN OF THE GENERAL LEDGER MASTER TABLE ROW FOR "GL_ROWID [ACCT_SUBTYPE].D"
SUBTRACT "INV_PRNCPL" FROM THE BALANCE COLUMN OF THE GENERAL LEDGER MASTER TABLE ROW FOR "GL_ROWID [SNGL_DFLT].""
EXIT

FIG. 14

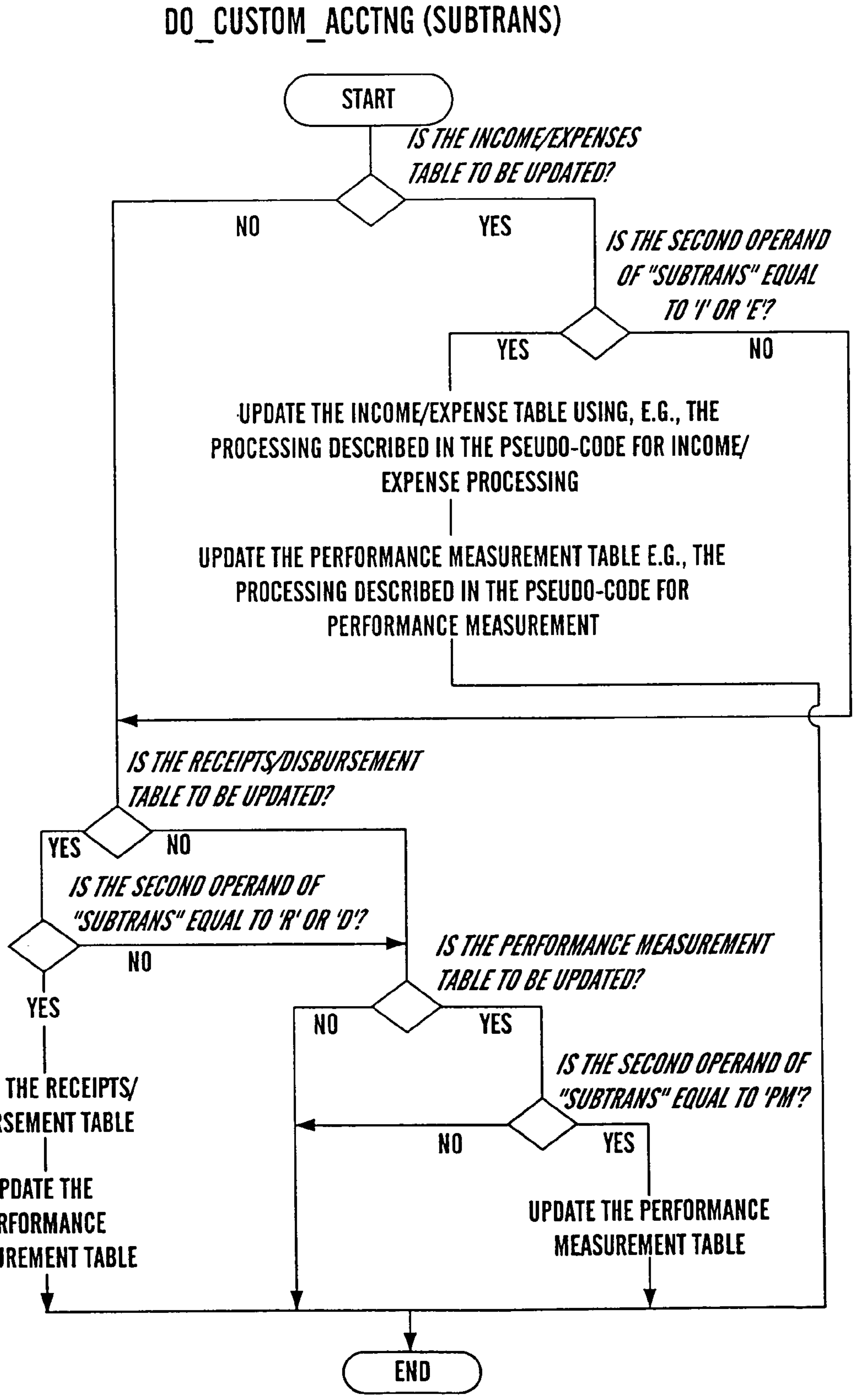
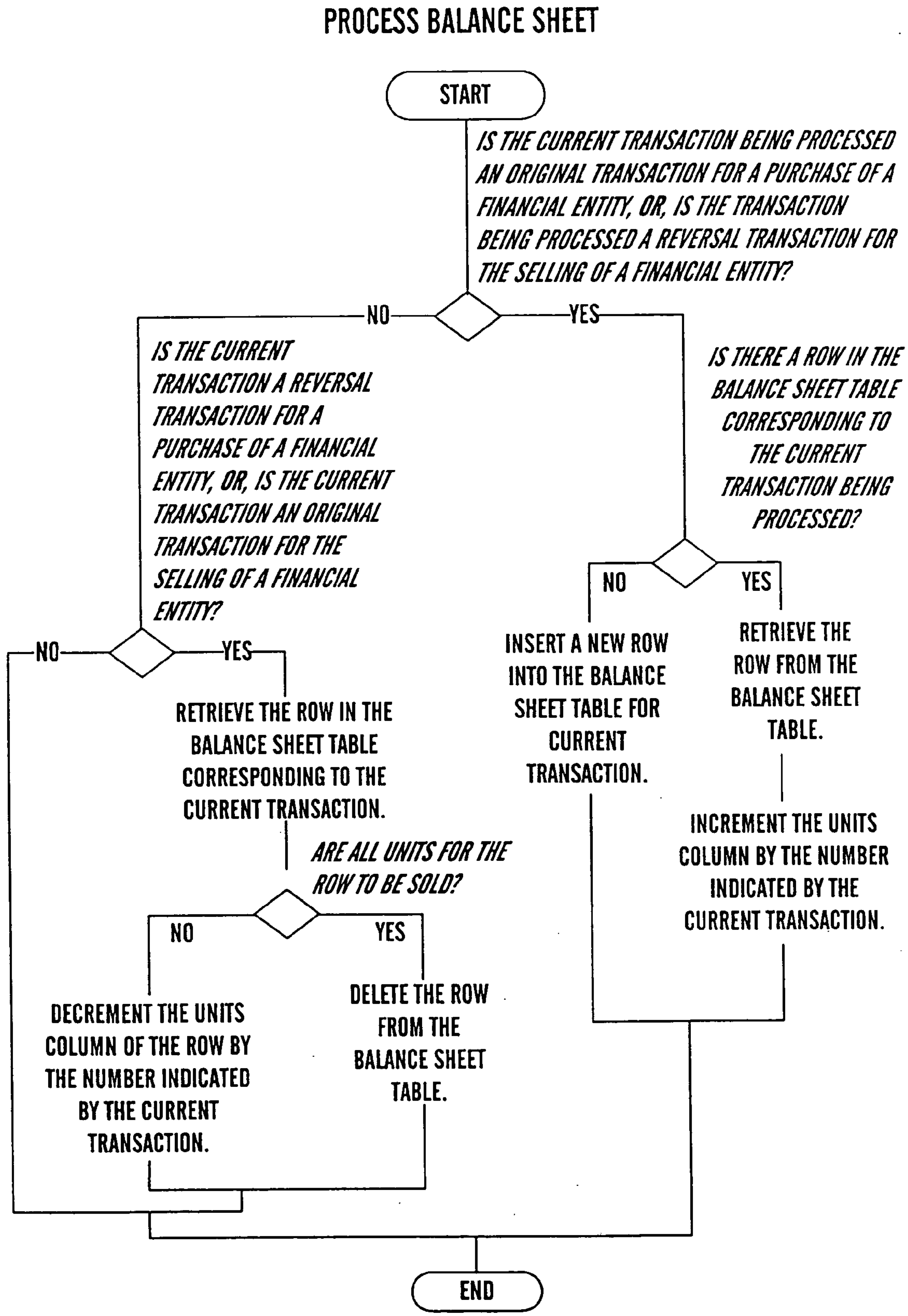

DO_CUSTOM_ACCTNG (SUBTRANS)
START
IS THE INCOME/EXPENSES TABLE TO BE UPDATED?
NO
YES
IS THE SECOND OPERAND OF "SUBTRANS" EQUAL TO 'I' OR 'E'?
YES
NO
UPDATE THE INCOME/EXPENSE TABLE USING, E.G., THE PROCESSING DESCRIBED IN THE PSEUDO-CODE FOR INCOME/ EXPENSE PROCESSING
UPDATE THE PERFORMANCE MEASUREMENT TABLE E.G., THE PROCESSING DESCRIBED IN THE PSEUDO-CODE FOR PERFORMANCE MEASUREMENT
IS THE RECEIPTS/DISBURSEMENT TABLE TO BE UPDATED?
YES
NO
IS THE SECOND OPERAND OF "SUBTRANS" EQUAL TO 'R' OR 'D'?
NO
YES
IS THE PERFORMANCE MEASUREMENT TABLE TO BE UPDATED?
NO
YES
IS THE SECOND OPERAND OF "SUBTRANS" EQUAL TO 'PM'?
NO
YES
UPDATE THE RECEIPTS/ DISBURSEMENT TABLE
UPDATE THE PERFORMANCE MEASUREMENT TABLE
UPDATE THE PERFORMANCE MEASUREMENT TABLE
END

FIG. 15


PROCESS BALANCE SHEET
START
IS THE CURRENT TRANSACTION BEING PROCESSED AN ORIGINAL TRANSACTION FOR A PURCHASE OF A FINANCIAL ENTITY, OR, IS THE TRANSACTION BEING PROCESSED A REVERSAL TRANSACTION FOR THE SELLING OF A FINANCIAL ENTITY?
NO
YES
IS THE CURRENT TRANSACTION A REVERSAL TRANSACTION FOR A PURCHASE OF A FINANCIAL ENTITY, OR, IS THE CURRENT TRANSACTION AN ORIGINAL TRANSACTION FOR THE SELLING OF A FINANCIAL ENTITY?
IS THERE A ROW IN THE BALANCE SHEET TABLE CORRESPONDING TO THE CURRENT TRANSACTION BEING PROCESSED?
NO
YES
INSERT A NEW ROW INTO THE BALANCE SHEET TABLE FOR CURRENT TRANSACTION.
RETRIEVE THE ROW FROM THE BALANCE SHEET TABLE.
INCREMENT THE UNITS COLUMN BY THE NUMBER INDICATED BY THE CURRENT TRANSACTION.
NO
YES
RETRIEVE THE ROW IN THE BALANCE SHEET TABLE CORRESPONDING TO THE CURRENT TRANSACTION.
ARE ALL UNITS FOR THE ROW TO BE SOLD?
NO
YES
DECREMENT THE UNITS COLUMN OF THE ROW BY THE NUMBER INDICATED BY THE CURRENT TRANSACTION.
DELETE THE ROW FROM THE BALANCE SHEET TABLE.
END

MULTI-PROCESSING FINANCIAL TRANSACTION PROCESSING SYSTEM

FIELD OF THE INVENTION

[0001] The present invention relates to a financial transaction processing system, and in particular, to such a system that is capable of decomposing transactions into subtransactions and multi-processing subtransactions simultaneously.

BACKGROUND OF THE INVENTION

[0002] Computerized data processing systems for processing financial transactions have become increasingly more complex as further strides toward automation have occurred. Such complexity has generated a number of related difficulties for the financial data processing industry. In particular, complex financial transaction processing systems may have subtle programming defects or errors that may go unnoticed for long periods of time before the extent of the problems thereby generated are fully recognized. For example, the number of positions allotted for the dating of transactions has recently been problematic, wherein the dates for the millennium starting at the year 2000 can be problematic for many financial transaction processing systems.

[0003] In addition, such complex financial transaction processing systems also are typically incapable of being fully audited. That is, it is common practice in the financial data processing industry to provide only partial auditability in that it is generally believed that the amount of data required to be stored for full auditability is so large as to not be cost effective.

[0004] Further, in many circumstances, the rate of transaction increase is becoming problematic in that progressively larger computers are required for processing financial transactions at an acceptable rate. This problem is exacerbated by the fact that such transaction processing systems are not architected for use on multi-processing machines having a plurality of processors. Thus, the advantages of parallel-processing computers cannot be fully utilized by such systems.

[0005] Accordingly, it would be advantageous to have a financial transaction processing system that alleviates the above difficulties, and that additionally, provides flexibility to adapt to the changing business needs of business enterprises so that the transactions processed and the respective reports generated may be modified easily according to business constraints and demands.

SUMMARY OF THE INVENTION

[0006] The present invention is a financial transaction processing system that achieves substantial increases in auditability and processing efficiency. In particular, the present invention provides auditable trails or history in a number of different ways. For example, financial data within transactions is used in the present invention to update various control fields in different tables or files so that cross-checks of system financial integrity can be performed for assuring that, for example, cash fields, total units fields, and cost fields balance appropriately across system data tables provided by the present invention. Additionally, the present invention provides a full range of auditable history files for each system data table having information that is required during auditing.

[0007] The present invention also performs financial transaction processing using a novel computational paradigm. That is, the financial transaction processing system of the present invention has an architecture wherein financial transactions can be decomposed into corresponding collections of independent subtransactions, such that for each input transaction, the corresponding collection of subtransactions are performed by operations that are independent of one another. Thus, the subtransactions can be performed in any order, including in an overlapping fashion, such as may occur during multiprocessing of these subtransactions on a computer having multiple processors.

[0008] Further, note that each of the subtransactions is described by a relatively short (e.g., less than 8 characters) text string that can be straightforwardly interpreted as an operation (e.g., either plus or minus) together with a series of operands, in particular, a first operand having a value to be used in modifying a data table field (column) specified by a second operand. Such high level descriptions of subtransactions provide both compact conceptualization and a reduction in the total size of the executable code for the present invention. Accordingly, when one of the subtransactions is performed, not only is its corresponding operation performed on the operands, but additionally, control fields such as those mentioned above are updated appropriately in various data tables for the present invention to enhance auditability of the financial data resulting from the transaction processing. Further, note that since the subtransactions are independent of one another and their executable code is relatively small, there is no need for lengthy and complex flow of control transaction processing modules. That is, the size of the code for the present invention may be up to 100 times smaller than many prior art transaction processing systems. Accordingly, this has a substantial positive impact on the efficiency of the present invention in that the swapping of program elements in and out of primary computer memory is substantially reduced.

[0009] In another aspect of the present invention, the financial transactions of a plurality of business enterprises can be processed in an interleaved manner. In particular, since the present invention is substantially data driven, including the descriptions of the transactions and their related subtransactions, the present invention can be easily modified to incorporate both different or updated versions of transactions and associated data tables for an existing business enterprise (e.g., also denoted "licensee" hereinafter). Additionally, the transactions and related data tables for an entirely new or different business enterprise (licensee) may be straightforwardly incorporated into the present invention so that its transactions can be interleaved with the transactions of other business enterprises. Thus, transaction processing may be performed by the present invention for business enterprises having different transactions, different account record structures and differently organized general ledgers substantially without modifying the program elements of the transaction processing system.

[0010] For example, the present invention can be used to simultaneously process transactions for:

[0011] (1) a single software application such as an investment management or telecommunications billing system,

[0012] (2) multiple disparate software applications such as investment management, and telecommunications billing, paying agencies, etc., all with disparate definitions.

[0013] Accordingly, the present invention may be viewed as a software engine, or a user-definable transaction processing tool that can be adapted to a variety of industry specific software application needs without changing the actual program code. That is, by surrounding the present invention with application specific software for inputting transaction data to the multi-processing financial transaction processor of the present invention and retrieving data from the multi-processing financial transaction processor of the present invention, a particular business enterprise can have substantially all of its financial records in condition for auditing on a daily or weekly basis.

[0014] The present invention may be further characterized along the following dimensions: flexibility, auditability, multiprocessing, efficiency and size, these dimensions being discussed, in turn, hereinbelow.

[0015] Flexibility is achieved by permitting a business enterprise to define:

[0016] (1) a series of "reference" tables (also denoted "master tables") that describe the appropriate management decision-making, accounting structure, and regulatory information for the specific application;

[0017] (2) a series of audit controls and system procedures that provide for complete control of all processing and prevent the overwriting of any original data;

[0018] (3) a series of institutional and customer reporting files, known as the "driven" tables; and

[0019] (4) the specific processing content of each individual transaction to be processed via a series of table definitions, known as the "driving" tables.

[0020] Thus, transactions may be customized according to the business needs of a business enterprise.

[0021] Auditability is achieved by:

[0022] (1) providing separate control columns for cash, units and cost basis (if any) in detail records generated and stored for each financial transaction;

[0023] (2) repeating these three control columns, or variations thereof, in at least three different tables so that subsequent summations of each of the four tables will result in similar balances and thus prove that no critical data has been lost in the course of processing, as one familiar with auditing and financial transactions systems will understand;

[0024] (3) adding appropriate data columns:

[0025] (a) to each reference table or master row for maintaining a history of the effects of add, change and delete commands in a current database as well as an archive database;

[0026] (b) to each original file record (i.e. table row) that represents an add to a current database as well as the periodic archive and purge to a permanent database;

[0027] (c) to tables for retaining transaction processing data representing error identification, error negation and error correction.

[0028] Thus, auditabilty of transaction records is achieved by four sets of files for a specific period. These are: (a) a snapshot of all the reference files at the end of the period; (b) snapshots of a history file for each master table, wherein the corresponding history file (table) contains all changes to the master table during the specific period; (c) a snapshot of all financial transactions for the specific period, and (d) a snapshot of all of the "driven" tables at the end of the period.

[0029] Multiprocessing is achieved by:

[0030] (1) decomposing the processing of the present invention into a series of separate and independent subprocesses that may be simultaneously performed on any number of simultaneous processors, and

[0031] (2) decomposing input transactions into a series of subtransactions that are processed by independent processes, which may be executed in any particular order, with complete auditability.

[0032] For example, multiprocessing can be achieved by allocating the next prescribed subtransaction process to the next available processor.

[0033] Efficiency is achieved by:

[0034] (1) Defining and utilizing only four standard processing models that perform all prescribed functionality and auditability of the present invention. The models are:

[0035] (a) Processing Model 1 provides an architecture for maintaining historical transaction data so that financial changes can be traced through time;

[0036] (b) Processing Model 2 provides an architecture for automatically maintaining data columns such as Units, Debits and Credits for cross checking table sums to assure that the financial records for a business enterprise balance;

[0037] (c) Processing Model 3 provides an architecture for automatically maintaining financial records relating to financial instruments such as stocks, bonds, real estate, etc.; and

[0038] (d) Processing Model 4 provides an architecture for producing a common processing format for maintaining customer and institutional data tables.

[0039] (2) Defining only four primary program modules for controlling functionality of the present invention, these modules being:

[0040] (a) a transaction processing controller module for receiving transactions to be processed, and controlling the processing thereof;

[0041] (b) a preprocessor and decomposer module for determining the validity of a received transaction, assuring that all data tables and rows thereof are available for processing the transaction, and retrieving the appropriate subtransactions data descriptions to be processed;

[**0042**] (c) a subtransaction scheduling module for scheduling instantiations of the subtransaction processing module on each of one or more processors; and

[**0043**] (d) a subtransaction processing module for performing each subtransaction retrieved by the preprocessor and decomposer module.

[**0044**] (3) Utilizing a number of software switches to control which tables within collection of "driven" tables are to be updated when a specific type of transaction is to be processed.

[**0045**] Thus, by providing a small number of processing models, decomposing input transactions, and supplying only the necessary subtransaction descriptions, the reliability of the transaction processing system of the present invention is substantially increased.

[**0046**] The software for the present invention is small in size (both source code and object code) due to the following:

[**0047**] (1) defining business enterprise financial data processing methods, accounting structures, and regulatory definitions as data rather than program code;

[**0048**] (2) reducing the processing content to a series of individual transactions; and

[**0049**] (3) reducing all financial transactions to a collection of subtransactions wherein each subtransaction includes an operator and two or more operands in an 8-character string.

[**0050**] Thus, the financial processing by the present invention may be performed on several transactions at a time, one transaction at a time, or different processors within a multiprocessor context. Or, the subtransactions for a specific transaction may be spread over several simultaneous processors. This means that the business enterprise is afforded a large number of options in tailoring the present invention.

