Note To File
Author: Garvin H Boyle
Date: 160505

```
;;-----------------------------------------------------------------------------|
;; SECTION A – AUTHOR IDENTIFICATION AND CODE ABSTRACT
;;-----------------------------------------------------------------------------|
;;
;; File Name: CmLab_V1.xx.nlogo
;; By Orrery Software
;; Dated: 2016-03-30
;; Author contact:
;;   Garvin H Boyle
;;   orrery@rogers.com
;;   orrery-software.webs.com

;; As the author, I welcome questions, discussion of issues and suggestions
;;   for improvements.

;;-----------------------------------------------------------------------------|
;; This CmLab app is a laboratory in which students can study aspects
;;   of the proposed law of conservation of money.

;;-----------------------------------------------------------------------------|
;; SECTION B – INITIAL DECLARATIONS OF GLOBALS AND BREEDS
;;-----------------------------------------------------------------------------|
;;
;;-----------------------------------------------------------------------------|
;; This program was developed on NetLogo Version 5.0.5
;;

;;-----------------------------------------------------------------------------|
;; code-determined global variables
globals
[
  ;; The version should be coded in this global variable to be included in
  ;;   output files.
  gs-Version

  ;; Note: Some global variables are declared inside of switches, sliders and
  ;;   choosers when the interface is constructed and are not declared here.
  ;;   For the sake of clarity and completeness, they are noted here.

  ;; There are several uses of global variables:
  ;;  - Toggles (switches), and choosers which enable or disable features;
  ;;  - Numbers (in variables or sliders) which act as parameters;
  ;;  - Numbers (in variables) which collect data.
  ;;
  ;; Those marked as 'native Boolean' have values of true or false.
  ;; Those marked as 'numeric Boolean' have values of 1 or 0.

  ;;---------------------
  ;; MODELING ENVIRONMENT
  ;;---------------------

  ;; Assumed “Model Settings” on startup
  ;; horizontal wrap: on
  ;; vertical wrap: on
  ;; location of origin: centre
  ;; patch size: 9.63 pixels
```

```
;;-----------------------------------------------------------------------------|
;; Implicit global variables due to model settings – patch locations
;; min-pxcor  -15
;; max-pxcor   15
;; min-pycor  -15
;; max-pycor   15

;;----------------------------
;; SCENARIO SELECTION CONTROLS
;;----------------------------

;; gs-scenario         ;; Chooser, string converts to a scenario number
g-scenario-number      ;; scenario no., 0 or 1; interpretation of gs-scenario
;; The possible scenarios.
ge-scenario-with-prsns ;; scenario 0
ge-scenario-with-corps ;; scenario 1

;; To halt a scenario at a pre-determined tick.
;; g-halt-at-tick       ;; Has it's own input box

;; Initialize the Pseudo Random Number Generator (PRNG).
;; g-use-this-seed      ;; Slider, ( 1 <= g-use-this-seed <= 100 )

;;-----------------------------------------------
;; ECONOMIC MODEL PARAMETERS AND CONTROLS
;;-----------------------------------------------

;; SWITCHES
;; These can be turned on and off during operations.
;; They are declared in the switches, and noted here.
;; -btfs- stands for bank-to-prsns flows, and these control the way
;;   that interest collected by banks can flow back into the real
;;   economy.
;; gb-btpfs-bankruptcies     ;; Always on, set in do-pre-tick.
;; gb-btpfs-daily-purchases  ;; Banks buy but do not sell.
;; gb-btpfs-monthly-taxes    ;; All C1 assets taxed and redistributed

;; INTEREST RATES (Sliders) [min, inc, max, val]
;; Sliders can be altered during operations.
;; g-iorr    ;; Interest On Required Reserves [ 0 .1 100 2 ]
;; g-ioer    ;; Interest On Excess Reserves   [ 0 .1 100 1 ]
;; g-iosd    ;; Interest On Savings Deposits  [ 0 .1 100 1 ]
;; g-iobl    ;; Interest On Bank Loans        [ 0 .1 100 2 ]
;; TODO: Put g-docs into a % slider when Corps activated.
;; g-docs    ;; Dividends on Corporate Stocks [ 0 .1 100 2 ]

;; OTHER SLIDERS:
;; The first three can be changed at any time, but are effective only
;;   during setup.
;; g-no-of-banks-max       ;;                   [ 1   1   20    10     ]
;; g-no-of-prsns-per-bank  ;;                   [ 1   1   200   10     ]
;; g-crb-assets-per-prsn   ;; currency at start [ 100 100 10000 1000   ]
g-no-of-corps-per-bank  ;; at start          [ 1   1   20    4     ]

;; These are effective during operations.
;; g-net-worth-tax-rate    ;; Calculate taxes   [ 0 0.1   0.5   10     ]
;; g-reserve-requirement-ratio ;;               [ 1 0.1   100   20    ]

;; REALLY ADVANCED CONTROLS - PANEL 04
;; gb-bank-insurance       ;; When true, banks share loss of bankruptcy.
;; g-bankruptcy-factor     ;; Used to determine bankruptcy.

;; Derived variables:
g-no-of-banks              ;; Calculated value
;; g-no-of-banks-max       ;; A slider
g-no-of-prsns              ;; Calculated value
g-no-of-prsns-max          ;; Calculated value
g-no-of-corps              ;; Calculated value
g-no-of-corps-max          ;; Calculated value

;; Various internal global constants derived from g-crb-assets-per-prsn.
g-p-daily-cost-of-living   ;; Used to determine daily purchases.
g-p-daily-L0-allocation    ;; Used to determine daily cash purchases.
g-p-daily-L1-allocation    ;; Used to determine daily purchases by check.
g-p-standard-loan          ;; Used to set up loans.
g-p-standard-loan-payment  ;; Used to pay principal on loans.
g-minimum-vault-cash       ;; Used to manage reserves

;;-------------------------------------
;; END OF MODEL PARAMETERS AND CONTROLS
;;-------------------------------------

;;-------------------------------------
;; DATA COLLECTION AND DISPLAY CONTROLS
;;-------------------------------------

;; The following global variables are not model controls or paramaters,
;;   but, rather, are variables used to collect data about the model
;;   for display in the user interface, in some fashion (monitors or plots),
;;   or used to manage all of the debug routines and output.

;; DATA COLLECTION

;; In the following I use "debts" to mean "liabilities".
;; Money supplies
g-msi-ttl-assets        ;; Money supply I, Physical money supply.
g-msii-ttl-assets       ;; Money supply II, Logical money supply.
g-msiii-ttl-assets      ;; Money supply III, Shadow money supply.
g-msi-ttl-debts         ;; Money supply I, Physical money supply.
g-msii-ttl-debts        ;; Money supply II, Logical money supply.
g-msiii-ttl-debts       ;; Money supply III, Shadow money supply.
g-msi-net               ;; Money supply I, Net money
g-msii-net              ;; Money supply II, Net money
g-msiii-net             ;; Money supply III, Net money

;; Money Categories - by money supply.
;; MS-I - The money base - Physical money supply.
g-msi-prsn-P0-cash      ;; cash in circulation - assets
g-msi-corp-P0-cash      ;; cash in circulation - assets
g-msi-bank-vc           ;; bank vault cash - assets
g-msi-bank-rr-assets    ;; bank required reserves - assets
g-msi-bank-er-assets    ;; bank excess reserves - assets
g-msi-bank-rr-debts     ;; bank required reserves - assets
g-msi-bank-er-debts     ;; bank excess reserves - assets
g-msi-crb-L0-assets     ;; money base logical endowment
g-msi-crb-P0-assets     ;; money base physical endowment
g-msi-crb-L0-debts      ;; money base logical endowment
g-msi-crb-P0-debts      ;; money base physical endowment
```

```
g-msi-crb-rr            ;; CRB required reserves - debts
g-msi-crb-er            ;; CRB excess reserves - debts

;; MS-II - The logical money supply.
g-msii-prsn-L0-cash     ;; cash in circulation, overlaps with MS-I.
g-msii-corp-L0-cash     ;; cash in circulation, overlaps with MS-I.
g-msii-crb-C1-assets    ;; private corp level debts
;; xx g-msii-crb-c2-assets ;; private corp level assets

g-msii-gcra-L1-assets   ;; govt checking assets
g-msii-gcra-L1-loan-debts ;; govt loan debts
;; xx g-msii-gcra-L2-assets ;; govt savings assets
;; ss g-msii-gcra-L3-debts    ;; govt bond debts

g-msii-bank-L1-assets   ;; bank checking assets
g-msii-bank-L1-loan-assets ;; bank loan assets
g-msii-bank-L1-debts    ;; bank checking debts
g-msii-bank-L2-assets   ;; bank savings assets
g-msii-bank-L2-debts    ;; bank savings debts
;; ss g-msii-bank-L3-assets   ;; bank bond assets
g-msii-bank-C1-assets   ;; private L1 checking assets
;; g-msii-bank-c2-assets   ;; private L2 savings assets

g-msii-prsn-L1-assets   ;; prsn checking assets
g-msii-prsn-L1-loan-debts ;; prsn loan debts
g-msii-prsn-L2-assets   ;; prsn savings assets
;; ss g-msii-prsn-L3-assets   ;; prsn bond assets
;; ss g-msii-prsn-L4-assets   ;; prsn bond assets

g-msii-corp-L1-assets   ;; corp checking assets
g-msii-corp-L1-loan-debts ;; corp loan debts
g-msii-corp-L2-assets   ;; corp savings assets
;; ss g-msii-corp-L3-assets   ;; corp bond assets
;; ss g-msii-corp-L3-debts    ;; corp bond debts
;; ss g-msii-corp-L4-assets   ;; corp bond assets
;; ss g-msii-corp-L4-debts    ;; corp bond debts

;; MS-III - The shadow money supply.
g-msiii-crb-S1-rrip-debts  ;; interest payable on rr - debts
g-msiii-crb-S1-erip-debts  ;; interest payable on er - debts
g-msiii-gcra-S1-L1ip-debts ;; govt interest payable on loan - debts
;; ss g-msiii-gcra-S1-L3ip-debts   ;; govt interest payable on bonds - debts
g-msiii-bank-S1-L1ir-assets  ;; bank interest receivable on loans - assets
g-msiii-bank-S1-L2ip-debts   ;; bank interest payable on savings - debts
g-msiii-bank-S1-rrir-assets  ;; bank interest receivable on rr - assets
g-msiii-bank-S1-erir-assets  ;; bank interest receivable on er - assets
g-msiii-prsn-S1-L1ip-debts   ;; prsn interest payable on L1 loans - debts
g-msiii-prsn-S1-L1tp-debts   ;; prsn 30day total payables - debts
g-msiii-prsn-S1-L1tr-assets  ;; prsn 30day total receivables - assets
g-msiii-prsn-S1-L2ir-assets  ;; prsn interest receivable on savings - assets
;; ss g-msiii-prsn-S1-L3ir-assets  ;; prsn interest receivable on bonds - assets
;; ss g-msiii-prsn-S1-L4dr-assets  ;; prsn dividend receivable on stocks - assets
g-msiii-corp-S1-L1tp-debts   ;; corp 30day total payables - debts
g-msiii-corp-S1-L1tr-assets  ;; corp 30day total receivables - assets
g-msiii-corp-S1-L2ir-assets  ;; corp interest receivable on savings - assets
;; ss g-msiii-corp-S1-L3ip-assets  ;; corp interest payable on bonds - debts
;; ss g-msiii-corp-S1-L4dp-assets  ;; corp dividend payable on stocks - debts

;; Public funds in trust vs Private funds
g-crb-P0-assets        ;; In public trust
g-crb-publ-assets      ;; In public trust
g-crb-priv-assets      ;; Profit/Loss related
```

```
  g-crb-publ-debts       ;; In public trust
  g-crb-priv-debts       ;; Profit/Loss related
  g-crb-publ-net-worth   ;; In public trust
  g-crb-priv-net-worth   ;; Profit/Loss related

  g-gcra-P0-assets       ;; In public trust
  g-gcra-publ-assets     ;; In public trust
  g-gcra-priv-assets     ;; Profit/Loss related
  g-gcra-publ-debts      ;; In public trust
  g-gcra-priv-debts      ;; Profit/Loss related
  g-gcra-publ-net-worth  ;; In public trust
  g-gcra-priv-net-worth  ;; Profit/Loss related

  g-bank-P0-assets       ;; In public trust
  g-bank-publ-assets     ;; In public trust
  g-bank-priv-assets     ;; Profit/Loss related
  g-bank-publ-debts      ;; In public trust
  g-bank-priv-debts      ;; Profit/Loss related
  g-bank-publ-net-worth  ;; In public trust
  g-bank-priv-net-worth  ;; Profit/Loss related

  g-prsn-P0-assets       ;; In public trust
  g-prsn-publ-assets     ;; In public trust
  g-prsn-priv-assets     ;; Profit/Loss related
  g-prsn-publ-debts      ;; In public trust
  g-prsn-priv-debts      ;; Profit/Loss related
  g-prsn-publ-net-worth  ;; In public trust
  g-prsn-priv-net-worth  ;; Profit/Loss related

  g-corp-P0-assets       ;; In public trust
  g-corp-publ-assets     ;; In public trust
  g-corp-priv-assets     ;; Profit/Loss related
  g-corp-publ-debts      ;; In public trust
  g-corp-priv-debts      ;; Profit/Loss related
  g-corp-publ-net-worth  ;; In public trust
  g-corp-priv-net-worth  ;; Profit/Loss related

  ;; DATA DISPLAY - Histogram axes
  g-agents-nw-xaxis-min  ;; Minimum value on prsn net worth histogram.
  g-agents-nw-xaxis-max  ;; Maximum value on prsn net worth histogram.
  g-prsns-nw-xaxis-min   ;; Minimum value on prsn net worth histogram.
  g-prsns-nw-xaxis-max   ;; Maximum value on prsn net worth histogram.
  g-banks-nw-xaxis-min   ;; Minimum value on prsn net worth histogram.
  g-banks-nw-xaxis-max   ;; Maximum value on prsn net worth histogram.
  g-banks-P0-xaxis-min   ;; Minimum value on P0-all-assets.
  g-banks-P0-xaxis-max   ;; Maximum value on P0-all-assets.
  g-banks-P0-all-assets-min  ;; Minimum value on P0-all-assets.
  g-banks-P0-all-assets-mean ;; Mean value on P0-all-assets.
  g-banks-P0-all-assets-max  ;; Max value on P0-all-assets.