[**0051**] Hence, by defining the accounting structure and processing functionality as data rather than actual program code, the size of the total code required to process a specific industry application may be substantially reduced compared to prior art transaction processing systems. For example, the executable code for the present invention may be less than one megabyte (1MB). Thus, since the secondary cache attached to each processor in multiprocessing personal computer servers can be one megabyte, substantially the entire executable for the present invention can be provided to each processor. Thus, the positive impact on total system efficiency is believed to be substantial in that secondary cache is typically about four times faster than normal cache, so productivity gains of about three-hundred percent would not be unreasonable. In other words, the executable code for the present invention can reside in the secondary cache of each processor, thereby allowing the off-loading of any processing function to any processor with relative ease. Additionally, given that a typical RAM memory for a personal computing devices is 16 megabytes, it is believed that such a device will have the capability to process the back office financial transactions of a major money center financial institution or communications billing system.

[**0052**] Additional features and benefits of the invention will become evident from the detailed description and the accompanying drawings contained herein.

BRIEF DESCRIPTION OF THE DRAWINGS

[**0053**] **FIG. 1** is a high level block diagram illustrating the present invention conceptually.

[**0054**] **FIGS. 2A and 2B** is another block diagram of the present invention illustrates: (a) the high level transaction processing modules, and (b) the data tables (represented by the symbols with arcuate vertical sides) provided and maintained by the present invention. Furthermore, the present figure shows the data flows as solid arrows and control flows as dashed arrows. Moreover, this figure also indicates the data tables effected by process models No. 2 and No. 3 of the present invention.

[**0055**] **FIG. 3** is another high level block diagram of the present invention during activation of the preprocessor and decomposer **54** wherein the solid arrows are illustrative of the data flows that occur during the activation of the preprocessor and decomposer **54**. Moreover, the tables within boxes represent tables having a process model No. 1 representation, and the tables having account balancing control fields include the identifier, "CNTLS."

[**0056**] FIGS. **4**-A through **4**-E illustrate the steps of a flowchart for initializing the database tables of the present invention for a new business enterprise licensee that is to have its financial transactions subsequently processed by the present invention.

[**0057**] **FIG. 5** is a block diagram illustrating process model No. 1 of the present invention.

[**0058**] **FIG. 6** is a high level flowchart of the steps of an embodiment of the transaction processing controller **52** of **FIG. 2A**.

[**0059**] FIGS. **7**-A through **7**-D show the high level steps performed by an embodiment of the preprocessor and decomposer **54** of **FIG. 2A**.

[**0060**] FIGS. **8**-A and **8**-B show the steps of a flowchart for obtaining indexes or pointers to particular rows of a general ledger table wherein the rows are used in processing a transaction.

[**0061**] FIGS. **9**-A and **9**-B show the steps for a flowchart of an embodiment of the subtransaction processing module **64 (FIG. 2A)**.

[**0062**] **FIG. 10** is an embodiment of a flowchart of the steps performed for processing income cash transactions by the present invention.

[**0063**] **FIG. 11** is an embodiment of a flowchart of the steps performed for processing principal cash transactions by the present invention.

[**0064**] **FIG. 12** is an embodiment of a flowchart of the steps performed for processing invested income transactions by the present invention.

[**0065**] **FIG. 13** is an embodiment of a flowchart of the steps performed for processing invested principal transactions by the present invention.

[0066] **FIG. 14** is an embodiment of a flowchart of the steps for performing custom accounting such as income expenses, and cash flow for a business enterprise.

[0067] **FIG. 15** is an embodiment of a flowchart of the steps for maintaining a business enterprise's balance sheet related to buys and sells of financial entities or instruments.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT OF THE INVENTION

[0068] **FIG. 1** shows a high level conceptual block diagram of a transaction processing system **50** according to the present invention. In particular, the present invention is conceptualized in the present figure as including five functional components, these being:

[0069] (a) transaction processing controller **52** for: (i) receiving transactions **58** from business enterprises, (ii) controlling the processing of such transactions, including the scheduling of subtransactions to be performed, and (iii) writing of transaction details to, for example, a transaction journal file or table;

[0070] (b) a transaction preprocessor and decomposer **54** for initially receiving a transaction **58** from any one of a plurality of business enterprises as shown, wherein the preprocessor and decomposer **54** decomposes transactions into subtransactions;

[0071] (c) a subtransaction processing module **64** for performing the instructions for each subtransaction determined by the transaction preprocessor and decomposer **54**. In particular, the subtransaction processing module **64** utilizes a collection of subtransaction programmatic data descriptions **66** that can be independently scheduled and performed for processing each transaction **58** provided to the transaction processing system **50**;

[0072] (d) a subtransaction scheduler **62** for scheduling the execution of each subtransaction output by the preprocessor and decomposer **54**;

[0073] (e) a collection of databases **70** containing financial information for each of the one or more business enterprises. Note that the term "database" in the present context includes both the data therein as well as database management functional elements and data structure definitions.

[0074] Another illustration of the present invention is provided in **FIG. 2**. This figure is a block diagram providing both the processing components of **FIG. 1**, and additionally, greater detail is provided of the tables or files within the databases **70**. However, to simplify the discussion hereinafter, the database terminology used will be that of a relational database. Accordingly, files may also be equivalently referred to as tables, records may also equivalently be referred to as rows, and record fields may also be equivalently referred to as columns. Thus, all the data storage symbols having the collective label of **70** are provided within the like numbered databases of **FIG. 1**. It is worth noting, however, that in one embodiment of the present invention, the data tables for distinct business enterprises may be provided in the same collection of tables such as those represented in **FIG. 2**. That is, it is an aspect of the present invention that the accounting and transaction processing of the present invention can use the same plurality of financial data tables for business enterprises having substantially different financial transactions and accounting categories. Thus, although **FIG. 1** illustrates the databases **70** as being distinct for each business enterprise, many of these databases (if not most) may be combined into a single database having a plurality of data tables such as those labeled collectively "**70**" in **FIG. 2**, these tables being discussed in detail hereinafter.

[0075] Referring still to **FIG. 2**, a high level view of the processing performed when processing a transaction **58** is provided. In particular, the transaction processing controller **54** receives an input transaction **58** and invokes the preprocessor and decomposer **54**. The preprocessor and decomposer **54** subsequently performs, for each transaction **58**, the following functions:

[0076] (a) determines, using input from the business enterprise databases **70**, whether all necessary data for performing the transaction is available and otherwise rejects the transaction without performing any portion thereof. In particular, the transaction preprocessor and decomposer **54** determines that all data tables to be accessed are available;

[0077] (b) retrieves the data needed to perform the transaction;

[0078] (c) checks to determine that the transaction operation(s) requested is available, and that the transaction is legitimate to be performed on the data for the input transaction **58**;

[0079] (d) retrieves the subtransaction data descriptors for decomposing the input transaction **58** into subtransactions.

[0080] Accordingly, the preprocessor and decomposer **54** retrieves into the working storage **72** (shown in **FIG. 3**) of a host computer (not shown), upon which the transaction processing system **50** is operating, substantially all data and table rows that are necessary to process the transaction **58**. Additionally, note that as one skilled in the art will understand, if some portion of the required data to process the transaction is unavailable, then the preprocessor and decomposer **54** terminates processing and subsequently writes appropriate error messages and/or details of the transaction into the reject table **74** (**FIG. 2**).

[0081] Assuming that the preprocessor and decomposer **54** successfully performs the gathering of information for the decomposing of the transaction into subtransactions appropriately, then control is returned to the transaction processing controller **52**, wherein this controller then writes the details of the transaction to the transaction journal **78** along with identification data uniquely identifying the transaction (e.g., a transaction sequence number and/or time and date stamp). Following this, the transaction processing controller **52** invokes the subtransaction scheduler **62** for scheduling the performance of each subtransaction by an invocation of the subtransaction processing module **64**. Note that it is an important aspect of the present invention that since the subtransactions can be processed independently of one another for a given transaction, instantiations of the subtransaction processing module **64** can be executed in substantially any desired order. In particular, such instantiations of the subtransaction processing module **64** can be performed concurrently, thus providing a substantial increase in transaction processing efficiency when such concurrency is provided on a computer having a plurality of processors.

[0082] Given that a subtransaction is performed successfully by the subtransaction processing module **64**, various accounting tables within the transaction processing system **50** are updated. In general, each subtransaction conceptually indicates a single operation of either plus or minus that is to be performed with two operands also indicated in the subtransaction. That is, the first operand indicates the data to be added or subtracted from a particular field or column of a table row identified by the second operand. Additionally, each subtransaction updates other tables within the transaction processing system **50** automatically in order to provide consistency among the data tables so that: (a) substantially on-line account balancing capabilities can be performed, and (b) full auditability of the records of the business enterprise providing the transaction can be facilitated by retaining history records of table updates, as will be discussed with reference to "master table transaction cluster processing" described hereinbelow. Accordingly, each subtransaction processed by an instantiation of the subtransaction processing module **64** may update a plurality of the data tables contained in the collectively labeled database **70**. Note that for one skilled in the art of transaction data processing and accounting, the names provided to the tables are indicative of their information content and structure. However, for clarity, substantially all of the tables for the present invention will be discussed in detail and/or illustrated hereinbelow.

[0083] The subtransaction processing module **64** processes subtransactions derived from three general categories of transactions that may be input to the present invention. That is, there may be input transactions for each of the following types of financial transactions (1.1) through (1.3) hereinbelow.

[0084] (1.1) Transactions related to exchanges of funds such as cash debits and credits for accounts of a particular business enterprise are provided. At a high level, the tables related to this functionality include the account master table **84** (**FIG. 2**), the general ledger table **88**, and the entity attribute master table **92**.

[0085] (1.2) Transactions related to additional or customized accounting for clients having accounts in the account master table **84** are provided. For example, in addition to providing the functionality of the transactions described in (1.1) immediately above, a customer income statement (income/expense) table **96** may be provided with client account and transaction information related to income and expenses for tax purposes. Additionally, a customer cash flow (receipts/disbursements) table **100** is also provided for recording any account transaction information related to receipts and disbursements in client accounts. Further, a customer performance measurement table **104** is also provided for retaining client account performance information related to the performance of client portfolios in comparison to investment indexes such as the Dow Jones Industrial Average, the S&P 500, etc. Note that these tables will be discussed and/or illustrated hereinbelow.

[0086] (1.3) When transactions are additionally related to financial instruments other than cash, debits and credits, such as portfolio management wherein there is buying and selling of equities, income derived from equities, and trade settlements related thereto. Further, note that these additional capabilities also provide the same degree of flexibility, adaptability and simplicity as provided in relation to the transaction processing capabilities discussed in (1.1) and (1.2) immediately above. That is, financial equity transactions of various types and for various business enterprises may be easily modified and/or added or removed from the transaction processing system **50** of the present invention, since these transactions are also described by transaction data descriptors consisting of a collection of subtransactions that are capable of being performed in substantially any order that is determined by the subtransaction scheduler **62**.

[0087] Accordingly, in providing the functionality for the transactions related to portfolio management, the preprocessor and decomposer **54**, upon being invoked by the transaction processing controller **52**, also retrieves into working storage (as shown in **FIG. 2**) the necessary data for processing such portfolio maintenance transactions, this data including a subtransaction decomposition for the transaction. Subsequently, as discussed hereinabove, the subtransaction scheduler **62** invokes an instance of the subtransaction processing module **64**. However, in addition to updating any appropriate rows of the tables **84**, **88**, **92**, **96**, **100** and **104**, the subtransaction processing module **64** invokes a portfolio adjuster module **110** for capturing and/or updating detailed data of portfolio transactions that are not otherwise effectively captured for proper accounting and auditing. In particular, for a given subtransaction, the portfolio adjuster **110** invokes one of the following modules (2.1) through (2.4) hereinbelow.

[0088] (2.1) Original add module **114** for processing a subtransaction related to the addition of further financial instruments to a portfolio such as occurs when securities are bought and must be added to a given account.

[0089] (2.2) A reverse of add module **118** for reversing an addition of financial enterprises to a particular account portfolio. Note that this module is typically activated when financial enterprises are inadvertently added to an incorrect portfolio account.

[0090] (2.3) An original sell module **122** for processing subtransactions related to selling financial enterprises within a given account portfolio.

[0091] (2.4) A reversal of original sell module **126** for reversing the affects of an inadvertent sell of financial enterprises within an account portfolio.

[0092] These four modules **114**- **26** update the tables labeled collectively as **70**B. In particular, the processing performed herein and the tables updated herein are described below.

Major Programs and Functionality

[0093] Major Programs

[0094] The N_gine transaction processing system contains four major programs. These are:

[0095] (1) Transaction Processing controller **52**

[0096] (2) Transaction Preprocessor and Decomposer **54**

[0097] (3) Subtransaction Processing module **64**

[0098] (4) Subtransaction Scheduler **62**

[0099] Program Functionality

[0100] The purpose of the Transaction Processing controller **52**

[0101] (a) test for incoming transactions and once detected

[0102] (b) execute the Transaction Preprocessor and Decomposer **54** and then

[0103] (c) execute the Subtransaction Processing module **64** for each transaction.

[0104] The purpose of the Transaction Preprocessor and Decomposer **54** is to verify

[0105] (a) that all information in the transaction is accurate

[0106] (b) that all files and controls are available to properly process the transaction

[0107] (c) that the specific subtransaction processing instructions are loaded into working storage.

[0108] The purpose of the Subtransaction Processing module **64** is to

[0109] (a) execute all of the subtransactions that have been previously defined for a transaction

[0110] (b) create auditability for every transaction.

[0111] The purpose of the Subtransaction Scheduler **62** is to

[0112] (a) allocate a specific task to a specific processor

[0113] (b) return processing to the Transaction Processing controller **52**.

[0114] The present invention may be described as "Table-Driven Transaction Processing". That is, the present invention permits the processing of virtually any type of user-definable transaction by defining the processing for such transactions as data descriptors that are interpreted in real time and dynamically as needed for processing corresponding transactions. Accordingly, the transaction data descriptors are denoted as "driving data" and are defined by the transaction processing master table and the transaction master table. That is, the transaction master table provides a first initial collection of data for identifying each transaction and the transaction processing table provides the remainder of the data including the subtransaction decompositions. Accordingly, each transaction processed updates an appropriate set of user-definable tables (known as the "driven" data) for completing the processing of the transaction. Since both the "driving" and the "driven" information is expressed as data rather that actual code, the entire functionality of the system can be changed in a straightforward manner.

[0115] In the description hereinbelow, the functional components of the present invention are also identified by other naming conventions from the description above. Accordingly, the following table shows the pairing of the functional component identifications above with those also used below:

ABOVE	BELOW
TRANSACTION PROCESSING CONTROLLER 52	N_GINE COMMAND PROCESSOR
TRANSACTION PREPROCESSOR AND DECOMPOSER 54	N_GINE EDIT PROCESSOR
SUBTRANSACTION PROCESSING MODULE 64	N_GINE POSTING TO AM, EA AND GL
SUBTRANSACTION SCHEDULER 62	N_GINE SCHEDULER
PORTFOLIO ADJUSTER 110	AORS
ORIGINAL ADD MODULE 114	ORIGINATE ADD PROCESSING
REVERSER OF ADD MODULE 118	REVERSE ADD PROCESSING
ORIGINAL SELL MODULE 122	ORIGINATE SELL ROUTINE
REVERSE OF ORIGINAL SELL MODULE 126	REVERSER SUBTRACT PROCESS

N_gine System Design Rules

[0116] A. The Magic Number in Software Design is **1**. That is,

[0117] store data once,

[0118] program data once,

[0119] process data once.

[0120] B. Design a total system with the fewest number of processing models. For example,

[0121] One model for processing all adds (inserts), changes (updates), and deletes (deletes) for all Master (or Reference) Files (or tables).

[0122] Namely,

[0123] Begin Time

[0124] Number of Transactions

[0125] Number of Acceptances

[0126] Number of Rejects

[0127] End Time.

[0128] These variables represent the only true means of measuring actual productivity.

[0129] F. For reasons of auditability, never overwrite any original information. Move all original information from data entry (cradle) to data warehouse (grave) without any changes.

[0130] G. For reasons of reliability and profitability, system designs should focus on a "large number of small programs" rather than a "small number of large programs". The result is not only ease of maintenance but also the ability to spread the small programs across a number of simultaneous processors.

[0131] H. For reasons of manageability, all system designs should embrace one integrated enterprise-wide standard naming convention for all files (tables), records (rows), and fields (columns).

[0132] I. For reasons of portability, use the fewest number of language commands to code the system. Avoid vendor and/or language extensions.

[0133] J. For reasons of flexibility, never hard code what can be table-driven.

N_gine Design Concepts

[0134] A. Only 4 Processing Models for Financial Services and Telecommunications Applications

[0135] 1. Schema

[0136] 2. Units, Debit/Credit

[0137] 3. Assets/Liabilities

[0138] 4. File Maintenance Routine

[0139] B. Table-Driven Transaction Processing for maximum flexibility

[0140] 1. Number of Transactions

[0141] 2. Name of Each Transaction and Unique Details

[0142] 3. Processing Algorithms (at least 1, up to 20 depending upon complexity)

[0143] 4. Each algorithm has 3 components

[0144] a. Plus (P) or Minus (M)

[0145] b. Operand 1

[0146] c. Operand 2

[0147] C. 100% Auditability For Every Transaction by creating

[0148] 1. a Detail Record containing all relevant data and

[0149] 2. hash totals of three relevant fields in at least 3 other tables.

[0150] D. The 3 relevant fields for calculating all hash totals are:

[0151] 1. Cash

[0152] 2. Units

[0153] 3. Cost Basis

[0154] E. Basic Relational Database Management System Processing Concepts

[0155] 1. Commit/Rollback

[0156] 2. Row Level Locking

[0157] 3. Indexing, ROWID

[0158] 4. Stored Procedures

[0159] 5. Shared Memory

[0160] F. Some Financial Services Accounting Systems are not permitted to commingle funds. That is, separate accounting for both income and principal must be provided. Therefore, each account master must have a designated "income posting code" to define the proper processing. Such a code might be: (I) Income Only, (P) Principal Only, (B) Both Income and Principal.

N_gine's Basic Tables

[0161] Licensee Profile (The Licensee "Reference" or "Master" Tables)

[0162] LM The License Master table contains the necessary information to process any type of licensee using either single or multiprocessing computers.

[0163] LU The Licensee User Master identifies different users for the disparate systems that may be processed simultaneously.

[0164] LT The Licensee Account Type table contains the necessary information to process any type of account be it for a pension trust account, a communications account, or a corporate subsidiary.

[0165] LD The Licensee Default Definition table the default definitions for cash, units, and cost basis controls for total system control.

[0166] LL The Licensee General Ledger Definition is a list of all of the acceptable entries for the General Ledger. That is, it provides a framework for processing any type of accounting controls for any set of account types.

[0167] LS The Licensee Diversification Scheme contains a three level classification scheme for reporting an decision-making purposes for any set of assets and liabilities.

[0168] LP The Performance Measurement Group Master contains a three level classification scheme for measuring the performance of different investment groups.

[0169] LN The Licensee Summary Name Master contains a list of the entries on any type of Income Statement and Cash Flow Statement.

[0170] LW The Licensee Wholesaler Master contains name, address, sales volumes, etc. wholesalers of communications services.

[0171] LR The Licensee Reseller Master contains name, address, sales volumes, etc. for resellers of communications services.

[0172] Account Profile (The Customer "Reference" Tables)

[0173] AO The Account Objectives Table contains the different types of account objectives, such as income, growth, capital preservation, etc.

[0174] AL The Account Jurisdiction contains the different types of legal relationships, such as broker, agent, trustee, advisor, etc.

[0175] AJ The Account Jurisdiction contains the different types of legal jurisdiction, such as federal law, state law, foreign law, etc.

[0176] AR The Account Representatives Table houses the different representatives, their names and communication addresses.

[0177] AN The Account Registration Names is a list of legal names used in security settlement.

[0178] AM The Account Master table provides all of the necessary information to process any type of account by linking the Account Objective, Account Jurisdiction, Legal Capacity, Profit Center, Account Representative, and Registration tables plus other relevant data for reporting content and reporting cycles.

[0179] AC The Account Communications Links links the Account Number for Financial Services to the account numbers for communications services so that all information can be contained in one reporting scheme.

Transaction Profile (The "Driving" Tables)

[0180] TM The Transaction Master table provides all of the information to process any type of transaction, excepting the specific processing algorithms.

[0181] TP The Transaction Processing table provides all of the specific processing algorithms for any type of transaction master. The Transaction Master and Transaction Processing tables provide all of the necessary information to process any type of transaction.

[0182] TR The Transactions—Recurring Table (TR) contains the necessary information for automatically processing any type of transaction on a recurring basis.

Entity Profile (The Entity "Reference" Tables)

[0183] EM The Entity Master table provides all of the necessary information to process any type of financial entity.

[0184] EA The Entity Attribute table joins all relevant diversification (known as type, group, and class), general ledger (known as accounting control numbers), and performance group (known as type, group, and class) data into one table for only one access seek.

[0185] ET The Entity Transaction table links specific transactions to specific entities, such as BG (Buy Government) for a US Treasury Note, BF (Buy Tax-Free) for a tax-free bond, BE (Buy Equity) for common stocks, etc. Note: It is the correct assignment of such transactions to such entities that permits the proper accumulation of data for income tax purposes.

Licensee Status

[0186] SG The System General Ledger contains all of the information to process any type of institutional accounting control.

[0187] SJ The System Transaction Journal Table contains all of the transactions and all of the details for each transaction for a specific accounting period.

[0188] ST The System Trade Settlement Table contains all of the automatically generated offset transactions for Buys and Sells

[0189] SS The System Summary Table contains a record for each execution of the system with the Begin Time, End Time, Number of Total Records Read, Number of Accepts, Number of Rejects, etc.

[0190] SR The System Reject Table contains a list of all transactions rejected for whatever reason.

[0191] SC The System Transaction Count Table contains the number of each type of transaction processed on any given transaction.

Customer Status (The "Driven" Tables)

[0192] CS The Customer Income Statement contains all revenues, expenses, and profits or losses for all customer accounts.

[0193] CF The Customer Cash Flow Statement contains all receipts and disbursements for all customer accounts.

[0194] CB The Customer Balance Sheet table contains all assets and liabilities for all customer accounts.

[0195] CG The Customer Capital Gains table contains all of the realized capital gain details for all customer accounts.

[0196] CI The Pending Income table contains all of the pending income, such as interest or dividends, for all accounts.

[0197] CA The Pending Capital Adjustments table contains all of the pending capital adjustments, such as stock splits, stock dividends, mergers, acquisitions, etc., for all accounts.

[0198] CP The Performance Measurement contains all of the periodic performance records for all customer accounts.

The Control Tables (The "System Balance" Tables)

[0199] Since every transaction is recorded in a detail record plus hashed to three other control tables, the control values of cash, units, and cost basis are added to like values in the following control tables:

[0200] Account Master, System General Ledger, and Entity

[0201] Attribute tables.

[0202] For other reports such as the Income Statement and the Cash Flow Statements, the Performance Measurement table is used as a control table instead of the General Ledger.

[0203] The present invention includes four computational processing models (process models 1 through 4) for processing financial transactions and assuring full auditability and traceability.

[0204] The purpose of Process Model 1 (**FIG. 5**) is to create a single methodology for capturing, maintaining, and archiving the non-financial transaction data including a master table (reference table, or schema) data for 100% auditability within a single software system. This model provides:

[0205] A current database **300** (**FIG. 5**)(for additions, negations and corrections) and an archive database **304**(Read Only)

[0206] Eight tables (i.e. tables **312**, **316**, **320**, **324**, **328**, **332**, **336** and **340**, of **FIG. 5**)

[0207] Number of Modifications

[0208] 12 Control Fields per master table

[0209] A sequence number generator

[0210] A process flow methodology for add, change, and delete of data table rows.

[0211] The operation of Process Model 1 is as follows:

[0212] 1) Normal Updating to current database **300**

	Write to Reject	Write to Accept	Move Master to History	Add to Master	Change Master	Delete Master
Add						
IF Identifier Found	X					
IF Identifier Not Found		X		X		
Change						
IF Identifier Not Found	X					
IF Identifier Found		X	X		X	
Delete						
IF Identifier Not Found	X					
IF Identifier Found		X	X			X

[0213] 2) Periodic updating to the archive database **304** at the end of a pre-determined time period. That is,

[0214] (a) archive snapshots of the archive master **312** in the current database **300** to the master in archive database **304**;

[0215] (b) archive the archive history **332** in the current database **300** to the master history **340** in the archive database **304**;

[0216] (c) purge the history table **332** in the current database **304**.

[0217] The purpose of Process Model 2 (**FIGS. 2A, 2B**) is to create a single methodology for: capturing, maintaining, and archiving the financial transaction data including: units, and debit/credits for one or more disparate financial applications with 100% auditability, wherein the processing is performed by: (a) computing configurations containing any number of simultaneous processors, (b) decomposing each input financial transaction into separate and independent subcomponents, (c) allocating the subcomponents across any number of multiple processors.

[0218] The methodology of process model 2 utilizes a data-driven transaction processing strategy, wherein the manner in which a transaction is processed is determined by retrieving appropriate control data for processing a given input transaction. Thus, the present model provides the ability: (a) to process like systems (such as financial services systems) with different transaction definitions and accounting requirements (such as commercial banking, broker/dealers, mutual funds, insurance systems) and different debits and credits and/or (b) unlike systems (such as telecommunications systems) with disparate definitions (such as landline, wireless, satellite, cable systems) within the present invention at the same time.

[0219] The purpose of Process Model 3 (**FIGS. 2A, 2B**) is to create a single methodology for: capturing, maintaining, and archiving the financial transaction data including: units, debits/credits, financial instruments for one or more disparate financial applications with 100% auditability within a single software system on computing configurations containing any number of simultaneous processors, decomposing each disparate financial transaction into separate and independent subcomponents, allocating the subcomponents across any number of simultaneous processors, and processing the data with 100% auditability. The methodology of Model 3 provides:

[0220] "Detail Record Maintenance", that is, the ability to process transactions for similar business enterprises (such as portfolio management systems) relating to various financial instruments (such as disparate assets and liabilities) and/or transactions for dissimilar business enterprises (such as portfolio management systems, paying agencies, stock transfer systems) with disparate languages (such as English, Spanish, French, or German) and disparate definitions (such as management philosophy, accounting, and operating nomenclature) and unlike financial instruments (such as assets and liabilities) within the same software at the same time.

[0221] The ability to decompose, allocate, process, and audit each financial instrument transactions with 100% auditability.

[0222] The current databases **300** (for additions, negations and corrections) and the archive databases **304**(read only);

[0223] Sixteen data tables (some of which are shown in **FIGS. 2A-2B**) plus a sequence generator;

[0224] 12 control fields appended to the master tables for tracing master table changes;

[0225] One transaction three hash totals (mostly using AM, EA, and PM tables);

[0226] 4 currency fields;

[0227] Sequence number generation;

[0228] Reversing/reversed by detail;

[0229] Processing flow for additions, negations, and corrections.

[0230] The purpose of Process Model 4 is to create a single methodology for performing file maintenance including: creating a record (row) containing the initial data in a file (table) or modifying the initial data within an existing record (row) within a file (table) or deleting a current record (row) from a file (table) in any software application on computing configurations using simultaneous processors. Where the term, "Details", hereinbelow represents the identity of the specific financial transaction, the methodology of the process model 4 is provided by programs such as the following:

```
BEGIN
    IF Trxn is "ADD" then
        /*    Test for Duplicate Add                              */
        SELECT One or More Values from the Desired File (Table) into Working Storage
        IF Error then
            /*    Add New Record                                  */
            INSERT INTO Reject Report
            IF Error then
                Message "INSERT Reject ADD", Details
                Goto Write Reject Table
            ENDIF
        ELSIF
            /*    Increment Existing Record                       */
            Increment One or More Data Values
            UPDATE SET, Details
            IF Error then
                Message "UPDATE Error ADD", Details
                Goto Write Reject Table
            ENDIF
        ENDIF
    ELSIF Trxn is "SUBTRACT" then
        /*    Test for Valid Record                               */
        SELECT One or More Value(s) from Existing Record
        IF Error then
            Message "SELECT Error SUBTRACT", Details
            Goto Write Reject Table
        ENDIF
        /*    Test for Valid Amounts                              */
        IF One or More Amounts > One or More Values from Existing Record then
            INSERT INTO Reject Report
            IF Error then
                Message "INSERT Reject SUBTRACT", Details
                Goto Write Reject Table
            ENDIF
        /*    Delete Existing Record                              */
        ELSIF One or More Amounts = One or More Values from Existing Record
        AND    Special Deletion Criteria = TRUE then
            DELETE Record
            IF Error then
                Message "DELETE Error", Details
                Goto Write Reject Table
            ENDIF
        ELSE
            /*    Decrement Existing Record                       */
            Decrement One or More Values
            UPDATE SET, Details
            IF Error then
                Message "UPDATE Error SUBTRACT", Details
                Goto Write Reject Table
            ENDIF
        ENDIF
    ELSE
            /*    Invalid ADD or SUBTRACT Code                    */
            INSERT INTO Reject Report
            IF Error then
                Message "INSERT Reject AORS", Details
                Goto Write Reject Table
            ENDIF
        ENDIF
```

-continued

```
        Goto EOJ
        <<Write Reject Report>>
        ADD to Reject Table
        IF Error then
            Message "INSERT Reject Table Error", Details
            STOP
        ENDIF
        <<EOJ>>
            Null
    END
```

[0231] Accordingly, the methodology of process model 4 defines:

- [0232] (a) A current database (for additions, negations and corrections) and archive database (Read Only)
- [0233] (b) ADD or SUBTRACT;
- [0234] (c) Initial tests for values;
- [0235] (d) Special deletion criteria;

[0236] (e) Tests for action;

- [0237] INSERT or UPDATE;
- [0238] DELETE or UPDATE;
- [0239] INSERT INTO Reject Tables;

[0240] Processing Model 1:

[0241] Processing model 1 is a method for processing changes to files (or tables) denoted as master or reference tables (files) wherein these tables retain fundamental information that is not derivable from other tables. In particular, processing model 1 processes changes to master tables in an automated manner without losing historical financial information. Accordingly, 100% auditability of all data changes is able to be achieved.

[0242] The method of achieving this goal uses an architecture denoted as "Master Transaction Cluster Processing" (MTCP). MTCP is based on the premise of creating a logical flow of all original information from data capture (data entry) to permanent data repository (data warehouse) by replacing single master files (or tables) with a cluster of files (or tables). Therefore, MTCP addresses the complete life cycle of all information relevant to organizational decision-making. MTCP is targeted for use in the automatic generation of program code for multiple large-scale real-time transaction processing applications (such as securities trading, telecommunications billing, and work management) on multi-processing computers (using 4, 8, 16, 32 processors), where control is not only an increasing complex issue but an absolute necessity for future competition.

[0243] The circumstances leading to the invention of Master Transaction Cluster Processing are:

- [0244] a) Prior art financial transaction software architecture lacks the ability to identify transactions by table, transaction date, transaction number, and the person authorizing the transaction.
- [0245] b) Prior art financial transaction systems typically use only one table to contain all Master Information (i.e., non-derivable information) and the data in this table is overwritten, thereby losing historical information. Cases in point would be a record of all of the past mailing addresses or processing instructions for a specific customer.
- [0246] c) Without 100% retention of an organization's vital information, management has no idea of the accuracy of the information being used for decision-making purposes.
- [0247] d) The Year 2000 problem, know as Y2K, is proving that past software applications designs have reached technological limits and current maintenance costs are inordinately expensive.
- [0248] e) Competitive pressures are mounting for higher quality software with lower software development and maintenance costs. Totally new architectures for applications software is in great demand.
- [0249] f) The ComputerWorld article, "Information: America's Favorite Investment," by Paul Strassman, ComputerWorld Magazine, Aug. 5, 1996, states that over 1100 companies are spending more on automation annually than the net worths of their respective companies.
- [0250] g) The Standish Report as described in Development Patterns, InfoWorld Magazine, Feb. 3, 1997, p. 56, states that the success rate of Business Process Reengineering has increased from 16% in 1994 to only 27% in 1996.

[0251] Note, in the book "Oracle Design", Ensor & Stevenson, O'Reilly Press, it is a recommended practice to compromise data retention rather than achieve 100% auditability. Today's hardware costs suggest otherwise.

[0252] The advantages of the present invention over the approaches discussed above are:

- [0253] to provide 100% auditability which offers business management the capability to exercise its fiduciary responsibility to its stockholders and Board of Directors,
- [0254] to capture, maintain, and ensure the integrity of all vital information for business enterprise decision-making purposes, and
- [0255] to preserve such information consistent with business enterprise-defined data retention cycles. Additionally, the present invention allows accountants to certify in business enterprise annual reports that all vital corporate data is being properly preserved.

[0256] A detailed description of Master Transaction Cluster Processing corresponding to model 1 (the first computational model of the present invention) is as follows.

[0257] MTCP Overview

[0258] Master Transaction Clustering, or MTCP, performs the following tasks:

[0259] a) assigns a unique identifier based on (i) master table identification, (ii) transaction date, (iii) transaction number, and (iv) authorized user, to each transaction that causes a change in the state of a particular record of a master table. That is, if one or more data elements in the record change, then the previous record is written to history, and a new status is assigned to an identifier field used for tracking such changes;

[0260] b) creates a logical flow of data as it is originally entered from its inception (data entry) to its repository (data warehouse). The unique architecture of MTCP replaces the Master File (or Table) within prior art systems with a cluster of Master Files (or Tables), known as a "Master Transaction Cluster". This cluster is suitable for multiprocessing (or the use of simultaneous processors within a single computer to complete a common job). Hence, MTCP addresses 100% auditability via maintaining the total life cycle of information. Aged information may be deleted from the appropriate tables consistent with user-defined data retention policies;

[0261] c) offers a standard for processing all Master Tables within a total application;

[0262] d) provides a test bed for separately testing each Master Table Cluster under development and all Master Table Clusters in concert;

[0263] e) permits management to report that it is successfully capturing, maintaining, and preserving all critical information for decision-making purposes.

[0264] MTCP Scope

[0265] Master Transaction Cluster Processing utilizes the following (**FIG. 5**):

[0266] a) two databases (i.e., the current data base **300** and the archive data base **304**),

[0267] b) sequencing generator **308** having: (i) two external sequence generators; (ii) two internal counters,

[0268] c) eight tables (denoted master table **312**, input table **316**, summary table **320**, reject table **324**, accept table **328**, history table **332**, master archive table **336** and master history table **340**), and

[0269] d) twelve additional fields for every row in the master table **312**.

[0270] MTCP Independence

[0271] Master Transaction Cluster Processing of Model 1 is independent of any:

[0272] a) application—such as accounts receivable, customer billing, etc.

[0273] b) industry—such as financial services, telecommunication, or work management,

[0274] c) hardware manufacturer—such as Compaq, Digital, HP, IBM, NCR, Unisys,

[0275] d) operating system—such as MS-DOS, UNIX, OpenVMS, MVS, etc.

[0276] e) network—such as Novell, Ethernet, etc.

[0277] f) relational database management system—such as Oracle, Sybase, Microsoft SQL Server, Informix, etc., and

[0278] g) computer language—such as SQL, COBOL, FORTRAN, PL/1, Java, etc.

[0279] MTCP Architecture

[0280] The Master Transaction Cluster Processing (MTCP) architecture can be used for any application in any industry using any computer language. Within the typical structured processing scheme of input and process, the Master Transaction Cluster Processing focuses solely on the process function. Thus, the method permits users to define input screens and defined output reports.

[0281] MTCP Databases

[0282] Unlike prior art software system which contain only one table for each set of primary records, Master Transaction Cluster Processing uses eight related tables, or a cluster of tables, to track all information on a cradle to grave basis. The cradle being its point in inception (or data entry), and the grave being its permanent repository (or data warehouse). Consequently, the "Master Transaction Cluster" spans two different databases: one denoted the Current database **300** containing all relevant data for the current processing period and a second denoted the Archive database **304** containing all relevant data for all previous processing periods. The Current database **300** represents the area of high inquiry, and the Archive database **304** represents the area of low inquiry. Consequently, the Current database **300** is normally placed on high-speed internal disk drive and the Archive database **304** is normally placed on less expensive lower-speed CD-ROMs. Note that trailing information in the Archive database **304** may be destroyed consistent with defined data retention policies, statute of limitations, etc.

[0283] MTCP Tables

[0284] The six tables in the Current database **300** are the

[0285] a.) Master Table **312**(M) that will contain all records to be maintained.

[0286] b.) Input Table **316** (I) that will contain all records prior to updating.

[0287] c.) Reject Table **324** (R) that will contain all records rejected during processing.

[0288] d.) Accept Table **328** (A) that will contain all records accepted during processing.

[0289] e.) History Table **332** (H) that contain a complete snapshot of all records prior to updating.

[0290] f.) Summary Table **320** (S) that contains the results of a specific processing operation.

[0291] and the two tables in the Archive database **304** are the:

[0292] g.) Master Archive Table **336** that contains snapshots of the master table **312** at the end of each processing period.

[0293] h.) Master History Table **340** that contains a history of the master table **312** changes during a current processing period.

[0294] Note that the Master Table (M), Input Table (I), Reject Table (R), the Accept Table (A), the History Table (H) in the same "Master Transaction Cluster" share the same number and order of data elements consisting of alphabetic, numeric, and date items. Alternatively, the Summary Table (S) contains the start time, end time, number of accepts, and number of rejects for each time a series of master table **312** modifications are provided.

[0295] MTCP Generator and Counters

[0296] The Generators **308** include two different external counters and two internal counters used in effecting 100% auditability. The two external counters are the Accept Sequence Number Generator and the Reject Sequence Number Generator. The two internal counters are the Total Records Read Counter and the Number of Modifications Counter. All are used only in the Current database **300**, as the Archive database **304** is read-only in nature.

[0297] Regarding the external counters, the Accept Sequence Number Generator included in the Current database **300** automatically generates sequential numbers for the processing period (daily, weekly, monthly, etc.) starting with the number 1, and increments by 1, so that every transaction processed against the preceding (old) master table **312** will receive a specific transaction number, and accordingly, each transaction processed will be uniquely identifiable based on master table identity, transaction date, transaction number, and authorized user. Note that the transaction date is read off the internal system clock. The Reject Sequence Number Generator counts the number of rejects for the specific processing period. Its function is similar to the Accept Sequence Number Generator. Both the Accept Sequence Number Counter and the Reject Sequence Number Counter are "processing period" specific. That is, both are cleared to zero at, e.g., midnight on the end of the processing period so that each processing period may be separately identified and audited.

[0298] Regarding the internal counters, the Total Records Read Counter counts the number of transactions read during a specific processing performance. Since the Total Records Read Counter is "job execution" dependent, this counter is cleared to zero at the outset of every processing program execution. The Number of Modifications Counter counts the number of times a specific record has been changed. As this counter is "record" dependent, this counter is never cleared to zero, This specific counter should identify the number of individual records that may be retrieved, viewed, and

[0299] To achieve 100% auditability of a complete system, every master file (or table in relational database management systems has a Master Transaction Cluster. Therefore, a total system containing 15 tables would require 15×8 or 120 tables to achieve full 100% auditability. Since each table will require at least 4 SQL scripts to (1) Create Table, (2) Select data from the table, (3) Delete data from the table, and (4) Drop the Table in the event of redefinition, the number of SQL scripts is 15×8×4, or 960 SQL Scripts. Then, each Master Transaction Cluster will require at least a Processing Program plus a Review, Reset, and Retest, or at least four more programs for each cluster, or 4×15, or 60, more SQL Scripts. All of the SQL scripts would be stored in one SQL Script Library on the computer for future reference and ease of maintenance.

[0300] MTCP Multi-processing

[0301] The multi-processing of the Master Transaction Cluster occurs in the following manner:

[0302] For additions (or Insertions in SQL) of data

[0303] The Insertions to the Master Table **312** and

[0304] Insertions to the Accept Table **328** may be processed simultaneously.

[0305] For changes (or Updates in SQL) of data

[0306] The Update of the Master Table **312** and the Insert to the Accept Table **328** may be processed simultaneously after the original record from the Master Table **312** has been copied to the History Table **332**.

[0307] For deletes (or Deletes in SQL) of data

[0308] The Deletion from the Master Table **312** and the Insertion to the Accept Table **328** may be processed simultaneously after the current record in the Master Table **312** has been updated for the transaction identifier and then copied to the History Table **332**.