  ;; DATA DISPLAY - Line Graphs
  g-max-net-worth-priv-prsns   ;; What it says.
  g-mean-net-worth-priv-prsns  ;; What it says.
  g-min-net-worth-priv-prsns   ;; What it says.
  g-max-net-worth-priv-banks   ;; What it says.
  g-mean-net-worth-priv-banks  ;; What it says.
  g-min-net-worth-priv-banks   ;; What it says.

  ;; DATA DISPLAY - Event Counts
  g-counts-loans
  g-counts-p-deaths
  g-counts-p-births
  g-counts-b-deaths
  g-counts-b-births

  ;;---------------
  ;; DEBUG CONTROLS
  ;;---------------

  gb-debug-on                 ;; Numeric Boolean, opens debug log file, 0 or 1.
  gs-debug-status             ;; for monitor, '1 (On)' or '0 (Off)',
  ;; gs-debug-step-chooser    ;; Chooser, used with gb-debug-flow-on
  gb-debug-flow-on            ;; Numeric Boolean, in association with chooser,
  gs-log-file-name            ;; name of the debug log file
                              ;;   opens flow to log file
  ;; gb-debug-show-steps      ;; Switch, Native Boolean, show in command centre
]


;;-----------------------------------------------------------------------------|
;; Attributes of patches
patches-own
[
  ;; BUILT-IN ATTRIBUTES
  ;; pxcor        ;; min-pxcor <= pxcor < max-pxcor
  ;; pycor        ;; min-pxcor <= pxcor < max-pxcor
  ;; pcolor       ;; color of this patch ( 0 <= color < 140 )
  ;; plabel       ;; label of this patch
  ;; plabel-color ;; color of this patch's label ( 0 <= label-color < 140 )

  ;; CmLab-DETERMINED ATTRIBUTES
  ;; Nil.
]

;;-----------------------------------------------------------------------------|
;; Attributes of links
;;-----------------------------------------------------------------------------|
;; nil
;; I don't understand links and did not use any.

;;-----------------------------------------------------------------------------|
;; THEORY:  ATTRIBUTES WITH MONEY SUPPLY DESIGNATORS
;;          P0, L0, L1, L2, L3, L4, S1, C1.
;;          REPLACING M0, M1, M2, M3, M4.
;;-----------------------------------------------------------------------------|
;; WARNING -   I am NOT using the Mx designations as they are used in the
;;             the real world - for two reasons.
;;             1.  In the real world M4 includes M3, M3 includes M2, etc. until
;;                 the end where M1 includes M0.  For me, each category of money
;;                 is independent of the other.  It's easier to track.  The real
;;                 world meaning can be recovered simply by adding the included
;;                 data, at your choice.  So I use L0, L1, L2, ... and P0.
;;             2.  No two countries seem to have the same definitions for each
;;                 of the categories of money, so I do not try to accurately
;;                 simulate or replicate that money supply structure of any one
;;                 country, but, rather, I abstract a simplified model that is
;;                 relatively close to all of them.
;;
;;          In addition, I use C1 and S1 as special temporary designators.
;;
;; Which agents can hold which types of assets and debts is a bit of
;;   a tricky question.  I have resolved it this way.
;;
;; L0 assets - only prsns and corps can use cash.  All others make payments by
;;             check.  L0 assets are in the wallets of prsns and corps.
;; P0-assets - this is physical part of currency, stored in wallets and vaults.
;;             P0 savings accounts are the only investment option for commercial
;;             banks, but are called P0-RR and P0-ER deposits, with the CRB.
;;             Prsns and Corps hold P0-assets in their wallets.
;; L0-debts -  don't really exist.  They become L1 debts.
;; L1-assets - checking accounts are the work horse of this economy.  All agents
;;             have checking accounts.  They accept L1 payments into their
;;             L1 checking account and make L1 payments out of it.  In the case
;;             of the CRB or commercial banks, it is called C1-assets, to
;;             distinguish those accounts held in public trust from those that
;;             function as their private funds.  The CRB's C1-assets are a
;;             part of the GCRA L1-assets and get merged there regularly.
;; L1-loan-assets  - Commercial banks are the only ones that can provide loans.
;;             The loans stick with the borrower and the bank until they are paid
;;             off.  The loans are also the primary means for expanding the
;;             MS-II money supply, using a pair of double-entry records.
;;             When a loan is "signed" in two copies it creates a liability
;;             for the borrower and an asset for the lender.  Then the money
;;             is created by entering an L1 liability for the bank, and an L1
;;             asset for the borrower.  The two double-entries, or four entries
;;             in total, represent the loan.  No net worth is altered by such
;;             an event since the entries counter-balance each other.
;;                 Any payment that alters the networth of participants involves
;;             two entries that do not counter-balance.  When a payment is
;;             made on a loan, it requires two double-entries (four entries)
;;             that counter-balance again to record the payment. Again, no
;;             change in networth of either party happens, but the MS-II money
;;             supply constracts again.
;; L1-debts -  For commercial banks, this is the hind end of L1-assets and
;;             C1-assets.  Non-bank agents (GCRA, CRB, prsns, corps) have no
;;             need of these.  The sum of all explicit bank L1-debts is the
;;             standard money supply (MS-II).
;; L1-loan-debts - This is the second entry of the four that are required
;;             to record a loan.  This and the L1-loan-assets must always be
;;             incremented or decremented by matching records, indicating
;;             the expansion or reduction of the MS-II money supply.  Chartered
;;             banks do not have loan debts.  Their clients do.  I.e. loan
;;             debts are for prsns, corps, and the GCRA.
;;
;; Other L1-type assets - all receivables are S1-type assets.
;; Other L1-type debts  - all payables are S1-type debts.
;;             S1-type money is convertible to L1-type money when paid.
;;
;; L2-assets - L2 savings accounts are the primary investment option for agents
;;             other than banks.  GCRA, prsns and corps may hold L2-assets.
;; L2-debts -  only banks hold L2-debts.
;;
;; TODO: Beyond L2 nothing has been implemented.
;; In the real world M3 and M4 are more and more broad designations.  In this
;;   program I have changed that.  L3 are bonds.  L4 are stocks.
;;
;; L3-assets - these are the assets of bond buyers/holders.  That might include
;;             prsns and corps.
;; L3-debts -  these are the debts of bond sellers.  That includes
;;             The GCRA, banks and corps.
;;
;; L4-assets - these are the assets of stock buyers/holders.  That might include
;;             prsns and corps.
;; L4-debts -  these are the debts of stock sellers.  That includes
;;             only the corps.
;;
```

```
;; All interest on savings deposits (with CRB or banks), on bonds, on loans, or
;;             all dividends, are S1-type assets and debts, convertible to
;;             L1-type money when paid.
;;
;; C1-assets and C2-assets - both the CRB and chartered banks have a dual role.
;;             In the "back room" role they guard the public trust by ensuring
;;             that money is properly conserved at the level of client-to-client
;;             transactions.  In the "front room" role they are organizations
;;             that charge fees for financial services.  The net worth of the
;;             back room must always be zero.  The net worth of the front room
;;             is where corporate profits and losses are recorded.  The back
;;             room staff may have many "clients" consisting of prsns and corps,
;;             but they have one special client, which is their own front room
;;             organization.
;;             Each client must maintain its own checking and savings bank books
;;             (in the variables L1-assets and L2-assets.  The front room
;;             client must also keep such records separate from back room assets,
;;             which would also be in variables of the same name.  So the front
;;             room assets I have designated as C1-assets and C2-assets.
;;
;; S1-assets and S1-debts - those persistent debts that exist unpaid for a
;;             duration longer than the moment required to create them are
;;             part of the shadow money supply and are designated as S1-type.
;;             In some sense, I mean the shadow money supply to be that part of
;;             the money supply that is invisible to the governing monetary
;;             architecture (i.e. the CRB and its chartered banks), and I still
;;             think that is the best definition for a real-world system.  But
;;             for this model I have implemented the shadow money supply as
;;             all such persistent debts, excluding only the persistent debts
;;             associated with L1-loans from chartered banks.  Double-entry
;;             book-keeping still applies: for every S1-debt created a counter-
;;             balancing S1-asset is also created.
;; TODO: when stocks and bonds are implemented as part of the activation of
;;             corps, they will be in the shadow money supply, and I may change
;;             the implementation to be more consistent with the "visibility"
;;             criterion.

;;----------------------------------------------------------------------------|
;; Turtles and breeds
;;----------------------------------------------------------------------------|

breed [ GCRAs GCRA ]
breed [ CRBs  CRB  ]
breed [ banks bank ]
breed [ prsns prsn ]
breed [ corps corp ]

;;----------------------------------------------------------------------------|
;; Attributes of GCRAs (Government Consolidated Revenue Accounts)
GCRAs-own
[
  ;; BUILT-IN ATTRIBUTES
  ;; who         ;; fixed id number
  ;; breed       ;; to which breed this turtle belongs [GCRA]
  ;; heading     ;; 0 <= heading < 360, 0 = north
  ;; xcor        ;; min-pxcor <= xcor < max-pxcor
  ;; ycor        ;; min-pxcor <= xcor < max-pxcor
  ;; size        ;; size relative to a patch, default is 1
  ;; shape       ;; a shape chosen from the shape library
  ;; color       ;; color of this turtle ( 0 <= color < 140 )
  ;; pen-mode    ;; "up" or "down"
  ;; pen-size    ;; in pixels
  ;; hidden?     ;; true or false
  ;; label       ;; label of this turtle
  ;; label-color ;; color of this turtle's label ( 0 <= label-color < 140 )

  ;; USER-DETERMINED ATTRIBUTES
  ;; Associated with GCRA dynamics.
  default-colour       ;; as it says
  bank-who             ;; bank that holds the loan
  L1-assets            ;; assets of the government
  L1-loan-debts        ;; debts of the government (bank loans)
  S1-L1ip-debts        ;; interest payable on L1 loan

  ;; xx L2-assets           ;; savings of the government

  ;; ss L3-debts            ;; debts of the government - bonds
  ;; ss S1-L3ip-debts       ;; payable on bonds

  ttl-P0-assets     ;; aggregate of all physical assets
  ttl-publ-assets   ;; aggregate of all public assets
  ttl-publ-debts    ;; aggregate of all public debts
  ttl-priv-assets   ;; aggregate of all private assets
  ttl-priv-debts    ;; aggregate of all private debts
  net-worth-publ    ;; total public assets minus debts
  net-worth-priv    ;; total private assets minus debts

  ;; Money supply aggregates
  msi-assets        ;; Physical money supply
  msi-debts         ;; Physical money supply
  msii-assets       ;; Logical money supply
  msii-debts        ;; Logical money supply
  msiii-assets      ;; Shadow money supply
  msiii-debts       ;; Shadow money supply
]

;;-----------------------------------------------------------------------------|
;; Attributes of CRBs (Central Reserve Banks)
CRBs-own
[
  ;; BUILT-IN ATTRIBUTES
  ;; who         ;; fixed id number
  ;; breed       ;; to which breed this turtle belongs [CRB]
  ;; heading     ;; 0 <= heading < 360, 0 = north
  ;; xcor        ;; min-pxcor <= xcor < max-pxcor
  ;; ycor        ;; min-pxcor <= xcor < max-pxcor
  ;; size        ;; size relative to a patch, default is 1
  ;; shape       ;; a shape chosen from the shape library
  ;; color       ;; color of this turtle ( 0 <= color < 140 )
  ;; pen-mode    ;; "up" or "down"
  ;; pen-size    ;; in pixels
  ;; hidden?     ;; true or false
  ;; label       ;; label of this turtle
  ;; label-color ;; color of this turtle's label ( 0 <= label-color < 140 )

  ;; USER-DETERMINED ATTRIBUTES
  ;; Associated with CRB dynamics.
  default-colour       ;; as it says
  P0-assets            ;; physical assets of the CRB
  L0-assets            ;; logical assets of the CRB
  P0-debts             ;; physcial debts of the CRB
  L0-debts             ;; logical debts of the CRB
  P0-rr-assets         ;; required reserves of all banks
  P0-er-assets         ;; excess reserves of all banks
```

```
  ;; Associated with corporate bank dynamics.
  bank-who             ;; chartered bank that holds C1 account.