[0309] MTCP Creation

[0310] Before processing any Master Transaction Cluster, the necessary databases and files (or tables) must be created. For each business enterprise utilizing the present invention, these databases and files are created only once in the following manner:

```
(Begin Program)
    Create "Current" database
    Create "Archive" database
    in the "Current" database
        Create Master Table
        Create Input Table
        Create Reject Table
        Create Accept Table
        Create Second Accept Table (on separate
        disk unit, if desired)
        Create History Table
        Create Summary Table
    Create Sequence Number for Accepts
    Create Sequence Number for Rejects
    in the "Archive" database
        Create Master Archive
        Create History Archive
(End of Program)
```

[0311] MTCP Processing

[0312] Processing of the "Master Transaction Cluster" then occurs in the following manner.

[0313] Step 1: All required information for processing a transaction is first captured on an Input Form.

[0314] Step 2: Once this information is edited by, e.g., an operator, an Enter Key can be pressed by an operator to write this information to the Input Table **316** for particular master transaction clusters.

[0315] Step 3: For each input table **316**, a polling program notes that the Input Table is not empty and has a transaction action to be processed whereupon the action is processed by a process (denoted "process 1" in FIG. **M1**).

[0316] Step 4: The transaction processing program determines the type of file maintenance to perform; basically,

[0317] (1) add a record (entitled Insert a Row in SQL),

[0318] (2) change a record (entitled Update a Row in SQL), and

[0319] (3) delete a record (entitled Delete a Row in SQL),

[0320] which in turn determines the multi-processing potential as described above in the MTCP Multi-processing.

[0321] The normal daily processing flow to achieve 100% auditability in either real-time or batch mode is as follows:

```
    (Begin Program)
    Read System Clock to Store Begin Time
    (Read Next Transaction)
    If Last Transaction
        Read System Clock to Store End Time
        Write End Time, Begin Time, Number of Accepts, Number of Rejects,
            and Total Records Read to Summary Table
        Goto End of Program
    Increment Total Records Read by 1
    (Add a New Record)
    If transaction is "Add" then
        If record exists then
            Process Addition Error
            Goto Write Reject Table
    ********************************************************
        * Select System Clock Date       into Insert - Transaction Date        *
        * Increment Sequence Number      into Insert - Transaction Number      *
        * Select User Name               into Insert - Transaction User        *
        * Select Zero                    into Update - Transaction Number      *
        * Select Zero                    into Delete - Transaction Number      *
    *********************************************************
        Insert to Master Table
        Goto Write Accept Table
        (Change an Existing Record)
If transaction is "Change" then
        If record does not exist then
            Process Change Error
            Goto Write Reject Table
    *********************************************************
    * (Master Snapshot)                              *
    * Move Master Table Record to History Table      *
    *********************************************************
    * Select System Clock Date       into Update - Transaction Date        *
    * Increment Sequence Number      into Update - Transaction Number      *
    * Select User Name               into Update - Transaction User        *
    * Select Zero                    into Delete - Transaction Number      *
    * Increment Master Table Number of Modifications by 1          *
    *********************************************************
    Update Master Table with New Data
    Goto Write Accept Table
(Delete an Existing Record)
    If transaction is "Delete" then
        If record does not exist then
            Process Drop Error
            Goto Write Reject Table
    *********************************************************
        * Select System Clock Date       into Delete - Transaction Date    *
        * Increment Sequence Number      into Delete - Transaction Number *
        * Select User Name               into Delete - Transaction User      *
    *********************************************************
        * Update Master Table Record for Tran Date/Tran Num/User           *
    *********************************************************
        * (Master Snapshot)                            *
        * Move Master Table Record to History Table               *
    *********************************************************
```

-continued

```
      Delete Master Table Record From Master Table
      (Write MULTI-PROCESSED Accept Table)
****************************************
      * Move "Current"            into Archive - Status *
      * Move "System Date"        into Archive - Date   *
****************************************
      Increment Accept Counter
      Insert to Accept Table
      Insert Second Accept Table (on a separate disk drive, if desired)
      Goto Loop to Next Transaction
(Write Reject Table)
            Increment Reject Counter
            Insert to Reject Table
(Loop to Next Transaction)
            Goto Read Next Transaction
(End of Program)
            End
```

[0322] Step 5: At the end of the "proofing period", such as daily or weekly, when proof tallies are matched to computer tallies, the Accept Table can be deleted as follows:

[0323] (Begin Program)

[0324] Delete All Records from the Accept Table

[0325] (End Program)

[0326] Step 6: Backup all databases and tables before any information is purged as follows:

[0327] (Begin Program)

[0328] Write All Tables in the "Current" database to backup

[0329] Write All Tables in the "Archive" database to backup

[0330] (End of Program)

[0331] Step 7: At the end of a user-defined period, an archive and purge process occurs that

```
(Begin Program)
****************************************
* Move "Archive"              to Archive Status
* Move "System Date"        to Archive Date
****************************************
Move All Records in the Master Table to Master
Archive.
Move All Records in the History Table to the
History Archive.
(End Program)
```

[0332] Step 8: In the event that current records are wrongfully moved to the History Archive,

[0333] they may be retrieved by

[0334] (Begin Program)

[0335] Move Specific Records from the Master Archive to the Master Table

[0336] Move Specific Records from the History Archive to the History Table

[0337] (End Program)

[0338] This program should be executed only after Records have been moved from the Current database **300** to the Archive database **304**. It should never be run after new transactions have been processed to the Current database **300**.

[0339] MTCP Backup/Recovery

[0340] If necessary, a recovery program can be utilized at any time in the event of hardware failure. Upon complete recovery, Step **7** and Step **8** will have to be re-executed to insure the correct status before the next day's processing is begun. The Accept Table can then be used to as a substitute Input Table to return the system to its previous processing point. Once this table is exhausted, data from the Input Table would supply the remaining data for the processing job.

[0341] MTCP Management

[0342] Once test data are defined and processed, a business enterprise may

[0343] (a) Review lists of the contents of all Master Tables **312** for determining correctness.

[0344] (b) Reset the contents of all Master Tables for performing the next test.

[0345] (c) Retest.

[0346] MTCP Auditability

[0347] Once auditabilty is achieved, the business enterprise may query:

[0348] (a) When a Master Table Cluster was created.

[0349] (b) When each record was added (or inserted) to the Master Table **312**,

[0350] (c) How many authorized changes (or updates) have been made to a record of the Master Table **312**.

[0351] (d) Prove the integrity of the master transaction cluster by producing a sequential list of all record changes, and if the record was deleted, where the record is stored.

[0352] Accordingly, 100% auditability of every change, every day, for every application is possible.

[0353] Multiprocessing Defined

[0354] Unlike serial processing which processes all jobs in sequential fashion, multiprocessing processes some of the same jobs simultaneously, or in parallel. While multiprocessing is not new, major computer manufacturers such as Compaq, Digital, Hewlett-Packard, IBM, NCR, Unisys, etc. have announced offerings of low-cost multiprocessing machines based on 2, 4, 8, and sixteen processors. These machines will rapidly increase the demand for multiprocessing software, which is known as "multithreaded" software. Multithreaded software permits the simultaneous execution of more than one jobs or job sequences.

[0355] Multiprocessing takes two forms, Symmetrical Multiprocessing (SMP) and Massively Parallel Processing (MPP), the difference being that symmetrical multiprocessing machines collectively have only one bus between the processors and the peripheral storage. For example, a symmetrical multiprocessing machine may have eight processors, one bus, and sixteen disk drives. In contrast, massive parallel processing machines has one bus for each processor. For example, a massively parallel machine may have eight processor, eight busses, and sixteen disk drives. Therefore, symmetrical multiprocessing machines are best suited for applications with a high processing content and a low input/out content. In contrast, massively parallel processing machines are best suited for applications that can be parallelized and have a high input/output requirement, as is the case with many commercial systems.

[0356] In either event, multiprocessing machines are best utilized when carefully tuned to avoid bottlenecks. This is likely to mean that all of the layers constituting a computing environment are multiprocessing-enabled. That is, the hardware, operating system, relational database management system, and the specific application are capable of multiprocessing. Some multiprocessing mainframes have been available for several years as well as some versions of the UNIX operating system. Only a few multiprocessing relational databases exist and even fewer multiprocessing applications. It is believed by some that the success of multiprocessing is solely dependent upon the "knowledge of the application" rather than "knowledge of the underlying tools," the tools being the hardware, operating system, and relational database system.

[0357] Accordingly, it is believed that the limiting factors for the success of multiprocessing for financial systems depends on:

[0358] (1) the lack of financial transaction application knowledge,

[0359] (2) a lack of understanding of how multiprocessing can be used to effect 100% auditability, and

[0360] The value of MTCP is that it addresses the last form of multiprocessing which is believed to be the most critical to delivering rapid response times for real-time financial transaction processing systems. That is, by dividing a transaction into subtransactions that can be spread across several multiprocessors, processing throughput may be faster. Plus, the large number of small programs make maintenance much easier and less expensive.

[0361] A first embodiment of the transaction processing controller **52** is provided in the flowchart of **FIG. 6**. Note that for simplicity, error handling and related validity checking steps have been omitted. However, the performance of such steps is within the scope of the present invention, as one skilled in the art will appreciate. A second pseudo-code embodiment of the transaction processing controller **52** follows.

Pseudo-Code for the Command Processor

(Transaction Processing Controller **52**)

[0362]

```
BEGIN
    /* The following switches are global. They control both the activity of the system.      */
    /* The Processor Switches monitors the availability of an eight processor computer.       */
    /* The Process Switches monitors all of the jobs that are to be executed.                 */
    /* These switches initialize the system, and then change throughout processing            */
    /* as the subcomponents of the system and the processors finish.                          */
    /* The Processor Switches are turned ON as jobs are sent to specific processors.          */
    /* The Processor Switches are turned OFF after the jobs are completed.                    */
    Set Processor 1 Switch = 0
    Set Processor 2 Switch = 0
    Set Processor 3 Switch = 0
    Set Processor 4 Switch = 0
    Set Processor 5 Switch = 0
    Set Processor 6 Switch = 0
    Set Processor 7 Switch = 0
    Set Processor 8 Switch = 0
    Read Begin Time from Systems Clock into Working Storage
    Set Total Records Read = 0
    Set Number Accepts    = 0
    Set Number Rejects    = 0
    /* The Command Programs reads the transaction input from the operator, then               */
    /* edits the transaction for validity and loads the transaction processing algorithms     */
    /* from the Transaction Processing table (or cache file) to a temporary table. It then    */
    /* walks down all of algorithms in the temporary table to process the total transaction   */
    /* with 100% auditability. Each algorithm may be passed to a separate processor.
    /* Read operator instructions for starting and ending item in input stream                */
    /* For the purposes of restart in the event of mid-stream job failure                     */
    /* For the purpose of omissions in processing.                                            */
```

-continued

```
/*      Operator may enter Begin .......................... End for all items                    */
/*      Operator may enter Begin ..... End          for a beginning list                         */
/*      Operator may enter       Begin ..... End                 for an intermediate list        */
/*      Operator may enter              Begin ..... End for an ending list                       */
Read Beginning Item in Input Stream from Master Control Terminal
Read Ending Item    in Input Stream from Master Control Terminal
Set Beginning Item  to Next Transaction
Set Ending Item     to End of List
Read System Clock for Begin Time
Add Record with Begin Time
IF Error then
      Message "No System Table Record for Begin Time", Details
ENDIF
<<Read Next Transaction>>
/* The Process Switches are turned ON as each transaction subcomponent is completed.     */
/* The Process Switches are turned OFF after the total transaction is completed.       */
Set Process  1 Switch = 0
Set Process  2 Switch = 0
Set Process  3 Switch = 0
Set Process  4 Switch = 0
Set Process  5 Switch = 0
Set Process  6 Switch = 0
Set Process  7 Switch = 0
Set Process  8 Switch = 0
Set Process  9 Switch = 0
Set Process 10 Switch = 0
Set Process 11 Switch = 0
Set Process 12 Switch = 0
Set Process 13 Switch = 0
Set Process 14 Switch = 0
Set Process 15 Switch = 0
Set Process 16 Switch = 0
Set Process 17 Switch = 0
Set Process 18 Switch = 0
Set Process 19 Switch = 0
Set Process 20 Switch = 0
Set Process 21 Switch = 0
Set Process 22 Switch = 0
Set Process 23 Switch = 0
Set Process 24 Switch = 0
Read Next Transaction into Working Storage
IF EOF then
      Read End Time from Systems Clock into Working Storage
      INSERT End-time, Begin Time
            Total Records Read, Number Accepts, Number Rejects
            into Summary Table
      IF Error-then
            Message "INSERT ST Table", Details
            STOP
      ENDIF
      Goto EOJ
ENDIF
IF Next Transaction = End of List
      Goto EOJ
ENDIF
Increment Total Records Read
<<Test Transaction Type>>
IF Transaction Type != ' ' then
/* Set Switches for Trade Offset and Settle Offset Processing          */
      Set Process  1 Switch = 0
      Set Process  2 Switch = 1
      Set Process  3 Switch = 1
      Set Process  4 Switch = 1
      Set Process  5 Switch = 1
      Set Process  6 Switch = 0
      Set Process  7 Switch = 1
      Set Process  8 Switch = 1
      Set Process  9 Switch = 1
      Set Process 10 Switch = 1
      Set Process 11 Switch = 0
      Set Process 12 Switch = 1
      Set Process 13 Switch = 1
      Set Process 14 Switch = 1
      Set Process 15 Switch = 1
      Set Process 16 Switch = 1
      Set Process 17 Switch = 0
```

-continued

```
    Set Process 18 Switch = 0
    Set Process 19 Switch = 1
    Set Process 20 Switch = 1
    Set Process 21 Switch = 1
    Set Process 22 Switch = 1
    Set Process 23 Switch = 1
    Set Process 24 Switch = 0
ENDIF
<<Test OORR>>
IF OORR = ‘O’ then
    *****************
    CALL N_gine EDIT
    *****************
    IF Edit Error
        Message “Edit Error”, Details
        Goto Write Reject Table
    ENDIF
    IF Tran-Type != ‘Sell’
    OR Tran-Type != ‘Withdraw’ then
        INSERT into Transaction Journal Table
        IF Error
            Message “Insert TJ Error”, Details
            Goto Write Reject Table
        ENDIF
        IF Correction Data then
            DELETE from Reject Table
            IF Error
                Message “Delete Reject Error”, Details
                Goto Write Reject Table
            ENDIF
        ENDIF
    ENDIF
    *********
    CALL TT           i.e., execute the algorithms in the temporary table
    *********
    IF Temporary Table Error then
        Message “Temporary Table Error”, Details
        Goto Write Reject Table
    ENDIF
    Generate Sequence Number
ELSIF OORR = ‘R’
    *****************
    CALL N_gine EDIT
    *****************
    IF Edit Error
        Message “Edit Error”, Details
        Goto Write Reject Table
    ENDIF
    Assign Transaction Number = ‘000000’
    Assign LOT Number       = 1
    <<Read Next Reversal>>
    Read Transaction Journal Table for reversal number
    IF “No Transaction Exists” where LOT = 1 then
        Message “No Transaction Exists”, Details
        Goto Write Reject Table
    ENDIF
    IF “No Transaction Exists” and LOT> 1 then
        Goto Transaction Wrap-up
    ENDIF
    IF Previously Reversed
        Message “Previously Reversed”, Details
        Goto Write Reject Table
    ENDIF
    INSERT Reversing Transaction” to Transaction Journal Table
    IF Error
        Message “INSERT TJ Reversing Error”, Details
        Goto Write Reject Table
    ENDIF
    UPDATE “Reversed” Transaction
    IF Error
        Message “”UPDATE TJ Reversed Error”, Details
        Goto Write Reject Table
    ENDIF
    Increment the LOT Number
    *********
```

-continued

```
    CALL TT         i.e., execute the algorithms in the temporary table
    *********
    IF Temporary Table Error then
        Message "Temporary Table Error", Details
        Goto Write Reject Table
    ENDIF
    Goto Read Next Reversal
    Generate Sequence Number
    UPDATE "Reversed" Transaction, ALL ROWS with Reversing Data
    IF Error then
        Message "UPDATE TL Table Reversed", Details
        Goto Write Reject Report
    ENDIF
    UPDATE "Reversing" Transaction, ALL ROWS with Reversed Data
    IF Error then
        Message "UPDATE TL Table Reversing", Details
        Goto Write Reject Report
    ENDIF
ELSE
    INSERT into Reject Table "No Originate or Reverse Code"
    IF Error then
        Message "Insert Reject Table", Details
        Goto Write Reject Table
    ENDIF
ENDIF
<<Transaction Wrap-up>>
INSERT INTO Transaction Count Table
Select Original-Count and Reversal Count from TC Table into Working Storage
IF Error then
    INSERT INTO TC Table, Details
    IF Error then
        Goto Write Reject Table
    ENDIF
ELSE
    IF      AORS = 'O' then
            Increment Original-Count
    ELSIF   AORS = 'R'
            Increment Reversal-Count
    ELSE
            Message "Invalid AORS Code", Details
            STOP
    ENDIF
ENDIF
<<Test Trade Settlement>>
IF      Transaction Switch = 2
        Goto Loop Next Transaction
ENDIF
IF      Transaction Switch = 1
OR      AORS = " then
        Goto Loop Next Transaction
ENDIF
/* COMMIT Work to Database            */
COMMIT Original Transaction Before Offset Transaction
IF      AORS = 'A' then
        Insert Licensee Trade Offset Buy in Transaction Identifier
ELSIF   AORS = 'S'
        Insert Licensee Trade Offset Sell in Transaction Identifier
ELSE
        Message "Invalid AORS", Details
ENDIF
/* Swap Account Numbers for Automatic Transaction        */
Move Account Number to Working Storage Account Number
Move Buyer/Seller Number to Account Number
Move Working Storage Account Number to Account Number
Multiply the Net Amount by    -1
Multiply the Amount Units by -1
Add Number of Settlement Days from Entity Master to Trade Date to determine Settlement Date
Add to Total Number of Accepts
UPDATE Row in System Table for Number of Accepts
IF Error then
    Message "Update Error for Accepts", Details
    Goto Write Reject Record
ENDIF
Go to Test Transaction Type
<<Loop Next Transaction>>
/* COMMIT Work to Database            */
COMMIT Original Transaction or Offset Transaction, if any
```

-continued

```
    Goto Read Next Transaction
    <<Write Reject Record>>
    Add to Total Number of Rejects
    UPDATE Row in System Table for Number of Rejects
    IF Error then
        Message “Update Error for Rejects”, Details
    ENDIF
    INSERT Into Reject Table, Details
    IF Error
        Message “Insert Command Reject Table”, Details
        STOP
    ENDIF
    Move Incoming Licensee Identifier to Stored Licensee Identifier
    Move Incoming Account Identifier to Stored Account Identifier
    Move Incoming Transaction Identifier to Stored Transaction Identifier
    Move Incoming Entity Identifier to Stored Entity Identifier
    Goto Read Next Transaction
    <<EOJ>>
    Read System Clock for End Time
    Add Record with End Time
    IF Error then
        Message “No System Table Record for End Time”, Details
    ENDIF
END
```

[**0363**] A first embodiment of the transaction preprocessor and decomposer **54** is provided in the flowcharts of FIGS. **7**-A through **7**-D and FIGS. **8**-A and **8**-B. Note that for simplicity, error handling and related validity check steps have been omitted. However, the performance of such steps is within the scope of the present invention, as one skilled in the art will appreciate.

[**0364**] A second pseudo-code embodiment of the transaction preprocessor and decomposer **54** follows.