  S1-rrip-debts        ;; interest payable on required reserves - debts
  S1-erip-debts        ;; interest payable on excess reserves - debts
  C1-assets            ;; corporate bank equivalent of L1-assets
  ;; xx c2-assets            ;; corporate bank equivalent of L2-assets

  ttl-P0-assets        ;; aggregate of all physical assets
  ttl-publ-assets      ;; aggregate of all public assets
  ttl-publ-debts       ;; aggregate of all public debts
  ttl-priv-assets      ;; aggregate of all private assets
  ttl-priv-debts       ;; aggregate of all private debts
  net-worth-publ       ;; total public assets minus debts
  net-worth-priv       ;; total private assets minus debts

  ;; Money supply aggregates
  msi-assets        ;; Physical money supply
  msi-debts         ;; Physical money supply
  msii-assets       ;; Logical money supply
  msii-debts        ;; Logical money supply
  msiii-assets      ;; Shadow money supply
  msiii-debts       ;; Shadow money supply
]

;;-----------------------------------------------------------------------------|
;; Attributes of banks (deposit-taking banks)
banks-own
[
  ;; BUILT-IN ATTRIBUTES
  ;; who         ;; fixed id number
  ;; breed       ;; to which breed this turtle belongs [bank]
  ;; heading     ;; 0 <= heading < 360, 0 = north
  ;; xcor        ;; min-pxcor <= xcor < max-pxcor
  ;; ycor        ;; min-pxcor <= xcor < max-pxcor
  ;; size        ;; size relative to a patch, default is 1
  ;; shape       ;; a shape chosen from the shape library
  ;; color       ;; color of this turtle ( 0 <= color < 140 )
  ;; pen-mode    ;; "up" or "down"
  ;; pen-size    ;; in pixels
  ;; hidden?     ;; true or false
  ;; label       ;; label of this turtle
  ;; label-color ;; color of this turtle's label ( 0 <= label-color < 140 )

  ;; USER-DETERMINED ATTRIBUTES
  ;; Associated with book-keeping bank dynamics.
  default-colour       ;; as it says
  b-bank-can-make-loans ;; boolean - 0 or 1
  b-bank-is-bankrupt    ;; boolean - 0 or 1

  L1-assets            ;; assets in checking accounts
  L1-loan-assets       ;; assets associated with a loan
  L1-debts             ;; debts in checking accounts
  S1-L1ir-assets       ;; interest receibable on L1 loans - C1-assets

  L2-assets            ;; assets in savings accounts
  L2-debts             ;; debts in savings accounts
  S1-L2ip-debts        ;; on savings deposits

  ;; ss L3-assets            ;; assets in bonds
  ;; ss L3-debts             ;; debts in bonds
  crb-who              ;; central reserve bank
  P0-vc-assets         ;; $c in the vault - assets
  P0-er-assets         ;; excess reserves - assets
  P0-er-debts          ;; excess reserves - debts
  P0-rr-assets         ;; required reserves - assets
  P0-rr-debts          ;; required reserves - debts
  P0-all-assets        ;; An aggregate of VC, ER and RR.

  ;; Associated with corporate bank dynamics.
  no-of-prsn-clients   ;; How many clients currently
  no-of-corp-clients   ;; How many clients currently
  no-of-gcra-clients   ;; How many clients currently
  no-of-crb-clients    ;; How many clients currently
  S1-rrir-assets       ;; interest on required reserves
  S1-erir-assets       ;; interest on excess reserves
  C1-assets            ;; corporate bank equivalent of L1-assets
  ;; c2-assets            ;; corporate bank equivalent of L2-assets

  ttl-P0-assets        ;; aggregate of all physical assets
  ttl-publ-assets      ;; aggregate of all public assets
  ttl-publ-debts       ;; aggregate of all public debts
  ttl-priv-assets      ;; aggregate of all private assets
  ttl-priv-debts       ;; aggregate of all private debts
  net-worth-publ       ;; total public assets minus debts
  net-worth-priv       ;; total private assets minus debts

  ;; Money supply aggregates
  msi-assets        ;; Physical money supply
  msi-debts         ;; Physical money supply
  msii-assets       ;; Logical money supply
  msii-debts        ;; Logical money supply
  msiii-assets      ;; Shadow money supply
  msiii-debts       ;; Shadow money supply
]

;;-----------------------------------------------------------------------------|
;; Attributes of prsns (non-corporate economic agents)
prsns-own
[
  ;; BUILT-IN ATTRIBUTES
  ;; who         ;; fixed id number
  ;; breed       ;; to which breed this turtle belongs [prsn]
  ;; heading     ;; 0 <= heading < 360, 0 = north
  ;; xcor        ;; min-pxcor <= xcor < max-pxcor
  ;; ycor        ;; min-pxcor <= xcor < max-pxcor
  ;; size        ;; size relative to a patch, default is 1
  ;; shape       ;; a shape chosen from the shape library
  ;; color       ;; color of this turtle ( 0 <= color < 140 )
  ;; pen-mode    ;; "up" or "down"
  ;; pen-size    ;; in pixels
  ;; hidden?     ;; true or false
  ;; label       ;; label of this turtle
  ;; label-color ;; color of this turtle's label ( 0 <= label-color < 140 )

  ;; USER-DETERMINED ATTRIBUTES
  ;; Associated with prsn dynamics.
  default-colour       ;; as it says
  b-prsn-is-bankrupt   ;; boolean - 0 or 1

  L0-assets            ;; assets of the prsn - logical
  P0-assets            ;; assets of the prsn - physical
```

```
  bank-who             ;; bank that holds the loan
  L1-assets            ;; assets in checking accounts
  L1-loan-debts        ;; debts associated with loans
  S1-L1ip-debts        ;; payable on bank loans - debts
  payables-30day       ;; debts to be paid in 30 days
  S1-30day-total-debts ;; sum of 30-day payables
  S1-30day-total-assets ;; sum of 30-day receivables

  L2-assets            ;; assets in savings accounts
  S1-L2ir-assets       ;; interest on savings accounts

  ;; ss L3-corpwho           ;; Holds a bond with this corp
  ;; ss L3-assets            ;; assets in bonds
  ;; ss S1-L3ir-assets ;; receivable on bond

  ;; ss L4-corpwho           ;; Holds a stock with this corp
  ;; ss L4-assets            ;; assets in stocks
  ;; ss L4-dividend-receivable ;; receivable on stocks

  ttl-P0-assets        ;; aggregate of all physical assets
  ttl-publ-assets      ;; aggregate of all public assets
  ttl-publ-debts       ;; aggregate of all public debts
  ttl-priv-assets      ;; aggregate of all private assets
  ttl-priv-debts       ;; aggregate of all private debts
  net-worth-publ       ;; total public assets minus debts
  net-worth-priv       ;; total private assets minus debts

  ;; Money supply aggregates
  msi-assets        ;; Physical money supply
  msi-debts         ;; Physical money supply
  msii-assets       ;; Logical money supply
  msii-debts        ;; Logical money supply
  msiii-assets      ;; Shadow money supply
  msiii-debts       ;; Shadow money supply
]

;;-----------------------------------------------------------------------------|
;; Attributes of corps (corporate economic agents)
corps-own
[
  ;; BUILT-IN ATTRIBUTES
  ;; who         ;; fixed id number
  ;; breed       ;; to which breed this turtle belongs [corp]
  ;; heading     ;; 0 <= heading < 360, 0 = north
  ;; xcor        ;; min-pxcor <= xcor < max-pxcor
  ;; ycor        ;; min-pxcor <= xcor < max-pxcor
  ;; size        ;; size relative to a patch, default is 1
  ;; shape       ;; a shape chosen from the shape library
  ;; color       ;; color of this turtle ( 0 <= color < 140 )
  ;; pen-mode    ;; "up" or "down"
  ;; pen-size    ;; in pixels
  ;; hidden?     ;; true or false
  ;; label       ;; label of this turtle
  ;; label-color ;; color of this turtle's label ( 0 <= label-color < 140 )

  ;; USER-DETERMINED ATTRIBUTES
  ;; Associated with corp dynamics.
  default-colour       ;; as it says
  b-corp-is-bankrupt   ;; boolean - 0 or 1

  L0-assets            ;; assets of the corp - logical
  P0-assets            ;; assets of the corp - physical
  bank-who             ;; Does banking with this bank
  L1-assets            ;; assets in checking accounts
  L1-loan-debts        ;; debts associated with loans
  S1-L1ip-debts        ;; payable on bank loans
  payables-30day       ;; debts payable in 30 days
  S1-30day-total-debts ;; sum of 30 day payables
  S1-30day-total-assets ;; sum of 30 day receivables

  L2-assets            ;; assets in savings accounts
  S1-L2ir-assets       ;; interest on savings accounts

  ;; ss no-of-bond-clients   ;; prsns owning bonds
  ;; ss L3-assets            ;; assets in bonds
  ;; ss L3-debts             ;; debts in bonds
  ;; ss S1-L3ip-debts        ;; payable on bond

  ;; ss no-of-stock-clients  ;; prsns owning stocks
  ;; ss L4-assets            ;; assets in stocks
  ;; ss L4-debts             ;; debts in stocks
  ;; ss S1-L4dp-debts  ;; payable-on-stocks

  ttl-P0-assets        ;; aggregate of all physical assets
  ttl-publ-assets      ;; aggregate of all public assets
  ttl-publ-debts       ;; aggregate of all public debts
  ttl-priv-assets      ;; aggregate of all private assets
  ttl-priv-debts       ;; aggregate of all private debts
  net-worth-publ       ;; total public assets minus debts
  net-worth-priv       ;; total private assets minus debts

  ;; Money supply aggregates
  msi-assets        ;; Physical money supply
  msi-debts         ;; Physical money supply
  msii-assets       ;; Logical money supply
  msii-debts        ;; Logical money supply
  msiii-assets      ;; Shadow money supply
  msiii-debts       ;; Shadow money supply
]

;;-----------------------------------------------------------------------------|
;; SECTION C – INITIALIZATION OR SETUP PROCEDURE( S )
;;-----------------------------------------------------------------------------|

;;-----------------------------------------------------------------------------|
;; The 'autostart' startup routine
to startup
  ;; This routine is to be executed by the observer.

  ;; The manual describes this routine as follows:
  ;; This procedure, if it exists, will be called when a model is first loaded in
  ;;   the NetLogo application.  Startup does not run when a model is run headless
  ;;   from the command line, or by parallel BehaviorSpace.

  ;; On loading the model, the debug feature is always off.
  set gb-debug-on 0
  set gs-debug-status "0 (Off)"

  ;; On loading the model, the choosers, switches and sliders are
  ;;   always reset to the values that are known to work.  Only the chooser
  ;;   for the scenario is not reset.  The last saved
  ;;   selection of scenario is persistant.  This allows the 'Reset Defaults'
  ;;   button to NOT reset the scenario.
```

```
  f-reset-default-parameters

  ;; Run the setup routine to initialize other globals.
  ;; End of startup
end

;;-----------------------------------------------------------------------------|
;; Reset the debug values for the interface-declared items.
to f-reset-debug-parameters
  ;; The observer executes this routine.

  ;; I only reset here the ones that differ for a debug run.c
  set g-no-of-banks-max           4
  set g-no-of-prsns-per-bank      2
  set g-reserve-requirement-ratio 40
  set g-bankruptcy-factor         1.5

  ;; Run the setup routine to initialize other globals.
  ;; End of f-reset-debug-parameters
end

;;-----------------------------------------------------------------------------|
;; Reset the default values for the interface-declared items.
to f-reset-default-parameters
  ;; The observer executes this routine.

  ;; Switches, sliders and choosers implicitly declare global variables.  The
  ;;   values in these variables are parameters for the model, and many
  ;;   combinations of those parameters are not sustainable.  However, the
  ;;   values in those user interface devices are stored with the model and
  ;;   are persistant across a save/load action.  The default values must
  ;;   be reset on load, or available to a user as a parameter set.  The
  ;;   purpose of this routine is to store at least one viable set of
  ;;   parameter values.

  ;; To be clear, variables declared in the interface should be initialized
  ;;   here and not in the setup procedure.  They will be reset on startup
  ;;   (i.e. on load) but not on "Setup".  A separate "Reset" button is on the
  ;;   interface to enable the user to reset these at will.  Any interface-
  ;;   declared variable (as opposed to those declared in the "globals"
  ;;   block) not included here will be persistent through a save/load
  ;;   action.

  ;;-----------------------------------------------
  ;; CHOOSERS, SWITCHES AND SLIDERS
  ;;-----------------------------------------------

  ;; Initialize the chooser.
  set gs-scenario "Prsns Only"

  ;; Initialize the Pseudo Random Number Generator (PRNG).
  set g-use-this-seed 7

  ;; Interest sliders
  set g-iorr 2
  set g-ioer 1
  set g-iosd 1
  set g-iobl 2
  ;; set g-docs 2

  ;; Other startup and operations sliders
  set g-crb-assets-per-prsn       3000
  set g-no-of-banks-max           20
  set g-no-of-prsns-per-bank      20
  set g-no-of-corps-per-bank      1
  set g-net-worth-tax-rate        0.5
  set g-reserve-requirement-ratio 20
  set g-bankruptcy-factor         2

  ;; Switches
  set gb-plot-data                true
  set gb-btpfs-bankruptcies       true
  set gb-btpfs-daily-purchases    false
  set gb-btpfs-monthly-taxes      false
  set gb-bank-insurance           true
end

;;------------------------------------------------------------------------------|
;; The setup button(s)
to setup
  ;; This routine is to be executed by the observer.

  ;; NOTE: The contents of switches, sliders, and choosers seem to be
  ;;   immune to these 'clear' commands.
  clear-ticks
  clear-turtles
  clear-patches
  clear-drawing
  clear-all-plots
  clear-output
  ;; clear-globals   ;; Suppressed to make gb-debug-on value persistent.
  ;; NOTE: Instead of 'clear-globals', you must ensure all globals are
  ;;   initialized properly in 'setup'.

  ;; import-drawing "01-B OrrSW.jpg"

  ;; The version should be coded in this global variable to be included in
  ;;   output files.
  set gs-Version "CmLab_V1.17"

  ;; Debug features may be off or on depending on history.
  ;;   - Perhaps 'setup' was called by 'to Startup'.
  ;;   - Perhaps 'setup' was called during a 'BehaviorSpace' run.
  ;;   - Perhaps 'setup' was called by a user-pushed 'setup' button.
  ;; Setup needs to handle some quasi-persistant values correctly regardless of
  ;;   the history.  For gb-debug-on, in particular, I want it to be
  ;;   persistant so I can have debug output from the 'setup' routine routed
  ;;   to the debug log file, or to the command centre.

  ;; 'startup' automatically sets gb-debug-on to 0 when the application is first
  ;;   loaded.  I want to be able to (A) toggle debug on, then, (B) press
  ;;   'setup' and watch the debug output of the 'setup' command.  The gb-debug-on
  ;;   must be persistant through the above 'clear' commands.  The debug log
  ;;   file name and status, however, should not be persistent and must be
  ;;   reset when setup runs, if appropriate.
  ifelse ( gb-debug-on = 1 )
  [
    ;; Debug is on due to user setting, so file name and status should be
    ;;   reset.  I do this by turn the feature off then on.
    ;; First toggle it off, closing any remnant log file, if needed.
    f-toggle-debug
    ;; Then toggle it back on, opening a new time-stamped log file.
    f-toggle-debug
  ]
```

```
  ;; else
  [
    ;; Debug is off, possibly due to startup execution, possibly due to user
    ;;   choice.
    ;; Ensure associated variables have compatible settings.
    set gb-debug-on 0              ;; Redundant but ensures consistency.
    set gs-debug-status "0 (Off)"  ;; Redundant but ensures consistency.
    set gb-debug-flow-on 0         ;; Step-specific flow is off.
    file-close-all                 ;; Close the debug log file.
    set gs-log-file-name "dummyname"
  ]

  ;; Now, do the standard check that is done at the start of each debuggable
  ;;   routine.  This must follow the clear commands, which reset everything
  ;;   except globals, switches, sliders and choosers.
  if( gb-debug-on = 1 )
  [
    ifelse( ( gs-debug-step-chooser = "all" ) or ( gs-debug-step-chooser = "setup" )
)
    [ set gb-debug-flow-on 1 LOG-TO-FILE "" LOG-TO-FILE word "Do-setup: Debug on;
tick = " 0 ]
    [ set gb-debug-flow-on 0 ]
  ]

  ;; g-use-this-seed comes from a slider, and is persistant.
  random-seed g-use-this-seed      ;; Tells the PRNG to use this seed.

  ;; Override the scenario chooser.
  set gs-scenario "Prsns Only"
  f-set-scenario-number

  ;; SETUP FOR CONSERVEMONEYLAB
  LOG-TO-FILE ( "  INTEREST RATES (Sliders):" )
  LOG-TO-FILE ( word  "  Int. on Required Reserves --- " g-iorr " %" )
  LOG-TO-FILE ( word  "  Int. on Excess Reserves ----- " g-ioer " %" )
  LOG-TO-FILE ( word  "  Int. on Savings Deposits ---- " g-iosd " %" )
  LOG-TO-FILE ( word  "  Int. on Bank Loans ---------- " g-iobl " %" )
  ;; LOG-TO-FILE ( word  "  Dividends on Corp Stocks ---- " g-docs " %" )

  LOG-TO-FILE ( "  OTHER GLOBALS" )
  LOG-TO-FILE ( word  "  g-crb-assets-per-prsn ------- " g-crb-assets-per-prsn )
  LOG-TO-FILE ( word  "  g-no-of-banks-max ----------- " g-no-of-banks-max )
  LOG-TO-FILE ( word  "  g-no-of-prsns-per-bank ------ " g-no-of-prsns-per-bank )

  ;; TODO: Remove this when slider is replaced.
  set g-no-of-corps-per-bank      1
  LOG-TO-FILE ( word  "  g-no-of-corps-per-bank ------ " g-no-of-corps-per-bank )
  LOG-TO-FILE ( word  "  g-net-worth-tax-rate -------- " g-net-worth-tax-rate " %" )
  LOG-TO-FILE ( word  "  g-reserve-requirement-ratio - " g-reserve-requirement-ratio
" %" )

  set g-no-of-banks ( count banks )
  set g-no-of-prsns-max ( g-no-of-banks-max * g-no-of-prsns-per-bank )
  set g-no-of-prsns ( count prsns )
  set g-no-of-corps ( g-no-of-banks-max * g-no-of-corps-per-bank )
  set g-p-daily-cost-of-living round( g-crb-assets-per-prsn / 30 ) ;; 30 days per
month
  set g-p-daily-L0-allocation round( g-p-daily-cost-of-living / 4 )
  set g-p-daily-L1-allocation ( g-p-daily-cost-of-living - g-p-daily-L0-allocation )
  set g-p-standard-loan ( g-p-daily-cost-of-living * 64 ) ;; 60+4; Used to set up
loans.
```

```
  set g-p-standard-loan-payment ( g-p-standard-loan / 8 ) ;; Used to pay principal
on loans.

  ;; TODO: The minimum vault cash must increase when corps are activated.
  ;;   Used to manage reserves
  set g-minimum-vault-cash ( g-p-daily-L0-allocation * g-no-of-prsns-per-bank )

  LOG-TO-FILE ( word  "  g-no-of-banks-max ----------- " g-no-of-banks-max )
  LOG-TO-FILE ( word  "  g-no-of-banks --------------- " g-no-of-banks )
  LOG-TO-FILE ( word  "  g-no-of-prsns-max ----------- " g-no-of-prsns-max )
  LOG-TO-FILE ( word  "  g-no-of-prsns --------------- " g-no-of-prsns )
  LOG-TO-FILE ( word  "  g-no-of-corps-max ----------- " g-no-of-corps-max )
  LOG-TO-FILE ( word  "  g-no-of-corps --------------- " g-no-of-corps )
  LOG-TO-FILE ( word  "  g-p-daily-cost-of-living ---- " g-p-daily-cost-of-living )
  LOG-TO-FILE ( word  "  g-p-daily-L0-allocation ----- " g-p-daily-L0-allocation )
  LOG-TO-FILE ( word  "  g-p-daily-L1-allocation ----- " g-p-daily-L1-allocation )
  LOG-TO-FILE ( word  "  g-p-standard-loan ----------- " g-p-standard-loan )
  LOG-TO-FILE ( word  "  g-p-standard-loan-payment --- " g-p-standard-loan-payment )
  LOG-TO-FILE ( word  "  g-minimum-vault-cash -------- " g-minimum-vault-cash )
  LOG-TO-FILE ( word  "  g-bankruptcy-factor --------- " g-bankruptcy-factor )

  LOG-TO-FILE ( word  "  gb-plot-data ---------------- " gb-plot-data )
  LOG-TO-FILE ( word  "  gb-bank-insurance ----------- " gb-bank-insurance )
  LOG-TO-FILE ( word  "  gb-btpfs-bankruptcies ------- " gb-btpfs-bankruptcies )
  LOG-TO-FILE ( word  "  gb-btpfs-daily-purchases ---- " gb-btpfs-daily-purchases )
  LOG-TO-FILE ( word  "  gb-btpfs-monthly-taxes ------ " gb-btpfs-monthly-taxes )

  ;; END OF SETUP FOR CONSERVEMONEYLAB


  ;; There are 2 scenarios possible
  set ge-scenario-with-prsns  0  ;; Prsns are active
  set ge-scenario-with-corps  1  ;; Corps are active

  ;; Use the input from the chooser gs-scenario to invoke the selected scenario.
  f-set-scenario-number

  ;; For debugging the setup procedure, log the values of the globals.
  LOG-TO-FILE ( word "  Scenario number ------------- " g-scenario-number )
  LOG-TO-FILE ( word "  Scenario name --------------- " gs-scenario )
  LOG-TO-FILE ( word "  Random seed ----------------- " g-use-this-seed )

  ;; For debugging the debug feature!!!
  LOG-TO-FILE ( word "SETUP: Debug Is --------------- " gb-debug-on )
  LOG-TO-FILE ( word "SETUP: Debug Status Is -------- " gs-debug-status )
  LOG-TO-FILE ( word "SETUP: Step Chooser Is -------- " gs-debug-step-chooser )
  LOG-TO-FILE ( word "SETUP: Flow Control Is -------- " gb-debug-flow-on )

  ask patches
  [
    set pcolor brown
  ]

  set g-agents-nw-xaxis-min 0
  set g-agents-nw-xaxis-max 1000
  set g-prsns-nw-xaxis-min  0
  set g-prsns-nw-xaxis-max  1000
  set g-banks-nw-xaxis-min  0
  set g-banks-nw-xaxis-max  1000
  set g-banks-P0-xaxis-min  0
  set g-banks-P0-xaxis-max  1000
  set g-banks-P0-all-assets-min  0    ;; Minimum value on P0-all-assets.
  set g-banks-P0-all-assets-mean 500  ;; Mean value on P0-all-assets.
  set g-banks-P0-all-assets-max  1000 ;; Max value on P0-all-assets.

  set g-counts-loans        0
  set g-counts-p-deaths     0
  set g-counts-p-births     0
  set g-counts-b-deaths     0
  set g-counts-b-births     0

  reset-ticks      ;; restarts tick counter and runs setup commands within plots

  ;; Set the switches to default setup values.
  set gb-plot-data            true ;; Enables all plotting calls.
  set gb-bank-insurance       true ;; Default insurance is on.

  if( g-scenario-number = ge-scenario-with-prsns )
  [
    set gb-plot-data            true ;; Enables all plotting calls.
  ]
  if( g-scenario-number = ge-scenario-with-corps )
  [
    set gb-plot-data            true ;; Enables all plotting calls.
  ]

  ;; Initalization of CmLab Turtles
  set-default-shape GCRAs  "triangle"   ;; pulled from shapes library
  set-default-shape CRBs   "triangle"   ;; pulled from shapes library
  set-default-shape banks  "target"     ;; pulled from shapes library
  set-default-shape prsns  "truck"      ;; pulled from shapes library
  set-default-shape corps  "house"      ;; pulled from shapes library
  f-initialize-basic-scenario

  ;; Do the bank visits to arrange deposits.
  f-everybody-visits-their-bank
  ;; Then update the net worth statements and global aggregates.
  ;; This call requires that 'reset-ticks' be called first.
  f-update-aggregates  ;; Totals and averages.

  ;; TODO: suppress or remove after debug.
  f-dump-all-agent-data

  ;; Clears unwanted zeros in plots.
  clear-all-plots
  setup-plots

  ;; Debug controls
  set gb-debug-flow-on 0 ;; Boolean, in association with chooser, turns debug LOG-
TO-FILE on/off
  set g-halt-at-tick -1  ;; input variable to set a tick for stopping

  ;; ASSERT ( frb-EMgr-is-valid ) ( "EMgr validity check: D-Setup" ) -1
  LOG-TO-FILE "  Do-Setup: procedure completed"

  ;; end of to setup
end

;;-----------------------------------------------------------------------------|
;; Set the scenario number using the input from the chooser.
to f-set-scenario-number
  ;; This routine is to be executed by the observer.

  set g-scenario-number ge-scenario-with-prsns  ;; default
  ;; if( gs-scenario = "Corps Not Implemented Yet" )
  ;;   [ set g-scenario-number ge-scenario-with-corps ]
  set gs-scenario "Prsns Only"

  ;; End f-set-scenario-number
end

;;-----------------------------------------------------------------------------|
;; Initialize a GCRA, CRB, banks, corps and prsns.
to f-initialize-basic-scenario
  ;; This routine is to be executed by the observer.

  ;; NOTE: the order of initialization is critical since there are links
  ;;   established between them, once appropriate linkable agents are created.

  ;; Initialize a GCRA.  (Government Consolidated Revenue Account)
  create-gcras 1
  [
    f-initialize-gcra
    setxy 0 0
  ]
  ;; Note: bank-who not set yet.

  ;; Initialize a CRB.   (Central Reserve Bank)
  create-crbs 1
  [
    f-initialize-crb
    ;; Move to a random point.
    setxy 0 1
  ]
  ;; Note: bank-who not set yet.

  ;; Initialize the banks.
  create-banks g-no-of-banks-max
  [
    set g-counts-b-births ( g-counts-b-births + 1 )
    f-initialize-new-bank
    ;; Move to a random point.
    setxy random-xcor random-ycor
  ]
  set g-no-of-banks ( count banks )
  ;; Move P0-assets to VC, ER and RR deposits, as appropriate.
  f-the-crb-reconciles-with-banks-daily

  ;; Assign a bank to the GCRA
  ask gcras [ f-bsvcs-gcra-find-bank ]
  ;; Assign a bank to the CRB
  ask crbs [ f-bsvcs-crb-find-bank ]

  ;; Initialize the prsns.