Pseudo-Code for the Edit Processor for all Incoming Transactions

(Transaction Preprocessor and Decomposer **54**)

[**0365**]

```
BEGIN
    Housekeeping
        Set Working Storage Alphas to Blanks
        Set Working Storage Numbers to Zeroes
    IF Incoming Licensee Identifier = Stored Licensee Identifier then
        Using Licensee Identifier from Input String, retrieve
            Licensee Name
            Trade Settlement Switch
            Trade Offset Buy
            Trade Offset Sell
            from Licensee Master into Working Storage
        IF Error then
            Message “No Licensee Master”, Detail
            Goto EOJ
        ENDIF
    ENDIF
    /***********************************************/
    IF      the Default Definition Table has not been loaded to memory then
        LOAD all records from the Default Definition Table consisting of
            Licensee
            DD Class
            DD Identification
            DD Sub-Class
            DD Accounting Control Number
            DD Name
                from the Default Definition Table
                into the Temporary Table (TA)
        IF Error then
            Message “NO TA Table”, Details
            Goto EOJ
        ENDIF
    ENDIF
    /************************************************************/
```

-continued

```
IF the Incoming Account Identifier = Stored Account Identifier
    Goto Access Transaction Master (TM)
ELSE
    /*** This is the first table containing control totals for cash, units, and cost basis ***/
    <<Access Account Master>>
    From the Account Master Table (TM)
    using the Licensee Identifier from the Input String
      and the Account Identifier from the Input String, retrieve
        Account Type
        Income Posting Code
        Income/Expense Switch
        Receipt/Disbursement Switch
        Performance Measurement Switch
        Fiscal Year - Month
        Fiscal Year - Day
        Fiscal Year - Number Periods
        Income Cash Balance
        Principal Cash Balance
        Invested Income
        Invested Principal
        Total Units - Assets
        Liabilities
        Total Units - Liabilities
        and the Row Identification of the Account Master Record
            from the Account Master Table (AM) into Working Storage
    IF Error then
        Report "Invalid Account Identifier", Details
        Goto Write Reject Report
    ENDIF
ENDIF
<<Access Transaction Master>>
IF      the Incoming Transaction Identifier = Stored Transaction Identifier
    Goto Test Cash Entry in Entity Attribute Table
ELSE
    Using the Licensee Identifier from the Input String
      and the Transaction Identifier from the Input String
        Transaction Name
        Add or Subtract Switch
        Settlement Switch
        and the Row Identification
            from the Transaction Master Table (TM) into Working Storage
    IF Error then
        Message "Invalid Transaction Identifier", Details
        Goto Write Reject Report
    ENDIF
    IF AORS = 'A' then
        Using the Licensee Identifier from the Input String
          and the Trade Offset Buy from Working Storage, verify
          the existence of a Trade Offset Buy in the TM Table
        IF Error then
            Message "No Trade Offset Buy", Details
            Goto Write Reject Table
        ENDIF
    ELSE AORS = 'S' then
        Using the License Identifier from the Input String
          and the Trade Offset Sell from Working Storage, verify
          the existence of a Trade Offset Sell in the TM Table.
        IF Error then
            Message "No Trade Offset Sell", Details
            Goto Write Reject Table
        ENDIF
    ELSE
        Message "Invalid AORS Code", Details
        Goto Write Reject Report
    ENDIF
    <<Access Transaction Processing Table (TP)>>
    Using the Licensee Identifier from the Input String
      and the Transaction Identifier from the Input String, retrieve
        ALL of the Transaction Processing algorithms
        from the Transaction Processing Table (TP)
        into a Temporary Table (TT) in Working Storage
    IF Error then
        Message "No Transaction Processing Algorithms", Details
        Goto Write Reject Report
    ENDIF
    /*** This is the second control table containing cash, units, cost basis, liabilities, etc. ***/
```

-continued

```
    <<Test Income Cash Posting Controls>>
    IF   the Working Storage Income Posting Code = 'I'
    OR the Working Storage Income Posting Code = 'B' then
        Count the number of IC entries in the TA table
        <<Test Income Cash>>
        IF count = 1 then
        Using Licensee Identifier from the Input String
         and the Class = 'IC'
        and the Sub-Class = ' ' retrieve
            Accounting Control Number from TA into Working Storage
        IF Error then
            Message "Invalid Income Cash ACN", Details
            Goto Write Reject Record
        ENDIF
        Using the Licensee Identifier from the Input String
         and the Accounting Control Number in Working Storage, retrieve
         Accounting Control Number
            and the Row Identification from General Ledger Table (SG)
        IF Error then
            Message "Invalid Income Cash on SG", Details
            Goto Write Reject Report
        ENDIF
    ELSIF count = 2 then
        Using the Licensee Identifier from the Input String
        and the Class     = 'IC'
        and the Sub-class = 'D', retrieve
            Accounting Control Number from TA into Working Storage
        IF Error then
            Message "Invalid Income Cash Demand ACN in TA", Details
            Goto Write Reject Report
        ENDIF
        Using the Licensee Identifier from the Input String
        and the Accounting Control Number in Working Storage, retrieve
            Accounting Control Number
            and the Row Identification from the General Ledger
        IF Error then
            Message "Invalid Income Cash Demand in GL", Details
            Goto Write Reject Report
        ENDIF
        Using the Licensee Identifier from the Input String
         and the Class     = 'IC'
         and the Sub-class = 'O', retrieve
         Accounting Control Number from TA table into Working Storage
        IF Error then
            Message "Invalid Income Cash Overdraft ACN in TA",
                    Details
            Goto Write Reject Report
        ENDIF
        Using the Licensee Identifier from the Input String
         and the Accounting Control Number in Working Storage, retrieve
            Accounting Control Number
            and the Row Identification from the General Ledger
        IF Error then
            Message "Invalid Income Cash Overdraft in GL", Details
            Goto Write Reject Report
        ENDIF
    ELSE
        Message "Invalid Income Cash Count on DD", Details
        Goto Write Reject Record
    ENDIF
<<Test Principal Cash Posting Controls>>
ELSIF the Working Storage Income Posting Code = 'P'
    Count the number of PC entries in the TA table
    <<Test Principal Cash>>
    IF count = 1 then
        Using the Licensee Identifier from the Input String
         and the Class = 'PC'
         and the Sub-Class = ' ' retrieve
            Accounting Control Number from TA into Working Storage
        IF Error then
            Message "Invalid Principal Cash ACN", Details
            Goto Write Reject Record
        ENDIF
        Using the Licensee Identifier from the Input String
         and the Accounting Control Number in Working Storage, retrieve
         Accounting Control Number
            and the Row Identification from General Ledger Table (SG)
```

-continued

```
            IF Error then
                Message “Invalid Principal Cash on SG”, Details
                Goto Write Reject Report
            ENDIF
        ELSIF count = 2 then
            Using the Licensee Identifier from the Input String
            and the Class     = ‘PC’
            and the Sub-class = ‘D’, retrieve
                Accounting Control Number from TA into Working Storage
            IF Error then
                Message “Invalid Principal Cash Demand ACN in TA”,
                        Details
                Goto Write Reject Report
            ENDIF
            Using the Licensee Identifier from the Input String
            and the Accounting Control Number in Working Storage, retrieve
                Accounting Control Number
                and the Row Identification from the General Ledger
            IF Error then
                Message “Invalid Principal Cash Demand in GL”, Details
                Goto Write Reject Report
            ENDIF
            Using the Licensee Identifier from the Input String
            and the Class     = ‘PC’
            and the Sub-class = ‘O’, retrieve
            Accounting Control Number from TA table into Working Storage
            IF Error then
                Message “Invalid Principal Cash Overdraft ACN in TA”.
                        Details
                Goto Write Reject Report
            ENDIF
            Using the Licensee Identifier from the Input String
             and the Accounting Control Number in Working Storage, retrieve
                Accounting Control Number
                and the Row Identification from the General Ledger
            IF Error then
                Message “Invalid Principal Cash Overdraft in GL”, Details
                Goto Write Reject Report
            ENDIF
        ELSE
            Message “Invalid Principal Cash Count on DD”, Details
            Goto Write Reject Record
        ENDIF
    ELSE
        Message “Invalid Posting Code”, Details
        Goto Write Reject Report
    ENDIF
ENDIF
<<Test Cash Entry in Entity Attribute Table>>
Using the Licensee Identifier from the Input String
 and the Account Control Number from the TU Record in Working Storage, retrieve
    The Total Units - Assets
    and the Row Identifier from the Entity Attribute Table (EA)
IF Error then
    Message “Invalid Total Units”, Details
    Goto Write Reject Table
    ENDIF
    <<Test Asset / Liability Processing>>
    IF Working Storage Add or Subtract Switch (AORS; is OFF then
        Goto EOJ
    ENDIF
    IF Incoming Entity Identifier = Stored Entity Identifier then
        Goto EOJ
    ENDIF
    /*** This is the third table containing control table for cash, units, cost basis, liabilities, etc. ***/
    <<Access Entity Attribute Table (EA)>>
        Using the Licensee Identifier from the Input String
         and the Entity Identifier from the Input String, retrieve
            Accounting Control Number (Asset)
            Accounting Control Number (Liability)
            Diversification Type
            Diversification Group
            Diversification Class
            Invested Income Balance
            Invested Principal Balance
```

-continued

```
        Total Units - Assets
        Total Units - Liabilities
        and the Row Identification of the Entity Attribute Record
            from the Entity Attribute Table (EA) into Working Storage
    IF Error then
        Message "Invalid Entity Identifier in EA", Details
        Goto Write Reject Table
    ENDIF
<<Access the Entity Transaction Table (ET)>>
    Using the Licensee Identifier from the Input String
     and the Entity Identifier from the Input String, verify
        the existence of an acceptable transaction
        in the Entity Transaction Table (ET) for the Entity Identifier.
    IF Error then
        Message "Invalid Transaction for this Entity", Details
        Goto Write Reject Table
    ENDIF
    <<Access the Entity Master Table (EM)>>
    Using the Entity Identifier from the Input String, retrieve
        Income Rate
        Income Ex-Date
        Income Record Date
        Income Payment Date
        Cap-Adj Rate
        Cap-Adj Ex-Date
        Cap-Adj Record Date
        Cap-Adj Payment Date
        Settlement Days
        Current Price
            from the Entity Master Table (EM) into Working Storage
    IF Error then
        Message "No Entity Master", Details
        Goto Write Reject Report
    ENDIF
<<Test Other Assets>>
    Using the Licensee Identifier from the Input String
     and the Account Type from Working Storage
     and the Accounting Control Number - Asset from Working Storage, retrieve
        the Accounting Control Number - Asset
        and Row Identifier from the General Ledger (SG)
    IF Error then
        Message "Invalid ACN - Asset", Details
        Goto Write Reject Report
    ENDIF
<<Test Other Liabilities>>
    Using the Licensee Identifier from the Input String
     and the Account Type from Working Storage
     and the Accounting Control Number - Liability from Working Storage, retrieve
        the Accounting Control Number - Liability
        and Row Identifier from the General Ledger (SG)
    IF Error then
        Message "Invalid ACN - Liabilities", Details
        Goto Write Reject Report
    ENDIF
<<Test Invested Income>>
    Using the Licensee Identifier from the Input String
     and the Account Type Code from Working Storage
     and the Invested Income Identifier from Working Storage, retrieve
        the Invested Income Balance
        and the Row Identifier from the General Ledger Table (SG)
    IF Error then
        Message "Invalid Invested Income"
        Goto Write Reject Table
    ENDIF
<<Test Invested Principal>>
    Using the Licensee Identifier from the Input String
     and the Account Type Code from Working Storage
     and the Invested Principal Identifier from Working Storage, retrieve
        the Invested Principal Balance
        and the Row Identifier from the General Ledger Table (SG)
    IF Error then
        Message "Invalid Invested Principal"
        Goto Write Reject Table
    ENDIF
    Goto EOJ
```

-continued

```
    <<Write Reject Table>>
        Add to Reject Table
        IF Error then
            Message "Invalid Insert to Reject Table", Details
            STOP
        ENDIF
    <<EOJ>>
        Null
END
```

Pseudo-Code for the SCHEDULER

(Subtransaction Scheduler **62**)

[**0366**]

```
BEGIN
        <<Read Next Process>>
        Read Next Transaction in Temporary Table (TT)
                IF EOJ then
                        <<Test All Switches - AORL>>
                        IF      All 18 Process Switches = 0
                                Goto EOJ
                        ENDIF
                        Wait 10 milliseconds
                        Goto Test All Switches - AORL
                ENDIF
        <<Test Processor Availability>>
        IF Processor 1 Switch = 0 then
                Set Processor 1 Switch = 1
                Initiate Process on Processor 1         @ end, Set Processor 1 Switch = 0
                Goto Next Process Loop
        ENDIF
        IF License Master (LM) Number of Processors = 1 then
                <<Test 1 Processor>>
                IF Processor 1 Switch = 1 then
                        Wait 10 Milliseconds
                        Goto Test 1 Processor
                ENDIF
                Goto Test Processor Availability
        ENDIF
        IF Processor 2 Switch = 0 then
                Set Processor 2 Switch = 1
                Initiate Process on Processor 2         @ end, Set Processor 2 Switch = 0
                Goto Next Process Loop
        ENDIF
        IF License Master (LM) Number of Processors = 2 then
                <<Test 2 Processors Busy>>
                IF      Processor 1 Switch = 1
                AND     Processor 2 Switch = 1 then
                        Wait 10 milliseconds
                        Goto Test 2 Processors Busy
                ENDIF
                Goto Test Processor Availability
        ENDIF
        IF Processor 3 Switch = 0 then
                Set Processor 3 Switch = 1
                Initiate Process on Processor 3         @ end, Set Processor 3 Switch = 0
                Goto Next Process Loop
        ENDIF
        IF Processor 4 Switch = 0 then
                Set Processor 4 Switch = 1
                Initiate Process on Processor 4         @ end, Set Processor 4 Switch = 0
                Goto Next Process Loop
        ENDIF
        IF License Master (LM) Number of Processors = 4 then
                <<Test 4 Processors Busy>>
                IF      Processor 1 Switch = 1
                AND     Processor 2 Switch = 1
```

-continued

```
                AND     Processor 3 Switch = 1
                AND     Processor 4 Switch = 1 then
                        Wait 10 milliseconds
                        Goto Test 4 Processors Busy
                ENDIF
                Goto Test Processor Availability
        ENDIF
        IF Processor 5 Switch = 0 then
                Set Processor 5 Switch = 1
                Initiate Process on Processor 5        @ end, Set Processor 5 Switch = 0
                Goto Next Process Loop
        ENDIF
        IF Processor 6 Switch = 0 then
                Set Processor 6 Switch = 1
                Initiate Process on Processor 6        @ end, Set Processor 6 Switch = 0
                Goto Next Process Loop
        ENDIF
        IF Processor 7 Switch = 0 then
                Set Processor 7 Switch = 1
                Initiate Process on Processor 7        @ end, Set Processor Switch 7 = 0
                Goto Next Process Loop
        ENDIF
        IF Processor 8 Switch = 0 then
                Set Processor 8 Switch = 1
                Initiate Process on Processor 8        @ end, Set Processor 8 Switch = 0
                Goto Next Process Loop
        ENDIF
        IF Licensee Master (LM) Number of Processors = 8 then
                <<Test 8 Processors Busy>>
                IF      Processor 1 Switch = 1
                AND     Processor 2 Switch = 1
                AND     Processor 3 Switch = 1
                AND     Processor 4 Switch = 1
                AND     Processor 5 Switch = 1
                AND     Processor 6 Switch = 1
                AND     Processor 7 Switch = 1
                AND     Processor 8 Switch = 1 then
                        Wait 10 milliseconds
                        Goto Test 8 Processors Busy
                ENDIF
                Goto Test Processor Availability
        ENDIF
        <<Next Process Loop>>
                Goto Read Next Process
        <<EOJ>>
        Null
END
```

Process the Controls Process Routine in the Temporary Table (TT)

[0367]

```
BEGIN
        IF OORR = “O” then
                Set Factor = + 1
        ELSIF OORR = ‘R’ then
                Set Factor = − 1
        ENDIF
        <<Total Units>>
        IF      Operand 2 = ‘TU’ then
                (AMU)   Process AM      Units
                (EAU)   Process EA      Units
                (PMU)   Process PM      Units
        <<Cash Balances>>
        ELSIF   Operand 2 = ‘IC’
        OR      Operand 2 = ‘PC’ then
                (AMC)   Process AM      Income Cash Demand
                                        Income Cash Overdraft
                                        Principal Cash Demand
                                        Principal Cash Overdraft
                (EAC)   Process         EA Income Cash
                                        Principal Cash
```

-continued

```
        (GLC)   Process GL              Assets - Income Cash
                                        Demand
                                        Assets - Income Cash
                                        Overdraft
                                        Assets - Principal
                                        Cash Demand
                                        Assets - Principal
                                        Cash Overdraft
                                        Liab - Income Net Worth
                                        Liab - Principal Net Worth
        <<Investment Balances>>
        ELSIF   Operand 2 = ‘II’
        OR      Operand 2 = ‘IP’ then
                (AMI)   Process AM      Invested Income
                                        Invested Principal
                (EAI)   Process EA      Cost
                (GLI)   Process GL      Assets - Actg
                                        Control Number
                                        Liab - Income Net Worth
                                        Liab - Principal Net Worth
```

-continued

```
        <<Other Customized Investment Reporting>>
        ELSIF   Operand 2 = ‘I’ and Report Request = ‘Y’
        OR      Operand 2 = ‘E’ and Report Request = ‘Y’ then
                (IEE)   Process IE
                (PME)   Process PM
        <<Receipts/Disbursements>>
        ELSIF   Operand 2 = ‘R’ and Report Request = ‘Y’
        OR      Operand 2 = ‘D’ and Report Request = ‘Y’ then
                (IEC)   Process RD
                (PMC)   Process PM
        <<Performance Measurement>>
        ELSIF   Operand 2 = ‘PM’ and Report Request = ‘Y’ then
                (PMP)   Process PM
        <<Contributions/Distributions>>
        ELSIF   Operand 2 = ‘CN’ and Report Request = ‘Y’
        OR      Operand 2 = ‘DN’ and Report Request = ‘Y’ then
                (CDC)   Process PM
        <<Management Fees>>
        ELSIF   Operand 2 = ‘MF’ and Report Request = ‘F’ then
                (PMM)   Process PM
        <<Commissions>>
        ELSIF   Operand 2 = ‘CM’ then
                (PCM)   Process PM
        <<Federal Taxes>>
        ELSIF   Operand 2 = ‘FT’ then
                (PMF)   Process PM
        <<State Taxes>>
        ELSIF   Operand 2 = ‘ST’ then
        (PMS)   Process PM
        ELSE
                Message “Invalid Operand 2”
                STOP
        ENDIF
END
```

Process the Detail Records Maintenance Routine (AORS)

[0368] Note: Leave all switches=1 until the last routine is completed. This forces the processing to loop through each succeeding routine until completed. Then turn set all swtiches=0 so that the Scheduler will revert back to the Command Program to read another transaction.

```
<<Originate ADD>>
IF OORR = ‘O’ and
      AORS = ‘A’ then
        IF      Process 1 Switch = 0 then
                 Set Process 1 Switch = 1
                 Initiate Process BS
        ELSIF   Process 2 Switch = 0 then
                 Set Process 2 Switch = 1
                 Initiate Process PI/PA
        ELSIF   Process 3 Switch = 0 then
                 Set Process 3 Switch = 1
                 Initiate Process TS
        ELSIF   Process 4 Switch = 0 then
                 Set Process 4 Switch = 1
                 Initiate Process PM
        ELSE
                 Set Process 1 Switch = 0
                 Set Process 2 Switch = 0
                 Set Process 3 Switch = 0
                 Set Process 4 Switch = 0
        ENDIF
```

-continued

```
<<Reverse ADD>>
ELSIF OORR = ‘R’ and
      AORS = ‘A’ then
        IF      Process 5 Switch = 0 then
                Set Process 5 Switch = 1
                Initiate Process BS
        ELSIF   Process 6 Switch = 0 then
                Set Process 6 Switch = 1
                Initiate Process PI/PA
        ELSIF   Process 7 Switch = 0 then
                Set Process 7 Switch = 1
                Initiate Process TS
        ELSIF    Process 8 Switch = 0 then
                Set Process 8 Switch = 1
                Initiate Process PM
        ELSE
                Set Process 5 Switch = 0
                Set Process 6 Switch = 0
                Set Process 7 Switch = 0
                Set Process 8 Switch = 0
        ENDIF
<<Originate SUB>>
ELSIF OORR = ‘O’ and
      AORS = ‘S’ then
        IF      Process 9 Switch = 0 then
                Set Process 9 Switch = 1
                Initiate Process BS
        ELSIF   Process 10 Switch = 0 then
                Set Process 10 Switch = 1
                Initiate Process PI/PA
        ELSIF   Process 11 Switch = 0 then
                Set Process 11 Switch = 1
                Initiate Process TS
        ELSIF   Process 12 Switch = 0 then
                Set Process 12 Switch = 1
                Initiate Process CG
        ELSIF   Process 13 Switch = 0 then
                Set Process 13 Switch = 1
                Initiate Process PM
        ELSE
                Set Process 9 Switch = 0
                Set Process 10 Switch = 0
                Set Process 11 Switch = 0
                Set Process 12 Switch = 0
                Set Process 13 Switch = 0
        ENDIF
<<Reverse SUB>>
ELSIF OORR = ‘R’ and
      AORS = ‘S’ then
        IF      Process 14 Switch = 0 then
                Set Process 14 Switch = 1
                Initiate Process BS
        ELSIF   Process 15 Switch = 0 then
                Set Process 15 Switch = 1
                Initiate Process PI/PA
        ELSIF   Process 16 Switch = 0 then
                Set Process 16 Switch = 1
                Initiate Process TS
        ELSIF   Process 17 Switch = 0 then
                Set Process 17 Switch = 1
                Initiate Process CG
        ELSIF   Process 18 Switch = 0 then
                Set Process 18 Switch = 1
                Initiate Process PM
        ELSE
                Set Process 14 Switch = 0
                Set Process 15 Switch = 0
                Set Process 16 Switch = 0
                Set Process 17 Switch = 0
                Set Process 18 Switch = 0
        ENDIF
ENDIF
```

[0369] A first embodiment of the processing for the subtransaction processing module **64** is provided in the flowcharts of FIGS. **9**-A through **9**-B, **FIGS. 10, 11**, **12**, **13** and **14**. Note that for simplicity, error handling and related validity checking steps have been omitted. However, the performance of such steps is within the scope of the present invention, as one skilled in the art will appreciate.