  ;; Must do banks and corps first, then link prsns to both.
  create-prsns g-no-of-prsns-max
  [
    set g-counts-p-births ( g-counts-p-births + 1 )
    f-initialize-new-prsn
    set heading 90
    ;; Move to a random point.
    setxy random-xcor random-ycor
  ]
  set g-no-of-prsns ( count prsns )

  ;; Initialize the corps.
```

```
  ;; Must do banks first, then link corps to banks.
  ;; TODO: Initialization of corps suppressed.
  ;; create-corps g-no-of-corps
  ;; [
  ;;   set g-counts-c-births ( g-counts-c-births + 1 )
  ;;   f-initialize-new-corp
  ;;   ;; Move to a random point.
  ;;   setxy random-xcor random-ycor
  ;; ]

  ;; The initial endowment of cash must be distributed.
  ask crbs
  [
    f-cbsvcs-distribute-assets-to-prsns
    ;; TODO: When corps implemented, include here.
  ]

  ;; End f-initialize-basic-scenario
end

;;-----------------------------------------------------------------------------|
;; Initialize a single GCRA.
to f-initialize-gcra
  ;; This routine is to be executed by a GCRA.
  ;; I.e. government consolidated revenue account.
  set heading 0  ;; direction of motion
  set color black

  ;; USER-DETERMINED ATTRIBUTES
  ;; Associated with GCRA dynamics.
  set default-colour      black              ;; distinctive colour for GCRA
  set bank-who           -1                  ;; bank that holds the loan
  set L1-assets           0                  ;; standard checking account
  set L1-loan-debts       0                  ;; debts associated with loan
  set S1-L1ip-debts       0                  ;; payable on loans

  ;; TODO: If these are not used, remove them.
  ;; xx set L2-assets           0                  ;; standard savings account

  ;; ss set L3-debts            0                  ;; bonds
  ;; ss set S1-L3ip-debts       0                  ;; payable on bonds

  LOG-TO-FILE ( word "  Initialize GCRA " who )
  LOG-TO-FILE ( word "  L1-assets ------------------- " L1-assets )
  LOG-TO-FILE ( word "  L1-loan-debts --------------- " L1-loan-debts )
  LOG-TO-FILE ( word "  S1-L1ip-debts --------------- " S1-L1ip-debts )
  ;; xx LOG-TO-FILE ( word "  L2-assets ------------------- " L2-assets )
  ;; ss LOG-TO-FILE ( word "  L3-debts -------------------- " L3-debts )
  ;; ss LOG-TO-FILE ( word "  S1-L3ip-debts -------- " S1-L3ip-debts )

  set ttl-P0-assets     0 ;; aggregate of all physical assets
  set ttl-publ-assets   0 ;; aggregate of all public assets
  set ttl-publ-debts    0 ;; aggregate of all public debts
  set ttl-priv-assets   0 ;; aggregate of all private assets
  set ttl-priv-debts    0 ;; aggregate of all private debts
  set net-worth-publ    0 ;; total public assets minus debts
  set net-worth-priv    0 ;; total private assets minus debts

  ;; Money supply aggregates
  set msi-assets        0 ;; Physical money supply
  set msi-debts         0 ;; Physical money supply
  set msii-assets       0 ;; Logical money supply
  set msii-debts        0 ;; Logical money supply
  set msiii-assets      0 ;; Shadow money supply
  set msiii-debts       0 ;; Shadow money supply

  ;; Suppressed.  Done after all banks initialized.
  ;; f-bsvcs-gcra-find-bank ;; sets bank-who to a valid number

  ;; end f-initialize-gcra
end

;;-----------------------------------------------------------------------------|
;; Initialize a single CRB.
to f-initialize-crb
  ;; This routine is to be executed by a CRB.
  ;; I.e. central reserve bank.
  set heading 0  ;; direction of motion
  set color yellow

  ;; USER-DETERMINED ATTRIBUTES
  ;; Associated with CRB dynamics.
  set default-colour      yellow  ;; distinctive colour for CRB
  ;; TODO: Change when corps activated.

  ;; The functional values of the assets are set in
  ;;   the routine f-cbsvcs-distribute-assets-to-prsns
  set P0-assets           0
  set P0-debts            0 ;; initial physcial debts on start
  set L0-assets           0 ;; initial logical assets on start
  set L0-debts            0 ;; initial logical debts on start

  set P0-rr-assets        0 ;; required reserves
  set P0-er-assets        0 ;; excess reserves

  set bank-who           -1 ;; chartered bank for C1 account
  set S1-rrip-debts       0 ;; interest payable on required reserves
  set S1-erip-debts       0 ;; interest payable on excess reserves
  set C1-assets           0 ;; corporate bank assets
  ;; xx set c2-assets           0 ;; corporate bank assets

  LOG-TO-FILE ( word "  Initialize CRB " who )
  LOG-TO-FILE ( word "  CRB MS-I P0 Assets ----------- " P0-assets )
  LOG-TO-FILE ( word "  CRB MS-I F0 Assets ----------- " L0-assets )
  LOG-TO-FILE ( word "  CRB MS-I P0 debts ------------ " P0-debts )
  LOG-TO-FILE ( word "  CRB MS-I F0 debts ------------ " L0-debts )
  LOG-TO-FILE ( word "  CRB Required reserves -------- " P0-rr-assets )
  LOG-TO-FILE ( word "  S1-rrip-debts ---------------- " S1-rrip-debts )
  LOG-TO-FILE ( word "  CRB Excess reserves ---------- " P0-er-assets )
  LOG-TO-FILE ( word "  S1-erip-debts ---------------- " S1-erip-debts )

  set ttl-P0-assets     0 ;; aggregate of all physical assets
  set ttl-publ-assets   0 ;; aggregate of all public assets
  set ttl-publ-debts    0 ;; aggregate of all public debts
  set ttl-priv-assets   0 ;; aggregate of all private assets
  set ttl-priv-debts    0 ;; aggregate of all private debts
  set net-worth-publ    0 ;; total public assets minus debts
  set net-worth-priv    0 ;; total private assets minus debts

  ;; Money supply aggregates
  set msi-assets        0 ;; Physical money supply
  set msi-debts         0 ;; Physical money supply
  set msii-assets       0 ;; Logical money supply
  set msii-debts        0 ;; Logical money supply
```

```
  set msiii-assets      0 ;; Shadow money supply
  set msiii-debts       0 ;; Shadow money supply

  ;; Suppressed.  Done after all banks initialized.
  ;; f-bsvcs-crb-find-bank ;; sets bank-who to a valid number

  ;; end f-initialize-crb
end

;;-----------------------------------------------------------------------------|
;; Initialize a single bank.
to f-initialize-new-bank
  ;; This routine is to be executed by a bank.

  ;; BUILT-IN ATTRIBUTES
  set heading 0  ;; direction of motion
  set color red

  LOG-TO-FILE ( word "  Initialize bank " who )
  ;; USER-DETERMINED ATTRIBUTES
  ;; Associated with bank dynamics.
  set default-colour       red  ;; distinctive colour for banks
  set b-bank-can-make-loans  1  ;; boolean - 0 or 1
  set b-bank-is-bankrupt     0  ;; boolean - 0 or 1

  set L1-assets              0
  set L1-loan-assets         0
  set L1-debts               0
  set S1-L1ir-assets         0

  set L2-assets              0
  set L2-debts               0
  set S1-L2ip-debts          0

  ;; xx set L3-assets              0

  ;; There is only one CRB, but the breed must be treated as a set.
  set crb-who ( [who] of ( one-of crbs ) )

  set P0-vc-assets           0
  set P0-er-assets           0
  set P0-er-debts            0
  set P0-rr-assets           0
  set P0-rr-debts            0
  set P0-all-assets          0

  ;; Associated with corporate bank dynamics.
  set no-of-prsn-clients     0
  set no-of-corp-clients     0
  set no-of-gcra-clients     0
  set no-of-crb-clients      0
  set S1-rrir-assets         0  ;; interest on required reserves
  set S1-erir-assets         0  ;; interest on excess reserves
  set C1-assets              0  ;; corporate bank equivalent of L1-assets
  ;; xx set c2-assets           0  ;; corporate bank equivalent of L2-assets

  set ttl-P0-assets     0 ;; aggregate of all physical assets
  set ttl-publ-assets   0 ;; aggregate of all public assets
  set ttl-publ-debts    0 ;; aggregate of all public debts
  set ttl-priv-assets   0 ;; aggregate of all private assets
  set ttl-priv-debts    0 ;; aggregate of all private debts
  set net-worth-publ    0 ;; total public assets minus debts
  set net-worth-priv    0 ;; total private assets minus debts

  ;; Money supply aggregates
  set msi-assets        0 ;; Physical money supply
  set msi-debts         0 ;; Physical money supply
  set msii-assets       0 ;; Logical money supply
  set msii-debts        0 ;; Logical money supply
  set msiii-assets      0 ;; Shadow money supply
  set msiii-debts       0 ;; Shadow money supply

  ;; end f-initialize-new-bank
end

;;-----------------------------------------------------------------------------|
;; Initialize a single prsn.
to f-initialize-new-prsn
  ;; This routine is to be executed by a prsn.

  ;; BUILT-IN ATTRIBUTES
  set heading 0  ;; direction of motion
  set color green

  LOG-TO-FILE ( word "  Initialize prsn " who )
  ;; USER-DETERMINED ATTRIBUTES
  ;; Associated with prsn dynamics.
  set default-colour    green ;; distinctive colour for prsns
  set b-prsn-is-bankrupt   0 ;; boolean - 0 or 1

  set P0-assets            0
  set L0-assets            0

  set bank-who             -1  ;; Does banking with this bank.
  set L1-assets             0
  set L1-loan-debts         0
  set S1-L1ip-debts         0 ;; payable on bank loans
  set payables-30day       []  ;; A list of 30-day payables
  set S1-30day-total-debts  0  ;; sum of 30 day payables
  set S1-30day-total-assets 0  ;; sum of 30 day receivables

  set L2-assets             0

  ;; ss set L3-corpwho       -1    ;; Holds bond from this corp.
  ;; ss set L3-assets         0

  ;; ss set L4-corpwho       -1    ;; Holds stock from this corp.
  ;; ss set L4-assets         0

  set ttl-P0-assets     0 ;; aggregate of all physical assets
  set ttl-publ-assets   0 ;; aggregate of all public assets
  set ttl-publ-debts    0 ;; aggregate of all public debts
  set ttl-priv-assets   0 ;; aggregate of all private assets
  set ttl-priv-debts    0 ;; aggregate of all private debts
  set net-worth-publ    0 ;; total public assets minus debts
  set net-worth-priv    0 ;; total private assets minus debts

  ;; Money supply aggregates
  set msi-assets        0 ;; Physical money supply
  set msi-debts         0 ;; Physical money supply
  set msii-assets       0 ;; Logical money supply
  set msii-debts        0 ;; Logical money supply
  set msiii-assets      0 ;; Shadow money supply
  set msiii-debts       0 ;; Shadow money supply
```

```
  f-bsvcs-prsn-find-bank  ;; Assign a bank to this prsn.
  ;; end f-initialize-new-prsn
end

;;-----------------------------------------------------------------------------|
;; Initialize a single corp.
to f-initialize-new-corp
  ;; This routine is to be executed by a corp.

  ;; BUILT-IN ATTRIBUTES
  set heading 0  ;; direction of motion
  set color black

  LOG-TO-FILE ( word "  Initialize corp " who )
  ;; USER-DETERMINED ATTRIBUTES
  ;; Associated with corp dynamics.
  set default-colour   black ;; distinctive colour for corps
  set b-corp-is-bankrupt   0 ;; boolean - 0 or 1

  set P0-assets            0
  set L0-assets            0

  set bank-who             -1 ;; Does banking with this bank.
  set L1-assets            0
  set L1-loan-debts        0
  set S1-L1ip-debts        0 ;; payable on bank loans
  set payables-30day      []
  set S1-30day-total-debts 0
  set S1-30day-total-assets 0

  set L2-assets            0
  set S1-L2ir-assets       0 ;; receivable on savings

  ;; ss set no-of-bond-clients  0 ;; prsns holding bonds
  ;; ss set L3-assets           0
  ;; ss set L3-debts            0

  ;; ss set no-of-stock-clients 0 ;; prsns holding stocks
  ;; ss set L4-assets           0
  ;; ss set L4-debts            0

  set ttl-P0-assets     0 ;; aggregate of all physical assets
  set ttl-publ-assets   0 ;; aggregate of all public assets
  set ttl-publ-debts    0 ;; aggregate of all public debts
  set ttl-priv-assets   0 ;; aggregate of all private assets
  set ttl-priv-debts    0 ;; aggregate of all private debts
  set net-worth-publ    0 ;; total public assets minus debts
  set net-worth-priv    0 ;; total private assets minus debts

  ;; Money supply aggregates
  set msi-assets        0 ;; Physical money supply
  set msi-debts         0 ;; Physical money supply
  set msii-assets       0 ;; Logical money supply
  set msii-debts        0 ;; Logical money supply
  set msiii-assets      0 ;; Shadow money supply
  set msiii-debts       0 ;; Shadow money supply

  f-bsvcs-corp-find-bank ;; Assign a bank to this corp.
  ;; end f-initialize-new-corp
end
```

```
;;-----------------------------------------------------------------------------|
;; SECTION D – GO OR MAIN-LOOP PROCEDURE( S )
;;-----------------------------------------------------------------------------|

;;-----------------------------------------------------------------------------|
;; The go button
to go
  ;; This routine is to be executed by the observer.

  ;; Stop codes:
  ;; All stop decisions must be here in the 'go' procedure, as it causes an
  ;;   exit from the current procedure only.

  if( g-halt-at-tick = ticks  )
  [
    set g-halt-at-tick -1
    stop
  ]

  ;; Ensure that the gb-btpfs-bankruptcies flag is always on.
  set gb-btpfs-bankruptcies true

  ;; MANUAL CHANGE FOR DEBUG
  ;; If needed, each check for validity can be enabled between steps.
  ;; They have been suppressed (turned into comments) for the sake
  ;;   of speed of execution, but can be re-enabled if a bug has
  ;;   somehow been re-introduced.
  ;; A single call to the validity check has been left active inside of the
  ;;   Do-Post-Tick step.  If it flags a problem, re-activate these to
  ;;   narrow down where the problem starts.

  ;; Major steps or functions, done once per tick, in order of execution.
  do-pre-tick
  ;; if( frb-agents-are-all-valid = false )
  ;;   [ LOG-TO-FILE ( word "Agents failed validity test: Do-pre-tick." ) ]

  do-move
  ;; if( frb-agents-are-all-valid = false )
  ;;   [ LOG-TO-FILE ( word "Agents failed validity test: Do-move." ) ]

  do-buy-sell
  ;; if( frb-agents-are-all-valid = false )
  ;;   [ LOG-TO-FILE ( word "Agents failed validity test: Do-buy-sell." ) ]

  do-accrue-interest
  ;; if( frb-agents-are-all-valid = false )
  ;;   [ LOG-TO-FILE ( word "Agents failed validity test: Do-accrue-interest." ) ]

  do-monthly
  ;; if( frb-agents-are-all-valid = false )
  ;;   [ LOG-TO-FILE ( word "Agents failed validity test: Do-monthly." ) ]

  do-banking
  ;; if( frb-agents-are-all-valid = false )
  ;;   [ LOG-TO-FILE ( word "Agents failed validity test: Do-banking." ) ]

  do-post-tick
  ;; if( frb-agents-are-all-valid = false )
  ;;   [ LOG-TO-FILE ( word "Agents failed validity test: Do-post-tick." ) ]

  ;; end of go
end
```

```
;;-----------------------------------------------------------------------------|
;; D1 - do-pre-tick procedure( s )
;;-----------------------------------------------------------------------------|
to do-pre-tick
  ;; This routine is to be executed by the observer.

  if( gb-debug-on = 1 )
  [
    ifelse( ( gs-debug-step-chooser = "all" ) or ( gs-debug-step-chooser = "pre-
tick" ) )
    [ set gb-debug-flow-on 1 LOG-TO-FILE "" LOG-TO-FILE word "Do-pre-tick: Debug
on.; tick was " ticks ]
    [ set gb-debug-flow-on 0 ]
  ]

  ;; Enter all commands that need to be done before a tick begins.
  ;; f-update-aggregates

  ;; Override the scenario chooser.
  set gs-scenario "Prsns Only"
  f-set-scenario-number

  ;; Advance the tick counter by 1 tick.
  ifelse( gb-plot-data = true )
  [
    ;; Advance the ticks by one and update the plots.
    tick
    ;; 'tick' is exactly the same as 'update-plots' except that the tick counter
    ;;   is incremented before the plot commands are executed.

  ]
  ;; else
  [
    ;; Advance ticks by one but do not update the plots.
    tick-advance 1
  ]
  ;; End else

  ;; Once the data is plotted, the per-tick counts can be cleared.
  ;; TODO: Clear such data collection per-tick aggregates here.

  ;; Reset the scenario number, in case the chooser has been changed.
  f-set-scenario-number

  LOG-TO-FILE ( word "  Halt at tick - " g-halt-at-tick  )
  LOG-TO-FILE ( word "  Current tick - " ticks )

  LOG-TO-FILE "  Do-pre-tick: Routine completed."
;; end of Do-pre-tick
end

;;-----------------------------------------------------------------------------|
;; D2 – do-move procedure(s)
;;-----------------------------------------------------------------------------|
to do-move
  ;; This routine is to be executed by the observer.

  if( gb-debug-on = 1 )
  [
    ifelse( ( gs-debug-step-chooser = "all" ) or ( gs-debug-step-chooser = "move" )
)
```

```
    [ set gb-debug-flow-on 1 LOG-TO-FILE "" LOG-TO-FILE word "Do-move: Debug on;
tick = " ticks ]
    [ set gb-debug-flow-on 0 ]
  ]

  ;; Implement 'arrow' behaviour from PSoup application.  I.e. a strong
  ;;   probability of movement directly forward, and small probability of a
  ;;   slight turn.  This represents the most effective search pattern for
  ;;   an arena that is wrapped on all sides.  Of course, it doesn't matter
  ;;   since they don't actually feed.

  let heading-list [ -1 0 0 0 0 0 0 0 0 0 1 ]

  ;; The prsns move.  'Arrow' search pattern.
  ask prsns
  [
    let delta-heading ( item ( random length heading-list ) heading-list )
    set heading ( heading + delta-heading )
    if( heading > 115 ) [ set heading 115 ]
    if( heading <  65 ) [ set heading  65 ]
    forward 1
  ]  ;; End ask prsns

  ;; f-update-aggregates

  LOG-TO-FILE "  Do-move: procedure completed"
;; end of Do-move
end

;;---------------------------------------------------------------------------|
;; D3 - do-buy-sell procedure(s)
;;---------------------------------------------------------------------------|
to do-buy-sell
  ;; This routine is to be executed by the observer.

  if( gb-debug-on = 1 )
  [
    ifelse( ( gs-debug-step-chooser = "all" ) or ( gs-debug-step-chooser = "buy-
sell" ) )
    [ set gb-debug-flow-on 1 LOG-TO-FILE "" LOG-TO-FILE word "Do-buy-sell: Debug on;
tick = " ticks ]
    [ set gb-debug-flow-on 0 ]
  ]

  ;; Each tick the prsns are paired as (buyer, seller) for cash transactions.
  f-prsns-buy-sell-using-cash

  ;; Each tick the banks buy using checks on their C1 accounts.
  f-btpfs-banks-buy-using-checks

  ;; Each tick the prsns are re-paired as (buyer, seller) on 30-day terms.
  f-prsns-buy-sell-on-terms

  ;; Each tick each prsn then pays those bills that are 30 days old or more.
  f-process-30-day-payables
  ;; TODO: When corps implemented, this needs to be added for them too.

  f-update-aggregates

  LOG-TO-FILE "  Do-buy-sell: procedure completed"

;; end of Do-buy-sell
```

```
end

;;---------------------------------------------------------------------------|
;; Prsns buy and sell, using cash.
to f-prsns-buy-sell-using-cash
;; This routine is to be executed by the observer.

  ;; Prsns buy and sell using cash.
  ;; Each tick the prsns are paired as (buyer, seller) for cash transactions.
  LOG-TO-FILE ( word "" )
  LOG-TO-FILE ( word "Do-buy-sell: cash" )

  ;; Make a list.
  let mylist []
  ask prsns
  [
    set mylist lput self mylist
  ]

  let no-of-prsns-left ( length mylist )
  ;; LOG-TO-FILE ( word "  Do-buy-sell: no-of-prsns-left " no-of-prsns-left )

  while [ no-of-prsns-left > 1 ]
  [
    ;; Isolate the first two prsns.
    let buyer ( item 0 mylist )
    set mylist ( but-first mylist )
    let seller ( item 0 mylist )
    set mylist ( but-first mylist )
    set no-of-prsns-left ( length mylist )

    let buyer-who ( [who] of buyer )
    let seller-who ( [who] of seller )

    ask buyer
    [
      ;; Buyer transfers cash (P0+L0) to seller.
      ;; This is a similar technique to Yakovenko's capital exchange models.
      ;; Dragulescu and Yakovenko, 2000.
      let amount-to-spend ( 1 + ( random ( g-p-daily-L0-allocation - 1 ) ) )
      LOG-TO-FILE ( word "Buyer: " buyer-who "; Seller: " seller-who )
      LOG-TO-FILE ( word "  L0-assets of buyer ------------------ " L0-assets )
      LOG-TO-FILE ( word "  L0-assets of seller ----------------- " ( [L0-assets] of
seller ) )
      LOG-TO-FILE ( word "  L0 cost of purchase ----------------- " amount-to-spend
)

      f-bsvcs-prsn1-pays-prsn2-by-cash seller-who amount-to-spend

      LOG-TO-FILE ( word "  L0-assets of buyer ------------------ " L0-assets )
      LOG-TO-FILE ( word "  L0-assets of seller ----------------- " ( [L0-assets] of
seller ) )
    ]
  ]

;; end of f-prsns-buy-sell-using-cash
end

;;---------------------------------------------------------------------------|
;; Prsns buy and sell, on 30-day terms.
to f-prsns-buy-sell-on-terms
;; This routine is to be executed by the observer.
```

```
  ;; THEORY: Prsns buy and sell, paying by check after 30 days.
  ;; Each tick the prsns are randomly paired as (buyer, seller) on 30-day terms.
  LOG-TO-FILE ( word "  " )
  LOG-TO-FILE ( word "Do-buy-sell: 30-day terms" )

  ;; Make a list of prsns other than me.
  let mylist []
  ask other prsns ;; excludes me
  [
    ;; Add themself to my list of prsns.
    set mylist lput self mylist
  ]

  let no-of-prsns-left ( length mylist )
  ;; LOG-TO-FILE ( word "  Do-buy-sell: no-of-prsns-left " no-of-prsns-left )
  while [ no-of-prsns-left > 1 ]
  [
    ;; Isolate the first two prsns.
    let buyer  ( item 0 mylist )
    set mylist ( but-first mylist )
    let seller ( item 0 mylist )
    set mylist ( but-first mylist )
    set no-of-prsns-left ( length mylist )

    let buyer-who ( [who] of buyer )
    let seller-who ( [who] of seller )

    ask buyer
    [
      ;; THEORY:  This is totally happening in the shadow money supply, and
      ;;   no bank of any kind is involved.  So, there is no "banking services"
      ;;   routine (i.e. one with -bsvcs- in the name) to handle this.  It is
      ;;   coded in detail here.

      ;; Buyer puts purchase on a 30-day tab.
      ;; This puts the purchase into the MS-III money supply.
      let amount-to-spend ( 1 + ( random ( g-p-daily-L1-allocation - 1 ) ) )

      ;; Buyer spends expecting to pay by check in 30 days.
      ;; Buyer does not/cannot check for future solvency.
      ;; This must be paid 30 ticks from now.
      LOG-TO-FILE ( word "Buyer: " buyer-who "; Seller: " seller-who )
      LOG-TO-FILE ( word "  30day payables of buyer ------------- " S1-30day-total-
debts )
      LOG-TO-FILE ( word "  30day receivables of seller --------- " ( [S1-30day-
total-assets] of seller ) )
      set S1-30day-total-debts ( S1-30day-total-debts + amount-to-spend )
      ask seller [ set S1-30day-total-assets ( S1-30day-total-assets + amount-to-
spend ) ]
      let payable ( list ( [who] of seller ) ( ticks + 30 ) amount-to-spend )
      set payables-30day lput payable payables-30day
      LOG-TO-FILE ( word "  This purchase [sllr, tick due, amt] - " payable )
      LOG-TO-FILE ( word "  30day payables of buyer ------------- " S1-30day-total-
debts )
      LOG-TO-FILE ( word "  30day receivables of seller --------- " ( [S1-30day-
total-assets] of seller ) )
    ]
  ]

;; end of f-prsns-buy-sell-on-terms
end
```

```
;;------------------------------------------------------------------------------|
;; Corps buy and sell, using cash and on 30-day terms.
to f-corps-buy-sell
;; This routine is to be executed by the observer.

;; TODO: Not implemented yet.

;; end of f-corps-buy-sell
end

;;------------------------------------------------------------------------------|
;; Process 30-day payables.
to f-process-30-day-payables
;; This routine is to be executed by the observer.

  ;; THEORY:  This is a connection between the shadow and the logical
  ;;   money supplies.  The payables and receivables that were not in bank
  ;;   records are now paid by checks and a -bsvcs- routine, and they become
  ;;   visible to the banks and their back room accountants.

  ;;  All prsns may have 30-day payables.
  ask prsns
  [
    ;; If there are no payables, nothing need be done my this prsn.
    ;; TODO: For performance, add boolean to determine if payables are due
    ;;   this tick.
    if( S1-30day-total-debts > 0 )
    [
      ;; I used lput to put the payables into a list.  So I should be able to
      ;;   pull them off of the front until those that are payable this tick
      ;;   have been looked after.