[0370] A second pseudo-code embodiment of the transaction processing controller **52** follows.

Pseudo-Code for Processing for the Subtransaction Processing Module **64**

[0371]

```
BEGIN
  DO WHILE List of Subtransactions in the TT Table is Valid
        Select Next Row of Operator. Operand 1, and
        Operand 2 from TT into Working Storage
        /*      To choose the specific input field (or column) */
        IF      Operand 1 = 'N'
                Set Value = Net Amount                  from Input String
        ELSIF   Operand 1 = 'I'
                SetValue = Interest                     from Input String
        ELSIF   Operand 1 = 'P'
                Set Value = Principal                   from Input String
        ELSIF   Operand 1 = 'H'
                Set Value = Amount Units                from Input String
        ELSIF   Operand 1 = 'U'
                Set Value = Amount Units                from Input String
        ELSIF   Operand 1 = 'C'
                Set Value = Cost Basis                  from Input String
        ELSIF   Operand 1 = 'V'
                Set Value = Amount Units * Curr Price   from Input String
        ELSIF   Operand 1 = 'F'
                Set Value = Federal Taxes          from Input String
        ELSIF   Operand 1 = 'S'
                Set Value = State Taxes                 from Input String
        ELSIF   Operand 1 = 'L'
                Set Value = Local Taxes                 from Input String
        ELSIF   Operand 1 = 'M'
                Set Value = Management Fees             from Input String
        ELSE
                Message "Invalid Operand 1", Details
        ENDIF
        /*      To Adjust for Plus or Minus             */
        IF      Operator = 'P' then
                Set Multiplier = +1
        ELSIF   Operator = 'M' then
                Set Multiplier = -1
        ENDIF
        /*      To Adjust for Originate or Reversal     */
        IF      OORR = 'O' then
                Set Multiplier = Multiplier * +1
        ELSIF   OORR = 'R'
                Set Multiplier = Multiplier * -1
        ENDIF
        /* Test for Total Unit Changes                    */
        IF      Operand 2 = 'TU' then
                Add Value to AM - Total Units
                Add Value to EA - Total Units
        /* Test for Income Cash Changes                   */
        IF      Operand 2 = 'IC' then
                /* Add to First Controls - Account Master */
                Add Value to AM - Income Cash
                Add Value to AM - Units
                /* Add to Second Controls - Entity Attribute */
                Add Value to EA - Invested Income
                Add Value to EA - Units
                /* Add to Third Controls - General Ledger */
                IF      Number of Entries = 1 then
                        Add Value to GL - Income Cash
                ELSIF   Number of Entries = 2 then
                        IF      Value > 0 then
                                IF      ICD >= 0 then
                                          Add Value           to GL - Income Cash Demand
                                ELSE    ICD < 0
                                          Add (Value - ICO)   to GL - Income Cash Demand
                                          Set Zero            to GL - Income Cash Overdraft
                                ENDIF
                        ELSIF   Value <= 0 then
```

-continued

```
                        IF      ICD < 0 then
                                 Add Value           to GL - Income Cash Overdraft
                        ELSE    ICD >= 0 then
                                 Add (Value - ICD)   to GL - Income Cash Overdraft
                                 Set Zero            to GL - Income Cash Demand
                        ENDIF
                ELSE
                        Message "Invalid Value", Details
                ENDIF
                Add Value to Uninvested Income
        ELSE
                Message "Invalid Number Entries", Details
        ENDIF
/* Test for Principal Cash Changes                     */
ELSIF Operand 2 = 'PC' then
        /* Add to First Controls - Account Master      */
        Add Value to AM - Principal Cash
        Add Value to AM - Units
        /* Add to Second Controls - Entity Attribute */
        Add Value to EA - Invested Principal
        Add Value to EA - Units
        /* Add to Third Controls - General Ledger      */
        IF      Number of Entries = 1 then
                Add Value to GL - Principal Cash
        ELSIF   Number of Entries = 2 then
                IF      Value > 0 then
                        IF      PCD >= 0 then
                                 Add Value           to GL - Principal Cash Demand
                        ELSE    PCD < 0
                                 Add Value           to GL - Principal Cash Demand
                                 Set Zero            to GL - Principal Cash Overdraft
                        ENDIF
                ELSIF   Value <= 0 then
                        IF      PCD < 0 then
                                 Add Value           to GL - Principal Cash Overdraft
                        ELSE    PCD >= 0 then
                                 Add (Value - PCD)   to GL - Principal Cash Overdraft
                                 Set Zero            to GL - Principal Cash Demand
                        ENDIF
                ELSE
                        Message "Invalid Value", Details
                ENDIF
        ELSE
                Message "Invalid Number Entries", Details
        ENDIF
        Add Value to Uninvested Principal
/* Test for Invested Income Changes                    */
ELSIF   Operand 2 = 'II' then
        /* Add to First Controls - Account Master      */
        Add Value to AM - Invested Income
        /* Add to Second Controls - Entity Attribute */
        Add Value to EA - Invested Income
        /* Add to Third Controls - General Ledger      */
        /*      Update Assets                        */
        Add Value to ACN- Assets
        /*      Update Liabilities                   */
        IF      ACN-Liab = ' ' then
                Add Value to Invested Income
        ELSE
                Add Value to ACN_Liabilities
        ENDIF
/*      Test for Invested Principal Changes            */
ELSIF   Operand 2 = 'IP' then
        /* Add to First Controls - Account Master      */
        Add Value to AM - Principal Cash
        /* Add to Second Controls - Entity Attribute */
        Add Value to EA - Invested Principal
        /* Add to Third Controls - General Ledger      */
        /*      Update Assets                          */
        Add Value to ACN - Assets
        /*      Update Liabilities                   */
        IF      ACN_Liab = ' ' then
                Add Value to Invested Principal
        ELSE
                Add Value to ACN_Liabilities
        ENDIF
```

-continued

```
        /* Test for Other Customized Reporting Changes              */
        ELSIF   Operand 2 = ‘I’ and Report Request = ‘Y’
        OR      Operand 2 = ‘E’ and Report Request = ‘Y’ then
                (IEE)   Process IE
                (PME)   Process PM
        ELSIF   Operand 2 = ‘R’ and Report Request = ‘Y’
        OR      Operand 2 = ‘D’ and Report Request = ‘Y’ then
                (IEC)   Process RD
                (PMC)   Process PM
        /* Test for other Performance Measurement Data              */
        ELSIF   Operand 2 = ‘PM’ and Report Request = ‘Y’ then
                (PMP)   Process PM
        ELSIF   Operand 2 = ‘CN’
        OR      Operand 2 = ‘DN’ then
                (CDC)   Process PM
        ELSIF   Operand 2 = ‘MF’ then
                (PMM)   Process PM
        ELSIF   Operand 2 = ‘CM’ then
                (PCM)   Process PM
        ELSIF   Operand 2 = ‘FT’ then
                (PMF)   Process PM
        ELSIF   Operand 2 = ‘ST’ then
                (PMS)   Process PM
        ELSE
                Message “Invalid Operand 2”, Details
        ENDIF
        /* Test for Detail Record Maintenance of Financial Instruments        */
        IF      AORS != ‘ ‘ then
                ***********
                CALL PORTFOLIO ADJUSTER 110
                ***********
        ENDIF
  ENDDO
END
```

Pseudo-Code for Performance Measurement (PM)

Processing related to the Licensee Performance Measurement Table **104**

[**0372**]

```
BEGIN
        IF   Trxn = ‘A’ and Type = ‘O’ OR Trxn = ‘S’ and Type = ‘R’ (which
        means ADD)
                SELECT Data into Working Storage from PM Record
                IF Error then
                        INSERT INTO PM Record, Details
                        IF Error then
                                Message “INSERT PM Error”, Details
                                Goto Write Reject Report
                        ENDIF
                ELSE
                        Increment Units by amount to be increased
                        UPDATE Data to Table / Row
                        IF Error
                                Message “UPDATE PM Error 1”, Details
                                Goto Write Report Error
                        ENDIF
                ENDIF
        ELSIF Trxn = ‘A’ and Type = ‘R’ OR Trxn = ‘S’ and Type = ‘O’ (which
        means SUBTRACT)
                SELECT Data into Working Storage from PM Record
```

-continued

```
        IF Error then
            Message " SELECT PM Error 2", Details
            Goto Write Report Error
        ENDIF
        IF Units = 'ALL"
        and All Other Balances in the Row are Zero then
            DELETE from Table / Row
            IF Error
                Message "DELETE PM Error", Details
                Goto Write Report Error
            ENDIF
        ELSE
            Decrement Units by Amount to be reduced
            UPDATE PI SET Details
            IF Error then
                Message "UPDATE PM Error 2", Details
                Goto Write Report Writer
            ENDIF
        ENDIF
    ELSE
        Null
    ENDIF
    Goto EOJ
    <<Write Reject Report>>
    INSERT into Reject Table, Details
        IF Error
            STOP
        ENDIF
    <<EOJ>>
    Null
END
```

Pseudo-Code for Income/Expense Processing (IE)

Processing Related to the Customer Income Statement (Income/Expense) Table **96**

[0373]

```
BEGIN
    IF Trxn = 'Debit' and Type = 'O'         (which means ADD)
    OR Trxn = 'Credit' and Type = 'O' then
        SELECT Data into Working Storage from IE Record
        IF Error then
            INSERT INTO IE Table, Details
            IF Error then
                Message "INSERT IE Error 1",
                Details
                Goto Write Report Error
            ENDIF
        ELSE
            Increment Units by amount to be increased
            UPDATE Data to Table / Row
            IF Error then
                Message "UPDATE IE Error 1",
                Details
                Goto Write Report Error
            ENDIF
        ENDIF
    ELSIF Trxn = 'Debit' and Type = 'R'        (which
                                               means SUBTRACT)
    OR  Trxn = 'Credit' and Type = 'R' then
        SELECT Data into Working Storage from IE Record
        IF Error then
            Message "SELECT IE Error 2", Details
            Goto Write Report Error
        ENDIF
```

-continued

```
        IF Units = 'ALL" then
            DELETE from Table / Row
            IF Error then
                Message "DELETE IE Error",
                Details
                Goto Write Report Error
            ENDIF
        ELSE
            Decrement Units by Amount to be reduced
            UPDATE IE SET Details
            IF Error then
                Message "UPDATE IE Error 2",
                Details
                Goto Write Report Writer
            ENDIF
        ENDIF
    ELSE
        Null
    ENDIF
    Goto EOJ
    <<Write Reject Report>>
    INSERT into Reject Table, Details
        IF Error then
            STOP
        ENDIF
    <<EOJ>>
    Null
END
```

Pseudo-Code for AORS Processing

(Portfolio Adjuster **110** Processing)

[**0374**]

```
BEGIN
    /* The End AORS Switch is a global switch that signals the end of all AORS
    processing */
    /* otherwise known as the Detail Record (or Row) Maintenance Processing. */
    /* The switch is originally set = 0. Each called routine ends by setting the
    switch = 1. */
    Set End AORL Switch = 0
    DO WHILE End AORS Switch = 0
        IF Trxn = “ADD” then
            IF Type = ‘O’ then
                ************
                CALL Original Add Module 114  (Originate Add)
                ************
                IF Error
                    Message “No OADD Routine”
                    Goto Write Reject Report
                ENDIF
            ELSIF Type = ‘R’ then
                ************
                CALL Reverse Add Module 118  (Reverse Add)
                ************
                IF Error
                    Message “NO RADD Routine”
                    Goto Write Reject Routine
                ENDIF
            ELSE
                Message “Invalid O OR R Code for ADD”, Details
                Goto Write Reject Report
            ENDIF
        ELSIF Trxn = ‘SUBTRACT’ then
            IF Type = ‘O’ then
                ************
                CALL Original Sell Module 122  (Originate Subtract)
                ************
                IF Error then
                    Message “No OSUB Routine”, Details
                    Goto Write Reject Report
                ENDIF
            ELSIF Type = ‘R’ then
                ************
                CALL Reverse Sell Module 126  (Reverse Subtract)
                ************
                IF Error then
                    Message “No RSUB Routine, Details
                    Goto Write Reject Report
                ENDIF
            ELSE
                Message “Invalid O OR R for SUBTRACT”, Details
                Goto Write Reject Report
            ENDIF
        ELSE
            Message “Invalid Transaction”, Details
            Goto Write Reject Report
        ENDIF
        Goto EOJ
        <<Write Reject Report>>
        INSERT into Reject Table
            IF Error then
                STOP
            ENDIF
        Set End AORL Switch =1
        <<EOJ>>
        Null
    ENDDO
END
```

[0375] A first embodiment of the processing for the balance sheet table **130** is provided in the flowchart of Fig. BAL-SHT. Note that for simplicity, error handling and related validity checking steps have been omitted. However, the performance of such steps is within the scope of the present invention, as one skilled in the art will appreciate.

[0376] A second pseudo-code embodiment of the processing for the balance sheet table **130** follows.

[0377] Balance Sheet Processing (BS)

```
BEGIN
        IF      AORL = ‘A’ and OORR = ‘O’    (which means ADD)
        AND     AORL = ‘S’ and
                OORR = ‘R’ then
                SELECT Data into Working Storage from BS Record
                IF Error then
                        INSERT INTO BS Table, Details
                        IF Error then
                                Message “INSERT BS Error’,
                                Details
                                Goto Write Reject Table
                        ENDIF
                ELSE
                        Increment Units by amount to be increased
                        UPDATE Data to Table / Row
                        IF Error
                                Message “UPDATE BS Error 1”,
                                Details
                                Goto Write Report Error
                        ENDIF
                ENDIF
        ELSIF   AORL = ‘A’ and OORR = ‘R’    (which means
                                             SUBTRACT)
        OR      AORL = ‘S’ and OORR = ‘O’ then
                SELECT Data into Working Storage from BS Record
                IF Error then
                        Message “SELECT BS Error 2”, Details
                        Goto Write Report Error
                ENDIF
                IF Units = ‘ALL” then
                        DELETE from Table / Row
                        IF Error
                                Message “DELETE BS Error”,
                                Details
                                Goto Write Report Error
                        ENDIF
                ELSE
                        Decrement Units by Amount to be reduced
                        UPDATE IE SET Details
                        IF Error then
                                Message “UPDATE BS Error 2”,
                                Details
                                Goto Write Report Writer
                        ENDIF
                ENDIF
        ELSE
                Null
        ENDIF
        Goto EOJ
        <<Write Reject Report>>
        INSERT into Reject Table, Details
                IF Error
                        STOP
                ENDIF
        <<EOJ>>
        Null
END
```

Pseudo-Code For Processing The Capital Gains Table 140

[0378]

```
BEGIN
        IF AORL = ‘S’ and Type = ‘O’        (which means ADD)
                SELECT Data into Working Storage from CG Record
                IF Error then
                        INSERT INTO CG Table, Details
                        IF Error then
                                Message “INSERT CG Table”,
                                Details
                                Goto Write Report Error
                        ENDIF
                ELSE
                        Increment Units by amount to be increased
                        UPDATE Data to Table / Row
                        IF Error
                                Message “UPDATE CG Error 1”,
                                Details
                                Goto Write Report Error
                        ENDIF
                ENDIF
        ELSIF AORL = ‘S ’ and Type = ‘R’    (which means
                                            SUBTRACT)
                SELECT Data into Working Storage from CG Record
                IF Error then
                        Message “SELECT CG Error 2”, Details
                        Goto Write Report Error
                ENDIF
                IF Units = ‘ALL” then
                        DELETE from Table / Row
                        IF Error
                                Message “DELETE CG Error”,
                                Details
                                Goto Write Report Error
                        ENDIF
                ELSE
                        Decrement Units by Amount to be reduced.
                        UPDATE IE SET Details
                        IF Error then
                                Message “UPDATE CG Error 2”,
                                Details
                                Goto Write Report Writer
                        ENDIF
                ENDIF
        ELSE
                Null
        ENDIF
        Goto EOJ
        <<Write Reject Report>>
        INSERT into Reject Table, Details
                IF Error
                        STOP
                ENDIF
        <<EOJ>>
        Null
END
```

[0379] Note: do not turn switch OFF or back to 0 as these swithces indicate which processes remain.

Pseudo-Code for Original Add Module 114 Processing

[0380]

```
BEGIN
        IF      Process 1 Switch = 0 then
                Set Process 1 Switch = 1
                ********
                CALL BS
                ********
```

-continued

```
        ELSIF     Process 2 Switch = 0 then
                  Set Process 2 Switch = 1
                  ********
                  CALL PI
                  ********
        ELSIF     Process 3 Switch = 0 then
                  Set Process 3 Switch = 1
                  ********
                  CALL PA
                  ********
        ELSIF     Process 4 Switch = 0 then
                  Set Process 4 Switch = 1
                  ********
                  CALL TS
                  ********
        ELSIF     Process 5 Switch = 0 then
                  Set Process 5 Switch = 1
                  ********
                  CALL PM
                  ********
                  Set End AORS Switch = 1   Notes End of AORS
                                            Processing
        ELSE
                  NULL
        ENDIF
        *****************
        CALL Subtransaction Scheduler 62
        *****************
END
```

Pseudo-Code for Reverse of Add Module **118** Processing

[**0381**] Note: Do not turn switch OFF or back to 0 as these switches indicate which processes remain.

```
BEGIN
  IF        Process 6 Switch = 0 then
            Set Process 6 Switch = 1
            ********
            CALL BS
            ********
  ELSIF     Process 7 Switch = 0 then
            Set Process 7 Switch = 1
            **********
            CALL PI
            **********
  ELSIF     Process 8 Switch = 0 then
            Set Process 8 Switch = 1
            ********
            CALL PA
            ********
  ELSIF     Process 9 Switch = 0 then
            Set Process 9 Switch = 1
            ********
            CALL TS
            ********
  ELSIF     Process 10 Switch = 0 then
            Set Process 10 Switch = 1
            *********
            CALL PM
            *********
            Set End AORS Switch = 1    Notes End of AORS Processing
  ELSE
            NULL
  ENDIF
  *****************
  CALL Subtransaction Scheduler 62
  *****************
END
```

PSEUDO-CODE FOR ORIGINAL SELL MODULE **122** PROCESSING

[**0382**]

```
BEGIN
  IF Sell-Method = 'LOT' then
    Select LOT Amount into Working Storage from BS record
    IF Amount Sold > Lot Amount in Working Storage then
              Message "Lot Amount > Amount Available"
              Goto Write Reject Report
    ENDIF
    IF        Process 11 Switch = 0 then
               Set Process 11 Switch = 0
              ***************
              CALL BS
              ***************
    ELSIF     Process 12 Switch = 0 then
               Set Process 12 Switch = 0
              **********
              CALL PI
              **********
    ELSIF     Process 13 Switch = 0 then
               Set Process 13 Switch = 0
              **********
              CALL PA
              **********
    ELSIF     Process 14 Switch = 0 then
              Set Process 14 Switch = 0
              **********
              CALL CG
              **********
    ELSIF     Process 15 Switch = 0 then
               Set Process 15 Switch = 1
              **********
              CALL TS
              **********
    ELSIF     Process 16 Switch = 0 then
               Set Process 16 Switch = 0
              **********
              CALL PM
              **********
    ELSIF     Process 17 Switch = 0 then
               Set Process 17 Switch = 0
              **********
              CALL TL
              **********
              Set End AORS Switch = 1         Notes End of
              AORS Processing
    ELSE
              NULL
    ENDIF
    *****************
    CALL SUBTRACTION SCHEDULER 62
    *****************
  ELSE
    Select all LOTS into Temporary Working Storage Table
    Licn/Acct/Asset/Purch/Amt/Cost/Unit-Cost/ROWID)
    Set Total Amount Sold = Data Entry Amount Sold
    IF Total Amount Sold > Total Amount Available then
              Message "Total Amount Sold > Total Amount Available",
              Details
              Goto Write Reject Report
    ENDIF
    Avg-Factor = 1
    IF Sell-Method = "AVG" then
              Avg-Factor = (Total Amount Sold / Total
              Amount Available)
    ENDIF
  <<Sell Multiple Lot Routine>>
  DO While Total Amount Sold = 0
              IF Total Amount Sold > 0 then
                IF Sell-Method = 'FIF' or ' ' then
                  Select LOT Amount Available into WS Lot Amount
                    Where Purch = MIN (Purch)
                ENDIF
```