      let this-payable ( item 0 payables-30day )
      let seller-who item 0 this-payable
      let tick-when-due item 1 this-payable
      let this-amount item 2 this-payable

      if( tick-when-due <= ticks )
      [
        LOG-TO-FILE ( word "  " )
        LOG-TO-FILE ( word "PRSN " who " processing 30-day payables" )
      ]

      while [ tick-when-due <= ticks ]
      [
        let seller ( prsn seller-who )
        LOG-TO-FILE ( word "  This payable ---------------- " this-payable )
        LOG-TO-FILE ( word "  Seller ---------------------- " seller-who )
        LOG-TO-FILE ( word "  Tick-when-due --------------- " tick-when-due "; now -
" ticks )
        LOG-TO-FILE ( word "  Seller's assets were -------- " ( [L1-assets] of
seller ) )
        LOG-TO-FILE ( word "  Buyer's assets were --------- " L1-assets )
        LOG-TO-FILE ( word "  Amount due ------------------ " this-amount )
        f-bsvcs-prsn1-pays-prsn2-by-check seller-who this-amount
        LOG-TO-FILE ( word "  Seller's assets are --------- " ( [L1-assets] of
seller ) )
        LOG-TO-FILE ( word "  Buyer's assets are ---------- " L1-assets )

        ;; Update the aggregator of the buyer.
        set S1-30day-total-debts ( S1-30day-total-debts - this-amount )
        ;; Update the aggregator of the seller.
        ask seller [ set S1-30day-total-assets
          ( S1-30day-total-assets - this-amount ) ]

        ;; The first payable in list is done.  Drop from list.
        set payables-30day ( but-first payables-30day )
        ;; Check if there are any more.
        ifelse( 0 = length payables-30day )
        [
          set tick-when-due ( ticks + 1 )  ;; Create end condition.
        ]
        ;; Else
        [
          ;; Unpack the next payable.
          set this-payable ( item 0 payables-30day )
          set seller-who item 0 this-payable
          set tick-when-due item 1 this-payable
          set this-amount item 2 this-payable
        ]
      ]
    ]

  ]
;;  end of f-process-30-day-payables
end

;;-----------------------------------------------------------------------------|
;; D4 - do-accrue-interest procedure(s)
;;-----------------------------------------------------------------------------|
to do-accrue-interest
  ;; This routine is to be executed by the observer.

  if( gb-debug-on = 1 )
  [
    ifelse( ( gs-debug-step-chooser = "all" ) or ( gs-debug-step-chooser = "accrue-
interest" ) )
    [ set gb-debug-flow-on 1 LOG-TO-FILE "" LOG-TO-FILE word "Do-accrue-interest:
Debug on; tick = " ticks ]
    [ set gb-debug-flow-on 0 ]
  ]

  ;; TODO: Corps and GCRA do not presently take out L1 loans, or make savings
  ;;   deposits, so some of this code is anticipating that change.  When those
  ;;   things are added, walk through this again.

  ;; There are six kinds of interest that must be accrued, and paid monthly.
  ;;   - interest on L1 bank loans - client to bank
  ;;   - interest on L2 savings deposits - bank to client
  ;;   - interest on required reserves - CRB to bank
  ;;   - interest on excess reserves - CRB to bank
  ;;   - dividends on stocks - corps to shareholders (not implemented yet)
  ;;   - interest on bonds - GCRA and corps to bondholders (not implemented yet)

  f-accrue-interest-on-bank-loans-and-deposits
  f-accrue-interest-on-reserves
  ;; TODO: Implement when corps activated.
  ;; f-accrue-dividends-on-corporate-stocks

  f-update-aggregates

  LOG-TO-FILE "  Do-accrue-interest: procedure completed"
;; end of do-accrue-interest
end

;;-----------------------------------------------------------------------------|
;; In this routine all per-tick interest and dividends are accrued.
to f-accrue-interest-on-bank-loans-and-deposits
;; This routine is to be executed by the observer.

  ;; For each prsn (and corp, and gov't) figure out how much interest
  ;;   must be paid on the current extant amount on a loan.  This is calculated
  ;;   daily (per tick) and added up, and paid at the end of the month.

  ;; First, check the government's consolidated revenue account (GCRA).
  ;; TODO: enable this when GCRA loans are implemented.
  ;; ask gcras
  ;; [
  ;;   if( L1-loan-debts > 0 )
  ;;   [
  ;;     LOG-TO-FILE ( word "  " )
  ;;     LOG-TO-FILE ( word "GCRA Bank Loan " )
  ;;     LOG-TO-FILE ( word "  Size of L1 loan --------------- " L1-loan-debts )
  ;;     f-bsvcs-client-accrues-daily-interest-on-L1-loan
  ;;     LOG-TO-FILE ( word "  Total interest due ------------ " S1-L1ip-debts )
  ;;   ]
  ;; ]

  ;; Next, check the prsns loans (L1) and savings (L2) accounts.
  ;;
  ask prsns
  [
    ;; Loans appear as L1 debts.
    if( L1-loan-debts > 0 )
    [
      LOG-TO-FILE ( word "  " )
      LOG-TO-FILE ( word "PRSN " who " - Bank Loan" )
      LOG-TO-FILE ( word "  Size of L1 loan --------------- " L1-loan-debts )
      f-bsvcs-client-accrues-daily-interest-on-L1-loan
      LOG-TO-FILE ( word "  Total interest due ------------ " S1-L1ip-debts )
    ]

    ;; Savings appear as L2 assets.
    if( L2-assets > 0 )
    [
      LOG-TO-FILE ( word "  " )
      LOG-TO-FILE ( word "PRSN " who " - Savings Deposit" )
      LOG-TO-FILE ( word "  Size of L2 savings deposit ---- " L2-assets )
      f-bsvcs-client-accrues-daily-interest-on-L2-savings
      LOG-TO-FILE ( word "  Total interest due ------------ " S1-L2ir-assets )
    ]
  ]

  ;; TODO:  Interest for corps not yet implemented.  Do like prsns.
  ;; Savings acct for GCRA not yet implemented.

;; end of f-accrue-interest-on-bank-loans-and-deposits
end

;;-----------------------------------------------------------------------------|
;; In this routine all per-tick interest is accrued.
to f-accrue-interest-on-reserves
;; This routine is to be executed by the observer.

  ;; For each bank figure out how much interest is payable on their CRB
  ;;   deposits.  This is calculated daily (per tick) and added up,
  ;;   and paid at the end of the month.

  ask banks
  [
    ;; Do required reserves first.
    if( P0-rr-assets > 0 )
    [
      LOG-TO-FILE ( word "  " )
      LOG-TO-FILE ( word "BANK " who " - RR Deposit" )
      LOG-TO-FILE ( word "  Size of RR deposit ------------ " P0-rr-assets )
      f-cbsvcs-bank-accrues-daily-interest-on-RR-deposits      ;; Contact the bank.
      LOG-TO-FILE ( word "  Total interest due ------------ " S1-rrir-assets )
    ]

    ;; Now do excess reserves.
    if( P0-er-assets > 0 )
    [
      LOG-TO-FILE ( word "  " )
      LOG-TO-FILE ( word "BANK " who " - ER Deposit" )
      LOG-TO-FILE ( word "  Size of ER deposit ------------ " P0-er-assets )
      f-cbsvcs-bank-accrues-daily-interest-on-ER-deposits      ;; Contact the bank.
      LOG-TO-FILE ( word "  Total interest due ------------ " S1-erir-assets )
    ]
  ]
;; end of f-accrue-interest-on-reserves
end

;;---------------------------------------------------------------------------|
;; Accrue per-tick dividends on corporate stocks.
to f-accrue-dividends-on-corporate-stocks
;; This routine is to be executed by the observer.

  ;; TODO:  Add a body to this hook.

;; end of f-accrue-dividends-on-corporate-stocks
end

;;---------------------------------------------------------------------------|
;; D5 - do-monthly procedure(s)
;;---------------------------------------------------------------------------|
to do-monthly
  ;; This routine is to be executed by the observer.

  if( gb-debug-on = 1 )
  [
    ifelse( ( gs-debug-step-chooser = "all" ) or ( gs-debug-step-chooser = "monthly"
) )
    [ set gb-debug-flow-on 1 LOG-TO-FILE "" LOG-TO-FILE word "Do-monthly: Debug on;
tick = " ticks ]
    [ set gb-debug-flow-on 0 ]
  ]

  ;; There are four or five procedures that need to be done once a
  ;;   month (every 30 days)
  let check-value ( ticks mod 30 )
  if( check-value = 0 )
  [
    f-cbsvcs-gcra-reconciles-with-crb-monthly
    f-process-interest-payments-monthly
    f-process-payments-on-loans-monthly
    f-government-spends-and-taxes-monthly
    f-btpfs-government-special-monthly-transfer
  ]

  f-update-aggregates

  LOG-TO-FILE "  Do-monthly: procedure completed"
;;  end of do-monthly
end

;;---------------------------------------------------------------------------|
;; Process interest payments monthly.
to f-process-interest-payments-monthly
;; This routine is to be executed by the observer.

  ;; Monthly interest payments will be made by check
  ;;   from/to the L1 checking accts.

  ;; Prsns can make payments on L1 loans and collect payments on L2 savings.
  ask prsns
  [
    ;; Contact the bank.
    let mybank ( bank bank-who )

    ;; NOTE: a payment of interest on a loan does not affect the principal.
    ;;   It causes a change of net-worth of both participants.  The payables
    ;;   and receivables do not appear on the official books of either
    ;;   party until the month-end reconciliation happens.  The changes to the
    ;;   C1-assets and the L1-assets are the effective transfer of
    ;;   net-worth monthly.  Only due payments above $1 are processed.

    ;; Make interest payments on L1 loans.
    if( S1-L1ip-debts > 1 )
    [
      LOG-TO-FILE ( word "INTEREST PAYMENT ON LOAN:" )
      LOG-TO-FILE ( word "  Prsn " who " to bank " bank-who "." )
      LOG-TO-FILE ( word "  Prsn L1 loan ------------------ " L1-loan-debts )
      LOG-TO-FILE ( word "  Prsn L1 assets before payment - " L1-assets )
      LOG-TO-FILE ( word "  Bank C1 assets before payment - " ( [C1-assets] of
mybank ) )
      LOG-TO-FILE ( word "  Current amount payable -------- " ( S1-L1ip-debts ) )
      f-bsvcs-client-pays-monthly-interest-on-L1-loan
      ;; NOTE: Due to the rounding of the interest-paid, a residual
      ;;   of interest payable will remain each month.  I do this to
      ;;   keep net worth integral.
      LOG-TO-FILE ( word "  Prsn L1 assets after payment -- " L1-assets )
      LOG-TO-FILE ( word "  Bank C1 assets after payment -- " ( [C1-assets] of
mybank ) )
      LOG-TO-FILE ( word "  Residual payable -------------- " ( S1-L1ip-debts ) )
    ]

    ;; Collect interest payments on L2 savings deposits.
    if( S1-L2ir-assets > 1 )
    [
      let interest-due floor( S1-L2ir-assets )
      LOG-TO-FILE ( word "INTEREST PAYMENT ON SAVINGS ACCOUNT:" )
      LOG-TO-FILE ( word "  Bank " bank-who " to prsn " who )
      LOG-TO-FILE ( word "  Prsn L1 assets before payment - " L1-assets )
      LOG-TO-FILE ( word "  Prsn L2 assets ---------------- " L2-assets )
      LOG-TO-FILE ( word "  Bank C1 assets before payment - " ( [C1-assets] of
mybank ) )
      LOG-TO-FILE ( word "  Current amount receivable ----- " ( S1-L2ir-assets ) )
      f-bsvcs-client-paid-monthly-interest-on-L2-savings
      ;; NOTE: Due to rounding above, some residual interest-receivable
      ;;   will remain.
      LOG-TO-FILE ( word "  Prsn L1 assets after payment -- " L1-assets )
      LOG-TO-FILE ( word "  Bank C1 assets after payment -- " ( [C1-assets] of
mybank ) )
      LOG-TO-FILE ( word "  Residual receivable ----------- " ( S1-L2ir-assets ) )
    ]

    ;; Prsns can collect payments on stocks and bonds.
    ;; TODO:  Not yet implemented.

  ] ;; End ask prsns

  ;; Corps can make payments on L1 loans and collect payments on L2 savings.
  ;; TODO:  Not yet implemented.

  ;; The government can pay interest on bank loans.
  ask gcras
  [
    ;; Contact the bank.
    let mybank ( bank bank-who )
    ;; Make interest payments on L1 loans.
    if( S1-L1ip-debts > 1 )
    [
      LOG-TO-FILE ( word "INTEREST PAYMENT ON LOAN:" )
      LOG-TO-FILE ( word "  GCRA " who " to bank " bank-who "." )
      LOG-TO-FILE ( word "  GCRA L1 loan ------------------ " L1-loan-debts )
      LOG-TO-FILE ( word "  GCRA L1 assets pre-payment ---- " L1-assets )
      LOG-TO-FILE ( word "  Bank C1 assets pre-payment----- " ( [C1-assets] of
mybank ) )
      LOG-TO-FILE ( word "  Current payable --------------- " ( S1-L1ip-debts ) )
      f-bsvcs-client-pays-monthly-interest-on-L1-loan
      ;; NOTE: Due to the rounding of the interest-paid, a residual
      ;;   of interest payable will remain each month.  I do this to
      ;;   keep net worth integral.