-continued

```
            ELSIF
              IF Sell-Method = 'LIF"
                Select LOT Amount Available into WS Lot Amount
                  Where Purch = MAX(Purch)
              ENDIF
            ELSIF
              IF Sell-Method = 'LCF'
                Select LOT Amount Available into WS Lot Amount
                  Where Unit-Cost = MIN(Unit-Cost)
              ENDIF
            ELSIF
              IF Sell-Method = 'HCF"
                Select LOT Amount Available into WS Lot Amount
                  Where Unit-Cost = MAX(Unit-Cost)
              ENDIF
            ELSE
              <<for Sell-Method = 'AVG' or 'ALL'>>
                IF Amount Sold * Avg Factor < WS Lot Amount then
                  UPDATE Temporary Table Lot Amount
                  for Amount Sold
              ELSE
              DELETE Total Row Temporary Table
            ENDIF
            **********
    IF      Process 11 Switch = 0 then
            Set Process 11 Switch = 0
            ****************
            CALL BS
            ****************
    ELSIF   Process 12 Switch = 0 then
            Set Process 12 Switch = 0
            **********
            CALL PI
            **********
    ELSIF   Process 13 Switch = 0 then
            Set Process 13 Switch = 0
            **********
            CALL PA
            **********
    ELSIF   Process 14 Switch = 0 then
            Set Process 14 Switch = 0
            **********
            CALL CG
            **********
    ELSIF   Process 15 Switch = 0 then
            Set Process 15 Switch = 1
            **********
            CALL TS
            **********
    ELSIF   Process 16 Switch = 0 then
            Set Process 16 Switch = 0
            **********
            CALL PM
            **********
    ELSIF   Process 17 Switch = 0 then
            Set Process 17 Switch = 0
            **********
            CALL TL
            **********
            Set End AORS Switch = 1     Notes End of
            AORS Processing
    ELSE
            NULL
    ENDIF
    Decrement Total Amount Sold by Cap Gain Lot Amount
    Increment the e LOT Number
    *****************
    CALL SUBTRANSACTION SCHEDULE 62
    *****************
  ENDIF
ENDDO
   ENDIF
   <<EOJ>>
    NULL
  END
```

Originate Sell Routine

[0383]

```
BEGIN
        IF Sell-Method = 'LOT' then
            Select LOT Amount into Working Storage from BS record.
            IF Amount Sold > Lot Amount in Working Storage then
                Message "Lot Amount > Amount Available"
                Goto Write Reject Report
            ELSE
                ****************
                CALL BS Routine
                ****************
            ENDIF
            **********
            CALL PIPA
            **********
            **********
            CALL CG
            **********
            **********
            CALL TS
            **********
            **********
            CALL PM
            **********
            **********
            CALL CG
            **********
            **********
            CALL TL
            **********
        ELSE
            Select All LOTS into Temporary Working Storage Table
            Licn/Acct/Asset/Purch/Amt/Cost/Unit-Cost/ROWID)
            Set Total Amount Sold = Data Entry Amount Sold
            IF Total Amount Sold > Total Amount Available then
                Message "Total Amount Sold > Total Amount
                Available", Details
                Goto Write Reject Report
            ENDIF
            Avg-Factor = 1
            IF Sell-Method = 'AVG' then
                Avg-Factor = (Total Amount Sold /
                Total Amount Available)
            ENDIF
            DO While Total Amount Sold = 0
                IF Total Amount Sold > 0 then
            IF Sell-Method = 'FIF' or ' ' then
                Select LOT Amount Available into WS Lot Amount
                    Where Purch = MIN (Purch)
            ENDIF
        ELSIF
            IF Sell-Method = 'LIF"
                Select LOT Amount Available into WS Lot Amount
                    Where Purch = MAX (Purch)
            ENDIF
        ELSIF
            IF Sell-Method = 'LCF'
                Select LOT Amount Available into WS Lot Amount
                    Where Unit-Cost = MIN(Unit-Cost)
            ENDIF
        ELSIF
            IF Sell-Method = 'HCF"
                Select LOT Amount Available into WS Lot Amount
                    Where Unit-Cost = MAX (Unit-Cost)
            ENDIF
        ELSE
            <<for Sell-Method = 'AVG' or 'ALL'>>
            IF Amount Sold * Avg Factor < WS Lot Amount then
                UPDATE Temporary Table Lot Amount for
                Amount Sold
```

-continued

```
               ELSE
                    DELETE Total Row Temporary Table
               ENDIF
               **********
               CALL BS      with the amount of LOT sold
               **********
               ENDIF
               **********
               CALL PIPA
               **********
               **********
               CALL TS
               **********
               **********
               CALL PM
               **********
               **********
               CALL CG        with the amount of LOT sold
               **********
               **********
               CALL TL
               **********
               Decrement Total Amount Sold by Cap Gain Lot Amount
               Increment the LOT Number
          ENDIF
     ENDDO
ENDIF
Goto EOJ
<<Write Reject Report>>
          INSERT into Reject Table
               IF Error then
                    STOP
               ENDIF
          <<EOJ>>
END
```

Pseudo-Code for Reverse of

Original Sell Module **126** Processing

[0384]

```
BEGIN
   IF         Process 18 Switch = 0 then
              Set Process 18 Switch = 1
              ********
              CALL BS        with the amount of LOT sold
              ********
   ELSIF      Process 19 Switch = 0 then
              Set Processor 19 Switch = 1
              ********
              CALL PI
              ********
   ELSIF      Process 20 Switch = 0 then
              Set Process 20 Switch = 1
              ********
              CALL PA
              ********
   ELSIF      Process 21 Switch = 0 then
              Set Process 21 Switch = 1
              ********
              CALL TS
              ********
   ELSIF      Process 22 Switch = 0 then
              Set Process 22 Switch = 1
              *********
              CALL PM
              *********
```

-continued

```
   ELSIF      Process 23 Switch = 0 then
              Set Process 23 Switch = 1
              ********
              CALL CG        with the amount of LOT sold
              ********
   ELSIF      Process 24 Switch = 0 then
              Set Process 24 Switch = 1
              ********
              CALL TL
              ********
              Set End AORL Switch = 1              Notes End of AORS
Processing
   ELSE
              NULL
   ENDIF
   *****************
   CALL Subtransaction Scheduler 62
   *****************
END
```

Pseudo-Code for Processing Model #4

For All INSERTS, UPDATES, and DELETES to all Tables

[0385]

```
      BEGIN
        IF Trxn is 'ADD' then
          SELECT Data in Working Storage
          IF Error then
            INSERT INTO Table, Details
            IF Error then
              Message "INSERT Error", Details
              Goto Write Reject Report
            ENDIF
          ELSE
            Increment the Details
            UPDATE Set Table, Details
            IF Error then
              Message "UPDATE Error ADD", Details
              Goto Write Reject Report
            ENDIF
          ENDIF
        ELSIF Trxn is 'SUBTRACT' then
          SELECT Data into Working Storage
            IF Error then
              Message "SELECT Error Subtract", Details
              Goto Write Reject Report
            ENDIF
          If One or More Amounts > One or More Values from
          Existing Record then
            ADD to Reject Report
            IF Error then
              Message "INSERT Reject SUBTRACT", Details
              Goto Write Reject Report
            ENDIF
          IF Details = 'ALL" then
            DELETE From Table, Details
            IF Error then
              Message "DELETE Error", Details
              Goto Write Reject Report
            ENDIF
          ELSE
            Decrement the Details
            UPDATE SET, Details
            IF Error then
              Message "UPDATE Error SUBTRACT", Details
              Goto Write Reject Report
```

-continued

```
            ENDIF
          ENDIF
        ENDIF
        Goto EOJ
        <<Write Reject Report>>
        INSERT INTO Reject Table, Details
        IF Error then
          Message "INSERT Reject Table Error", Details
          STOP
        ENDIF
        <<EOJ>>
        NULL
      END
```

Pseudo-Code for Processing the Trade Settlement Table **142**

[0386]

```
BEGIN
  IF   Trxn = 'A' and Type = 'O' OR Trxn = 'S' and
  Type = 'O'   (which means ADD)
    INSERT into TS table, Details
    IF Error then
      Message "INSERT TS Error 1", Details
      Goto Write Report Error
    END
  ELSIF Trxn = 'A' and Type = 'R' OR Trxn = 'S' and Type = 'R'
  (which means SUBTRACT)
    SELECT Data into Working Storage from TS Record
    IF Error then
      Message "SELECT TS Error 2", Details
      Goto Write Report Error
    ENDIF
    DELETE from Table/Row
    IF Error
      Message "DELETE TS Error", Details
      Goto Write Report Error
    ENDIF
  ELSE
    Null
  ENDIF
  Goto EOJ
  <<Write Reject Report>>
  INSERT into Reject Table, Details
    IF Error
      STOP
    ENDIF
  <<EOJ>>
  Null
END
```

Pseudo-Code for Processing the Customer Cash Flow (Receipts/Disbursements) Table **100**

[0387]

```
BEGIN
  IF Trxn = 'Receipt' and Type = 'O'     (which means ADD)
  OR Trxn = 'Disbursement' and Type = 'O' then
    SELECT Data into Working Storage from RD Record
    IF Error then
      INSERT INTO RD Table, Details
      IF Error then
        Message "INSERT RD Error", Details
        Goto Write Report Error
      ENDIF
    ELSE
      Increment Units by amount to be increased
      UPDATE Data to Table/Row
      IF Error then
        Message "UPDATE RD Error 1", Details
        Goto Write Report Error
      ENDIF
    ENDIF
  ELSIF Trxn = 'Receipt' and Type = 'R'  (which means SUBTRACT)
  OR      Trxn = 'Disbursement' and Type = 'R'
    SELECT Data into Working Storage from RD Record
    IF Error then
      Message " SELECT RD Error 2", Details
      Goto Write Report Error
    ENDIF
    IF Units = 'ALL" then
      DELETE from Table/Row
      IF Error
        Message "DELETE RD Error", Details
        Goto Write Report Error
      ENDIF
    ELSE
      Decrement Units by Amount to be reduced
      UPDATE IE SET Details
      IF Error then
        Message "UPDATE RD Error 2", Details
        Goto Write Report Writer
      ENDIF
    ENDIF
  ELSE
    Null
  ENDIF
  Goto EOJ
  <<Write Reject Report>>
  INSERT into Reject Table, Details
    IF Error then
      STOP
    ENDIF
  <<EOJ>>
  Null
END
```

Pseudo-Code for Processing the Pending Adjustment Table **138**

[0388]

```
BEGIN
  IF   Trxn = 'A' and Type = 'O' OR Trxn = 'S' and Type = 'R'
  (which means ADD)
  AND Trade Date<Income Ex-Date then
    SELECT Data into Working Storage from PA Record
    IF Error then
      INSERT INTO PA Table, Details
      IF Error then
        Message "INSERT PA Error", Details
        Goto Write Report Error
      ENDIF
    ELSE
      Increment Units by amount to be increased
      UPDATE Data to Table/Row
      IF Error
        Message "UPDATE PA Error 1", Details
        Goto Write Report Error
      ENDIF
    ENDIF
```

-continued

```
  ELSIF Trxn = 'A' and Type = 'R' OR Trxn = 'S' and Type = 'O'
  (which means SUBTRACT)
  AND   Trade Date > Income Ex-date + 1 then
    SELECT Data into Working Storage from PA Record
    IF Error then
      Message " SELECT PA Error 2", Details
      Goto Write Report Error
    ENDIF
    IF Units = 'ALL" then
      DELETE from Table/Row
      IF Error
        Message "DELETE PA Error", Details
        Goto Write Report Error
      ENDIF
    ELSE
      Decrement Units by Amount to be reduced
      UPDATE PA SET Details
      IF Error then
        Message "UPDATE PA Error 2", Details
        Goto Write Report Writer
      ENDIF
    ENDIF
  ELSE
    Null
  ENDIF
  Goto PA-EOJ
  <<Write Reject Report>>
  INSERT into Reject Table, Details
    IF Error
      STOP
    ENDIF
  <<PA-EOJ>>
  Null
END
```

Pseudo-Code for Processing

the Pending Income Table **134**

[0389]

```
BEGIN
  IF Trxn = 'A' and Type = 'O' OR Trxn = 'S' and Type = 'R'
  (which means ADD)
  AND Trade Date < Income Ex-Date then
    SELECT Data into Working Storage from PI Record
    IF Error then
      INSERT INTO PI Table, Details
```

-continued

```
      IF Error then
        Message "INSERT PI Error", Details
        Goto Write Reject Report
      ENDIF
    ELSE
      Increment Units by amount to be increased
      UPDATE Data to Table/Row
      IF Error
        Message "UPDATE PI Error 1", Details
        Goto Write Report Error
      ENDIF
    ENDIF
  ELSIF Trxn = 'A' and Type = 'R' OR Trxn = 'S' and Type = 'O'
  (which means SUBTRACT)
  AND    Trade Date > Income Ex-date + 1 then
    SELECT Data into Working Storage from PI Record
    IF Error then
      Message " SELECT PI Error 2", Details
      Goto Write Report Error
    ENDIF
    IF Units = 'ALL" then
      DELETE from Table/Row
      IF Error
        Message "DELETE PI Error", Details
        Goto Write Report Error
      ENDIF
    ELSE
      Decrement Units by Amount to be reduced
      UPDATE PI SET Details
      IF Error then
        Message "UPDATE PI Error 2", Details
        Goto Write Report Writer
      ENDIF
    ENDIF
  ELSE
    Null
  ENDIF
  Goto PI-EOJ
  <<Write Reject Report>>
  INSERT into Reject Table, Details
    IF Error
      STOP
    ENDIF
  <<PI-EOJ>>
  Null
END
```

N_gine File (or Table) Structure and Likely Order of Creation

corresponding with FIGS. **4**-A through **4**-E

[0390]

		Data Source
Institutional Profile		
LM	Licensee Master	User-Definable
LU	Licensee Users	User-Definable
LT	Licensee Account Type	User-Definable
LD	Licensee Default Definitions	User-Definable
LL	Licensee General Ledger Definitions	User-Definable
LS	Licensee Diversification Scheme	User-Definable
LP	Licensee Performance Group	User-Definable
LN	Licensee Summary Names	User-Definable
LW	Licensee Service Wholesalers	User-Definable
LR	Licensee Service Resellers	User-Definable

-continued

		Data Source
Customer Profile		
AO	Account Objective	User-Definable
AL	Account Legal Capacity	User-Definable
AJ	Account Jurisdiction	User-Definable
AR	Account Representatives	User-Definable
AN	Account Registration Names	User-Definable
AM*	Account Master	User-Definable
AC	Account Communication Links	User-Definable
Transaction Profile		
TM**	Transaction Master	User-Definable “Driving” File
TP**	Transaction Processor	User-Definable “Driving” File
TR	Transactions - Recurring	User-Definable “Driving” File
Entity Profile		
EM	Entity Master	Public Market Data
EA*	Entity Attribute	User-Definable
ET	Entity Transaction	User-Definable
Licensee Status		
SG*	System General Ledger	User-Definable
SJ*	System Transaction Journal	Sytem Defined “Driven” File
ST	System Trade Settlement	System Defined “Driven” File
SS	System Summary Table	System Defined
SR	System Reject Table	System Defined
SC	System Transaction Count	System Defined
Customer Status		
CS	Customer Income Statement (Income/Expense)	System Defined “Driven” File
CF	Customer Cash Flow (Receipts/Disbursements)	System Defined “Driven” File
CB*	Customer Balance Sheet	System Defined “Driven” File
CG	Customer Capital Gain	System Defined “Driven” File
CI	Customer Pending Income	System Defined “Driven” File
CA	Customer Pending Capital Adjustments	System Defined “Driven” File
CP*	Customer Performance Measurement	System Defined “Driven” File

Notes:
*denotes Primary Control Tables
**denotes “Driving Tables”

SAMPLE DATA FOR LICENSE GENERAL LEDGER DEFINITION TABLE (LL)

[0391]

Licensee Identifier	Asset or Liab	Accounting Control Number	Accounting Name
LICN1	A	A05	Municipal Bonds
LICN1	A	A07	Corporate Bonds
LICN1	A	A10	Common Stocks
LICN1	A	A12	Mutual Funds
LICN1	A	A13	International Currencies
LICN1	A	A15	Oil Partnerships
LICN1	A	A20	Real Estate Partnerships
LICN1	A	A30	Foreign Equities
LICN1	A	A35	Objects of Art
LICN1	A	A40	Jewelry
LICN1	A	A45	Homes
LICN	A	A50	Automobiles
LICN	A	A90	Derivatives
LICN2	A	W10	MSA/RSA - North
LICN2	A	W20	MSA/RSA - East
LICN2	A	W30	MSA/RSA - South
LICN2	A	W40	MSA/RSA - West
LICN2	A	L10	Alabama
LICN2	A	L20	Alaska
.		.	.
.		.	.
.		.	.
LICN2	A	L500	Wyoming
LICN2	A	S10	Major Market 1
LICN2	A	S20	Major Market 2
LICN2	A	S30	Major Market 3
.		.	.
.		.	.
.		.	.
LICN2	A	S1000	Major Market N
LICN3	A	C10	Cash
LICN3	A	C20	Other Current Assets
LICN3	A	C30	Fixed Assets
LICN3	A	C40	Depreciation
LICN3	A	C50	Intangible Assets
LICN1	L	L05	Uninvested Income
LICN1	L	L10	Invested Income
LICN1	L	L15	Uninvested Principal
LICN1	L	L20	Invested Principal

-continued

Licensee Identifier	Asset or Liab	Accounting Control Number	Accounting Name
LICN1	L	L30	Personal Notes
LICN1	L	L40	Mortgages
LICN1	L	L90	Income
LICN1	L	L60	Short-Term Liabilities
LICN1	L	L65	Deferred Taxes
LICN1	L	L70	Long-Term Liabilities
LICN1	L	L75	Net Worth

SAMPLE DATA FOR SYSTEM GENERAL LEDGER TABLE

[0392]

	Asset		Accounting	
Licensee	or	Account	Control	Accounting
Master	Liab	Type	Number	Name
LICN1	A	000	000	Financial Services Assets
LICN1	A	100	000	Pension Trust
LICN1	A	100	A01	Income Cash Demand
LICN1	A	100	A02	Income Cash Overdraft
LICN1	A	100	A03	Principal Cash Demand
LICN1	A	100	A04	Principal Cash Overdraft
LICN1	A	100	A07	Corporate Bonds
LICN1	A	100	A10	Common Stocks
LICN1	A	100	A15	Oil Partnerships
LICN1	A	100	A20	Real Estate Partnerships
LICN1	A	100	A30	Foreign Equities
LICN1	A	200	000	Investment Advisory
LICN1	A	200	A01	Income Cash Demand
LICN1	A	200	A02	Income Cash Overdraft
LICN1	A	200	A03	Principal Cash Demand
LICN1	A	200	A04	Principal Cash Overdraft
LICN1	A	200	A05	Municipal Bonds
LICN1	A	200	A07	Municipal Bonds
LICN1	A	200	A10	Common Stocks
LICN1	A	200	A12	Mutual Funds
LICN1	A	200	A13	International Currencies
LICN1	A	200	A15	Oil Partnerships
LICN1	A	200	A20	Real Estate Partnerships
LICN1	A	100	A30	Foreign Equities
LICN1	A	100	A90	Financial Derivatives
LICN1	A	300	000	Estates
LICN1	A	300	A01	Income Cash Demand
LICN1	A	300	A02	Income Cash Overdraft
LICN1	A	300	A03	Principal Cash Demand
LICN1	A	300	A04	Principal Cash Overdraft
LICN1	A	300	A05	Municipal Bonds
LICN1	A	300	A07	Corporate Bonds
LICN1	A	300	A10	Common Stocks
LICN1	A	300	A12	Mutual Funds
LICN1	A	300	A15	Oil Partnerships
LICN1	A	300	A20	Real Estate Partnerships
LICN1	A	300	A30	Foreign Equities
LICN1	A	300	A35	Objects of Art
LICN1	A	300	A40	Jewelry
LICN1	A	300	A40	Homes
LICN1	A	300	A50	Automobiles
LICN1	A	400	000	Settlement Accounts - Buy
LICN1	A	400	A01	Income Cash Demand
LICN1	A	400	A02	Income Cash Overdraft
LICN1	A	400	A03	Principal Cash Demand
LICN1	A	400	A04	Principal Cash Overdraft
LICN1	A	400	A05	Corporate Bonds
LICN1	A	400	A07	Municipal Bonds
LICN1	A	400	A10	Common Stocks
LICN1	A	400	A15	Oil Partnerships
LICN1	A	400	A20	Real Estate Partnerships
LICN1	A	400	A30	Foreign Equities
LICN1	A	500	000	Settlement Accounts - Sell
LICN1	A	500	A01	Income Cash Demand
LICN1	A	500	A02	Income Cash Overdraft
LICN1	A	500	A03	Principal Cash Demand
LICN1	A	500	A04	Principal Cash Overdraft
LICN1	A	500	A05	Corporate Bonds
LICN1	A	500	A07	Municipal Bonds
LICN1	A	500	A10	Common Stocks
LICN1	A	500	A15	Oil Partnerships
LICN1	A	500	A20	Real Estate Partnerships
LICN1	A	500	A30	Foreign Equities
			(AND/OR)	
LICN2	A	1000	000	Communication Assets
LICN2	A	1000	W00	Wireless Communications
LICN2	A	1000	W10	MSA/RSA - North
LICN2	A	1000	W20	MSA/RSA - East
LICN2	A	1000	W30	MSA/RSA - South
LICN2	A	1000	W40	MSA/RSA - West
LICN2	A	2000	L00	Landline Communications
LICN2	A	2000	L10	Alabama
LICN2	A	2000	L20	Alaska
.				.
.				.
.				.
LICN2	A	2000	L500	Wyoming
LICN2	A	3000	S00	Satellite Broadcast
LICN2	A	3000	S10	Major Market 1
LICN2	A	3000	S20	Major Market 2
LICN2	A	3000	S30	Major Market 3
.				.
.				.
.				.
LICN2	A	3000	S1000	Major Market 4
			(AND/OR)	
LICN3	A	0000	000	Corporate Assets
LICN3	A	9000	000	Domestic Subsidiary
LICN3	A	9000	C10	Cash
LICN3	A	9000	C20	Other Current Assets
LICN3	A	9000	C30	Fixed Assets
LICN3	A	9000	C40	Depreciation
LICN3	A	9000	C50	Intangible Assets
LICN3	A	9000	000	Foreign Subsidiary
LICN3	A	9000	C10	Cash
LICN3	A	9000	C20	Other Current Assets
LICN3	A	9000	C30	Fixed Assets
LICN3	A	9000	C40	Depreciation
LICN3	A	9000	C50	Intangible Assets
LICN3	L	000	000	Financial Services Liabilities
LICN1	L	100	000	Pension Trust
LICN1	L	100	L15	Uninvested Principal
LICN1	L	100	L20	Invested Principal
LICN1	L	200	000	Investment Advisory
LICN1	L	200	L05	Uninvested Income
LICN1	L	200	L10	Invested Income
LICN1	L	200	L15	Uninvested Principal
LICN1	L	200	L20	Invested Principal
LICN1	L	300	000	Estates
LICN1	L	300	L05	Uninvested Income
LICN1	L	300	L10	Invested Income
LICN1	L	300	L15	Uninvested Principal
LICN1	L	300	L20	Invested Principal
LICN1	L	300	L30	Personal Notes
LICN1	L	300	L40	Mortgages
LICN1	L	400	000	Settlement - Buy
LICN1	L	400	L15	Uninvested Principal
LICN1	L	400	L20	Invested Principal

-continued

Licensee Master	Asset or Liab	Account Type	Accounting Control Number	Accounting Name
LICN1	L	500	000	Settlement - Buy
LICN1	L	500	L15	Uninvested Principal
LICN1	L	500	L20	Invested Principal
			(AND/OR)	
LICN2	L	1000	000	Communications
LICN2	L	1000	000	Wireless
LICN2	L	1000	L90	Income
LICN2	L	2000	000	Landline
LICN2	L	2000	L90	Income
LICN2	L	3000	000	Satellite Broadcast
LICN2	L	3000	L90	Income
			(AND/OR)	
LICN3	L	9000	000	Domestic Subsidiary
LICN3	L	9000	L60	Short-Term Liabilities
LICN3	L	9000	L65	Deferred Taxes
LICN3	L	9000	L70	Long-Term Liabilities
LICN3	L	9000	L75	Net Worth
LICN3	L	9000	000	Foreign Subsidiary
LICN3	L	9000	L60	Short-Term Liabilities
LICN3	L	9000	L65	Deferred Taxes
LICN3	L	9000	L70	Long-Term Liabilities
LICN3	L	9000	L75	Net Worth

A Standardized Method for Naming the Programs (or SQL Scripts) and Data Elements of Real-time Multiprocessed Automated Applications

[0393] The specific invention is a standardized file naming convention to be used in the automatic generation of program code for multiple large-scale transaction processing applications (such as securities trading, telecommunications billing, and work management) on multi-processing computers (using 4, 8, 16, 32 processors) with 100% auditability of user-defined controls. The standardized file naming convention is totally independent of any specific

[0394] a.) application such as accounts receivable, customer billing, etc.,

[0395] b.) industry such as financial services, telecommunications, or work management,

[0396] c.) hardware manufacturer such as Compaq, Digital, HP, IBM, NCR, Unisys,

[0397] d.) operating system such as MS-DOS, UNIX, OpenVMS, MVS, etc.,

[0398] e.) relational database management system such as Oracle, Sybase, MS-SQL Server,

[0399] f.) computer language such as SQL, COBOL, Fortran, PL/1, etc.

[0400] The standard naming convention contains the fewest number of characters in any naming conventions; namely, eleven characters used by MS-DOS. The naming convention of MS-DOS uses eight characters as a file name and three characters as a file extension wherein the user may define a file name using the alphabet and selected other characters. While this flexibility is suitable for home use are a small number of files and users, it is not acceptable for large-scale enterprise-wide applications with large number of files and large number of supporting technicians. Hence, the need for enterprise-wide standards.

[0401] The standard file naming convent n contains six elements that permit the technician to readily identify the functionality of the specific script (or program) without looking at its contents. Using ANSI Standard structured Query Language as an example language, the six elements are:

[0402] a.) a 2-character mnemonic for the SQL commands such as:

Mnemonic	ANSI Standard SQL Commands
CT	Create Table
SF	Select From Table
DF	Delete From
DT	Drop Table
II	Insert Into
SI	Select Into
CS	Create Sequence
DS	Drop Sequence
CI	Create Index
DI	Drop Index
RV	Review
RT	Retest
RS	Reset, etc.

[0403] b.) a 2-character mnemonic for the application name such as

Mnemonic	User Defined Application Name Examples
ST	Securities Trading
TC	Telecommunications Billing
WM	Work Management, etc.

[0404] c.) a 2-character mnemonic for the table (or file name) such as

Mnemonic	User-Defined Table Name Examples
AM	Account Master Name/Address/Etc.
SM	Securities Master
DC	Detail Calls
XB	External Billing, etc.

[0405] d.) a 1-character mnemonic for the table cluster role such as

Mnemonic	Standard Table Roles
M	Master
I	Input
A	Accepts
R	Rejects
H	History
S	Summary
1	Master History
2	Accepts History
O	Output

[0406] e.) a 1-character mnemonic for the table cluster type such as

Mnemonic	Standard Table Types
M	Master
J	Journal
T	Temporary
1–9	Index Numbers

[0407] f.) a 3-character extension is then added to the file name depending upon

[0408] the type of operating system being used such as MS-DOS, UNIX, OpenVMS, etc. and

[0409] whether or not the file is a source file for programmer use or a compiled file (or stored procedure) for machine use.

[0410] Hence, script name examples are:

[0411] CTXBMDMM.SQL—Create Table for the External Billing System, Master Definition Table Cluster, Master Table, and Master Role for SQL use.

[0412] DTXBDCOJ.SQL—Drop Table for the External Billing System, Detail Call Cluster, Output Table, and Journal Role for SQL use.

[0413] Circumstances Leading to the Invention

[0414] The circumstances leading to the invention of a standard SQL script naming convention are:

[0415] a.) one programmer will rarely adhere to the same naming conventions over time and unless an acceptable standard is defined each succeeding programmer added to the job will only complicate the issue by bringing their own standards. Hence, software maintenance becomes a matter of knowing which programmer wrote which program at what time.

[0416] b.) without a naming standard any programmer has no idea of what functions the programming is performing without opening the program and examining the program code. This process produces create inefficient maintenance by existing programmers and inefficient training for new programmers.

[0417] c.) Competitive pressures are mounting for the efficient of software maintenance.

[0418] Advantage of the Invention

[0419] Because no duplicate script names are permitted the name of each SQL Script should

[0420] a.) convey to the user the precise use of each SQL Script and

[0421] b.) permit the storage of all SQL scripts in a one SQL Script Library, or directory.

[0422] A standard naming convention also permits the user to determine what scripts may be automatically executed in sequence by use of a SQL command script, which is a single SQL script containing a list of SQL scripts to be executed in sequence. Hence, any single SQL scripts contained in the SQL Library can be reused in many different SQL command scripts.

[0423] Although any standard naming convention represents a unique entity separate and apart from the other technologies described immediately above, this particular naming convention is unique in that it embraces all of the logical information necessary to readily identify the role of the script in the total system.

[0424] Detailed Description of Invention:

[0425] std_name is a standard naming convention that constructs names for programs (or SQL Scripts), system tables, table clusters, and data elements. The seven basic elements are:

1.) org_name Organization	2
2.) com_name SQL Command	2
3.) app_name Application	2
4.) tab_name Table	2
5.) rol_name Table Role	1
6.) typ_name Table Type	1
7.) col_name Column (or Field)	4

[0426] std_name defines both "external" names used by the operating system and "internal" names used by the specific program.

1.) clu_name Cluster Name	4
2.) sys_name System Table Name	6
3.) ext_name Extension Name	3
4.) sql_name SQL Script Name	11 (8 name plus 3 extension)

[0427] where the SQL Script Names are used by the operating systems.

[0428] The "internal" resulting names are:

1.) tab_iden	Table Iden Name	4
2.) col_name	Column (or Field) Name	4
3.) dat_name	Data Element Name	8 or more, in increments of 4

[0429] where the Data Element Names are used by the programs (or SQL Scripts).

[0430] External Names used by the operating system in identifying programs (or SQL Scripts) are created by employing the following naming components:

com_name	SQL Command Mnemonic
app_name	Application Name Mnemonic
tab_name	Table Name Mnemonic
rol_name	Table Role Name Mnemonic
tab_name	Table Type Name Mnemonic
ext_name	Extension Mnemonic

-continued

Examples:	1	2	3	4	5	6	7	8	.	9	10	11
	C	T	X	B	M	D	M	M	.	S	Q	L
	S	F	X	B	M	D	M	M	.	S	Q	L
	clu_name											
				tab_iden								
			sys_name		ext_name							
				sql_name								

[0431] Internal Names used by the program (or SQL Script) in processing the data elements are created by employing the following naming components:

	5 6 7 8		
	tab_name		Table Name Mnemonic
		rol_name	Role Name Mnemonic
		typ_name	Type Name Mnemonic
		col_name	Column name
Examples:	M D M M	LNAM	... for last name
	M D M M	FNAM	... for first name
	M D M M	MNAM	... for middle name
	M D M M	ADR1	... address - 1st line
	M D M M	ADR2	... address - 2cd line
	M D M M	CITY	... city
	M D M M	STAT	... state
	M D M M	ZIPC	... zip code
	dat_name		

[0432] Data Tracing

[0433] By addressing both the external names for the operating system and the internal names for a specific program, the naming convention is global in nature. In the event that one data element derives its source of input from another table rather than its own specific input screen, then the data name is extended by placing the table identifier of the table supplying the data between the first four and second four characters of the intended data name. Should the data be derived from another table that also derived its data from another table, then eight characters are placed between the first four characters and the last four characters of the intended data name. In the fashion, the data name points backwards through all of the preceding tables to the original source of data and its input form. This process is called "data tracing", and it provides benefits to programmers in the testing and debugging stages of software development by identifying the original source of data. Thus, "data tracing" provides the programmer with thorough documentation of the data flow throughout an entire system.

[0434] Standard naming conventions do not apply to certain language extensions such as the script footings that, for example, specify the size of the table to be created in a "Create Table" script.

[0435] The foregoing discussion of the invention has been presented for purposes of illustration and description. Further, comments and description is not intended to limit the invention to the form disclosed herein. Consequently, variation and modification commensurate with the above teachings, and within the skill and knowledge of the relevant art, are within the scope of the present invention. The embodiment described herein above is further intended to explain the best mode presently known of practicing the invention and to enable others skilled in the art to utilize the invention as such, or in other embodiments, and with the various modifications required by their particular application or uses of the invention. It is intended that the appended claims be construed to include alternative embodiments to the extent permitted by the prior art.

1-26. (canceled)

27. A method for processing a financial transaction on a computing system using a plurality of processors comprising:

retrieving, in response to a request to perform a financial transaction having a transaction identifier and input values, a unique subset of user-defined algorithms from a set of user-defined algorithms, wherein the unique subset is associated with the transaction identifier via a user-defined transaction processing table, each user-defined algorithm, when executed by a processor on the input values, creates or changes a data record such that the unique subset of user-defined algorithms associated with the transaction identifier, when executed using the input values, creates or changes data records stored in a database as necessary to process the financial transaction;

executing each of the user-defined algorithms of the unique subset on the input values;

wherein the unique subset of algorithms are executed concurrently on different processors and the financial transaction is considered processed when all algorithms in the unique subset of user-defined algorithms have been executed on the input values.

28. The method of claim 27, wherein each request further includes a licensee identifier and the set of user-defined algorithms is further associated with the licensee identifier.

29. The method of claim 27, wherein executing each of the user-defined algorithms of the unique subset on the input values comprises:

executing a user-defined algorithm that creates a data record in a transaction journal file;

executing a user-defined algorithm that changes a data record in a general ledger master table;

executing a user-defined algorithm that changes a data record in an account master table; and

executing a user-defined algorithm that changes a data record in an entity attribute master table.

30. The method of claim 27 further comprising:

generating a transaction sequence number for the retrieved request to perform a financial transaction.

31. The method of claim 29, wherein executing each of the user-defined algorithms of the unique subset on the input values comprises:

executing a user-defined algorithm that creates, changes or deletes a data record in a cash flow table;

executing a user-defined algorithm creates, changes or deletes a data record in an income statement table;

executing a user-defined algorithm that creates, changes or deletes a data record in a capital gains table;

executing a user-defined algorithm that creates, changes or deletes a data record in a balance sheet table; and executing a user-defined algorithm that creates, changes or deletes a data record in a performance measurement table.

32. The method of claim 27, further comprising:

if the transaction identify is not associated with a unique subset of user-defined algorithms from the set of user-defined algorithms, writing the request to perform the financial transaction to a system reject table.

33. The method of claim 27 further comprising:

if one of the algorithms fails to execute,

reversing all algorithms of the unique subset that did execute; and

writing the request to perform the financial transaction to a system reject table.

34. The method of claim 27 wherein each of the unique subset of algorithms is executed concurrently on a different one of the plurality of processors.

35. The method of claim 27 wherein at least one of the unique subset of algorithms is executed on a first processor concurrently with the execution of at least one algorithm of a different unique subset of algorithms on a different processor.

36. The method of claim 27 wherein at least one of the unique subset of algorithms is executed on a first processor concurrently with the execution of at least two algorithms of different unique subsets of algorithms on different processors.

37. A method for processing financial transactions on a multiprocessing machine having a plurality of processors, each financial transaction having financial transaction data, the method comprising:

maintaining a set of user-defined algorithms, each user-defined algorithm being independently and simultaneously processable by any one of the plurality of processors in the multiprocessing machine and each user-defined algorithm when processed on financial data adds, changes or deletes only one financial data record based on the financial data;

receiving a first financial transaction having a first user-defined financial transaction type and first financial data;

identifying, for the first financial transaction, a first unique subset of user-defined algorithms from a set of user-defined algorithms based on the first user-defined financial transaction type;

processing each user-defined algorithm in the first unique subset of user-defined algorithms on the first financial data;

receiving a second financial transaction having a second user-defined financial transaction type and second financial data;

identifying, for the second financial transaction, a second unique subset of user-defined algorithms from the set of user-defined algorithms based on the second user-defined financial transaction type;

processing each user-defined algorithm in the second unique subset of user-defined algorithms on the second financial data; and

wherein all the user-defined algorithms in the first unique subset of user-defined algorithms on the first financial data and at least one of the second unique subset of user-defined algorithms are concurrently processed.

38. The method of claim 37, wherein each user-defined algorithm changes only one financial data record in only one user-defined financial report and includes an operator, a first operand that identifies input data from the financial transaction data, and a second operand that identifies the only one data record.

39. The method of claim 37 further comprising:

maintaining a user-defined transaction processing table that associates each user-defined transaction type with a unique subset of the set of user-defined algorithms including associating the first unique subset of user-defined algorithms with the first user-defined financial transaction type and the second unique subset of user-defined algorithms with the second user-defined financial transaction type.

40. The method of claim 37, wherein processing each unique subset of the set of user-defined algorithms associated with a user-defined transaction type comprises:

executing a user-defined algorithm on a first processor that creates a data record in a transaction journal file;

concurrently executing a user-defined algorithm on a second processor that changes a data record in a general ledger master table;

concurrently executing a user-defined algorithm on a third processor that changes a data record in an account master table; and

concurrently executing a user-defined algorithm on a fourth processor that changes a data record in an entity attribute master table.

41. The method of claim 37 further comprising:

wherein the at least one of the second unique subset of user-defined algorithms is completed prior to completion of processing all the user-defined algorithms in the first unique subset of user-defined algorithms.

42. The method of claim 37 further comprising:

receiving the second financial transaction after receiving the first financial transaction; and

completing processing of all of the user-defined algorithms in the second unique subset of user-defined algorithms prior to completing the processing of all of the user-defined algorithms in the first unique subset of user-defined algorithms.

43. A financial transaction multiprocessing system for an enterprise, each financial transaction having a financial transaction type and financial transaction data, comprising:

a multiprocessing computer having a plurality of processors;

a financial record database storing financial data records;

a database manager for creating, changing and deleting financial data records;

a queue for receiving financial transactions for processing, wherein each financial transaction when processed results in adding, changing or deleting to at least one financial data record and wherein each financial transaction is associated with one of a plurality of user-defined financial transaction types and includes financial transaction data;

an algorithm database storing a set of user-defined algorithms, each user-defined algorithm capable of being processed concurrently and independently with any other user-defined algorithm and each user-defined algorithm when processed by any one of the plurality of processors causing the database manager to add, change or delete a financial data record; and

a transaction processing table that associates each user-defined transaction type with a unique subset of the set of user-defined algorithms such that processing a first financial transaction having a first financial transaction type and first financial transaction data is achieved by processing a first unique subset of user-defined algorithms associated with the first financial transaction type on the first financial transaction data.

44. The financial transaction multiprocessing system of claim 43 further comprising:

a set of control tables;

a set of user-defined control algorithms, each user-defined control algorithm when processed causing the database manager to add, change or delete one or more data records in one or more control tables; and

wherein the transaction processing table further associates one or more user-defined control algorithms with each user-defined financial transaction type.

45. The system of claim 43 further comprising:

a processor queue for user-defined algorithms, the processor queue distributing queued user-defined algorithms to processors as the processors become available.

46. The system of claim 43, wherein each user-defined algorithm includes only one operator and only a first operand and a second operand, the first operand identifying input data from the financial transaction data of the financial transaction to be processed, and the second operand identifying the only one data record.

* * * * *