      LOG-TO-FILE ( word "  GCRA L1 assets post-payment --- " L1-assets )
      LOG-TO-FILE ( word "  Bank C1 assets post-payment --- " ( [C1-assets] of
mybank ) )
      LOG-TO-FILE ( word "  Residual payable -------------- " ( S1-L1ip-debts ) )
    ]
  ]

  ;; The CRB can pay interest to banks on reserve deposits.
  ask banks
  [
    ;; Collect interest payments on required reserve deposits.
    if( S1-rrir-assets > 1 )
    [
      let the-crb ( crb crb-who )
      LOG-TO-FILE ( word "INTEREST PAYMENT ON RR:" )
      LOG-TO-FILE ( word "  CRB " crb-who " to bank " who "." )
      LOG-TO-FILE ( word "  Bank C1 assets ---------------- " C1-assets )
      LOG-TO-FILE ( word "  Bank L1 debts ----------------- " L1-debts )
      LOG-TO-FILE ( word "  CRB C1 assets ----------------- " ( [C1-assets] of the-
crb ) )
      LOG-TO-FILE ( word "  Current receivable ------------ " ( S1-rrir-assets ) )
      f-cbsvcs-bank-paid-monthly-interest-on-rr-deposits
      LOG-TO-FILE ( word "  CRB C1 assets ----------------- " ( [C1-assets] of the-
crb ) )
      LOG-TO-FILE ( word "  Bank C1 assets ---------------- " ( C1-assets ) )
      LOG-TO-FILE ( word "  Residual receivable ----------- " ( S1-rrir-assets ) )
    ]

    ;; Collect interest payments on excess reserve deposits.
    if( S1-erir-assets > 1 )
    [
      let the-crb ( crb crb-who )
      LOG-TO-FILE ( word "INTEREST PAYMENT ON ER:" )
      LOG-TO-FILE ( word "  CRB " ( [who] of the-crb ) " to bank " who "." )
      LOG-TO-FILE ( word "  Bank C1 assets ---------------- " C1-assets )
      LOG-TO-FILE ( word "  Bank L1 debts ----------------- " L1-debts )
      LOG-TO-FILE ( word "  CRB C1 assets ----------------- " ( [C1-assets] of the-
crb ) )
      LOG-TO-FILE ( word "  Current receivable ------------ " ( S1-erir-assets ) )
      f-cbsvcs-bank-paid-monthly-interest-on-er-deposits
      LOG-TO-FILE ( word "  CRB C1 assets ----------------- " ( [C1-assets] of the-
crb ) )
      LOG-TO-FILE ( word "  Bank C1 assets ---------------- " ( C1-assets ) )
      LOG-TO-FILE ( word "  Residual receivable ----------- " ( S1-erir-assets ) )
    ]

  ] ;; End ask banks
;;  end of f-process-interest-payments-monthly
end

;;----------------------------------------------------------------------------|
;; Process payments on loans.
to f-process-payments-on-loans-monthly
;; This routine is to be executed by the observer.

  ;; Monthly loan payments of principal will be made by check
  ;;   from/to the loan accts.

  ;; The GCRA can make a payment on L1 loans.
  ask gcras with [L1-loan-debts > 0]
  [
    LOG-TO-FILE ( word "GCRA'S PAYMENT ON L1 BANK LOAN" )
    f-bsvcs-agent-makes-a-payment-on-loan
  ]

  ;; Prsns can make payments on L1 loans.
  ask prsns with [L1-loan-debts > 0]
  [
    LOG-TO-FILE ( word "PRSN-" who "'S PAYMENT ON L1 BANK LOAN" )
    f-bsvcs-agent-makes-a-payment-on-loan
  ]

  ;; Corps can make payments on L1 loans.
  ;; TODO: Not implemented yet.
  ;; ask corps with [L1-loan-debts > 0]
  ;; [
  ;;   LOG-TO-FILE ( word "CORP-" who "'S PAYMENT ON L1 BANK LOAN" )
  ;;   f-bsvcs-agent-makes-a-payment-on-loan
  ;; ]

;; end of f-process-payments-on-loans-monthly
end

;;----------------------------------------------------------------------------|
;; Government taxes and spends.
to f-government-spends-and-taxes-monthly
;; This routine is to be executed by the observer.

  ask gcras
  [
    ;; Tax first, spend second.  Ensures money is in the coffers.
    f-government-collects-taxes
    f-government-spends-money
  ]

;; end of f-government-spends-and-taxes-monthly
end

;;-----------------------------------------------------------------------------|
;; Government spends money.
to f-government-spends-money
;; This routine is to be executed the GCRA.

  ;; THEORY:
  ;; This applies to this routine, and also to f-government-collects-taxes.
  ;;
  ;; How government spending and taxes are implemented are a matter of social
  ;;   policy.  Of course the government performs services when money is spent,
  ;;   but as long as the money goes back into its own economy, efficiency of
  ;;   of delivery of those services is somewhat irrelevant to the economy.
  ;; Taxing and spending are a means to re-distribute the money from some agents
  ;;   to other agents.  If that also happens to build infrastructure, good.
  ;; So, I tax a slider-determined % based on net-worth-priv values.  Taxes
  ;;   are collected monthly, so, e.g., a 1% tax rate amounts to 12% annual tax.
  ;; Then I spend a fixed amount on each person.  This is as if they receive
  ;;   a regular wage, independent of their wealth.
  ;; The result is I redistribute money from the most wealthy to the most poor.
  ;;   For example, I will tax a large amount from a wealthy person and pay
  ;;   back a modest wage, while a poor person will pay little and receive a
  ;;   modest wage.
  ;; If you vary the tax rate, and the wage rate, then you should be able to
  ;;   effectively resist the effects of entropy production (inequitable
  ;;   distribution of wealth).
  ;; To achieve the best effect, I need to set the taxes and expenditures to
  ;;   roughly equal.  I.e. I need to balance the monthly gov't budget.

  LOG-TO-FILE ( word "" )
  LOG-TO-FILE ( word "GCRA SPENDS MONEY" )
  ;; Government spends by paying a wage to prsns.
  ;; The government will spend all of its assets.
  ;; I am assuming that taxes have been collected previously and are waiting
  ;;   to be spent.

  ;; Contact the bank of the GCRA.
  let gcra-bank ( bank bank-who )

  ;; Determine what the monthly wage will be.
  ;; All monies are spent.  The budget is balanced.
  let monthly-wage round( L1-assets / g-no-of-prsns )
  ;; Initialize an aggregate variable.
  let wages-paid 0

  LOG-TO-FILE ( word "  GCRA L1 assets prior to payments -- " L1-assets )
  LOG-TO-FILE ( word "  Monthly wage ---------------------- " monthly-wage )

  ;; This functions like a prsn-to-prsn check, and requires six entries.
  ;;   Two in client's check books.  Four in bank back room records.
  ask prsns
  [
    ;; Contact bank
    let prsn-bank ( bank bank-who )
```

```
    ;; Put money into prsn's bank account.  Entry #1.
    ask prsn-bank [ set L1-debts ( L1-debts + monthly-wage ) ]
    ;; Assets follow debts.  Entry #2.
    ask prsn-bank [ set L1-assets ( L1-assets + monthly-wage ) ]
    ;; Enter the deposit into prsns check-book.  Entry #3.
    ;; At this point the net change in prsn-bank is zero.
    LOG-TO-FILE ( word "  PRSN " who " L1 assets prior to payment - " L1-assets )
    set L1-assets ( L1-assets + monthly-wage )
    LOG-TO-FILE ( word "  PRSN " who " L1 assets after payment ---- " L1-assets )

    ;; Enter the payment into the gov't tally-book.
    set wages-paid ( wages-paid + monthly-wage )
  ]
  ;; Remove the money from GCRA bank account.  Entry #4.
  ask gcra-bank [ set L1-debts ( L1-debts - wages-paid ) ]
  ;; Assets follow debts.  Entry #5.
  ask gcra-bank [ set L1-assets ( L1-assets - wages-paid ) ]
  ;; At this point the net change in gcra-bank is zero.
  ;; Note the payments in the gov't check book.  Entry #6.
  set L1-assets ( L1-assets - wages-paid )
  LOG-TO-FILE ( word "  Total wages paid ------------------ " wages-paid )
  LOG-TO-FILE ( word "  GCRA L1 assets after all payments - " L1-assets )

  ;; TODO: When I start taxing banks and corps, I need to add payments
  ;;   to banks and corps.

;; end of f-government-spends-money
end

;;-----------------------------------------------------------------------------|
;; Government collects a tax of net worth.
to f-government-collects-taxes
;; This routine is to be executed by the GCRA.

  if( g-net-worth-tax-rate > 0 )
  [
    ;; THEORY: See the routine f-government-spends-money for a complete
    ;;   description of the approach to government taxing and spending.

    ;; The government collects a "net worth" tax and puts it into its
    ;;   "Government Consolidated Revenue Account", i.e. its GCRA.
    ;; It does not tax GCRA or crb accounts.
    ;; Private CRB "C" accounts are considered a sub-account of GCRA.

    ;; TODO: Add taxes for corps and private bank worth.

    ;; Identify the bank of the GCRA.
    ;; The GCRA is not a bank.  It keeps its accounts in a commercial bank.
    let gcra-bank ( bank bank-who )

    let taxes-due 0        ;; Initialize a working variable.
    let all-taxes-paid 0   ;; initialize an aggregate to collect all taxes paid.

    ;; This functions like a prsn-to-prsn check, and requires six entries.
    ;;   Two in client's check books.  Four in bank back room records.
    ask prsns
    [
      LOG-TO-FILE ( word "PRSN " who " PAYS TAXES" )
      f-compute-prsn-net-worth
      LOG-TO-FILE ( word "  Prsn net worth -------------------- " net-worth-priv )
      set taxes-due round( net-worth-priv * g-net-worth-tax-rate / 100 )
      ;; Taxes are paid by bank-to-bank check.
      ;; Contact the prsn's bank.
      let prsn-bank ( bank bank-who )

      LOG-TO-FILE ( word "  Prsn L1 assets before payment ----- " L1-assets )
      ;; Remove taxes from prsns bankbook.  Entry #1.
      set L1-assets ( L1-assets - taxes-due )
      ;; Remove the taxes from the prsns checking account. Entry #2.
      ask prsn-bank [ set L1-debts ( L1-debts - taxes-due ) ]
      ;; Assets follow debts.  Entry #3.
      ask prsn-bank [ set L1-assets ( L1-assets - taxes-due ) ]
      ;; Record the amount as paid, for later entry to GCRA bankbook.
      ;; At this point the net change in prsn-bank is zero.
      set all-taxes-paid ( all-taxes-paid + taxes-due )
      LOG-TO-FILE ( word "  Taxes paid ------------------------ " taxes-due )
      LOG-TO-FILE ( word "  Prsn L1 assets after payment ------ " L1-assets )
    ]
    LOG-TO-FILE ( word "  GCRA L1 assets before collection -- " L1-assets )
    LOG-TO-FILE ( word "  Total of all taxes collected ------ " all-taxes-paid )
    ;; Government adjusts its own bankbook.  Entry #4.
    set L1-assets ( L1-assets + all-taxes-paid )
    ;; Add the money to the gov't checking account.  Entry #5.
    ask gcra-bank [ set L1-debts ( L1-debts + all-taxes-paid ) ]
    ;; Assets follow debts.  Entry #6.
    ask gcra-bank [ set L1-assets ( L1-assets + all-taxes-paid ) ]
    ;; At this point the net change in gcra-bank is zero.
    LOG-TO-FILE ( word "  GCRA L1 assets after collection --- " L1-assets )

    ;; TODO: Add taxes on corporations.
    ;; TODO: Add taxes on private net worth of banks.
  ]

;; end of f-government-collects-taxes
end

;;-----------------------------------------------------------------------------|
;; Everybody visits their bank.
to f-everybody-visits-their-bank
;; This routine is to be executed by the observer.
;; It is executed on setup, and monthly.

  LOG-TO-FILE ( word "  EVERYBODY VISITS BANK" )
  ;; The prsns and corps must visit their banks.
  f-prsns-visit-banks-daily

  ;; TODO: Add corps here.
  ;; f-corps-visit-banks-daily

;; end of f-everybody-visits-their-bank
end

;;-----------------------------------------------------------------------------|
;; Each prsn has accounts with one bank.
to f-prsns-visit-banks-daily
;; This routine is to be executed by the observer.

  ask prsns
  [
    ;; The following routine is used for daily visits, but also for setup,
    ;;   and to "initialize" new prsns.
    f-prsn-visits-a-bank
  ]

;; end of f-prsns-visit-banks-daily
end

;;-----------------------------------------------------------------------------|
;; A prns deposits cash into an L1 (checking) account and moves it about.
to f-prsn-visits-a-bank
  ;; This routine is to be executed by a prsn.

  ;; This routine is used for daily visits, but also for setup,
  ;;   and to initialize new prsns.

  ;; THEORY: The money must be shifted from the broadest categories towards the
  ;;   most narrow categories to be useful when needed.  Each shift requires
  ;;   an assessment of needs and supply all of the way up the chain.
  ;;   That is tricky and tedious, and prone to coding error.
  ;; The easiest way to handle it is to work through the categories of money
  ;;   from L0, L1, L2 to loan, and at each step, (PART A) deposit all of
  ;;   it to the next broader category of money, and then (PART B) determine
  ;;   what is needed and move that much back. Ultimately any shortage must
  ;;   come from a bank loan if possible, and any overage goes to savings.
  ;; This approach depends on the use of negatives to handle subtractions
  ;;   implicitly, and so makes for much simpler code.

  ;; Contact the bank.
  let my-bank ( bank bank-who )
  LOG-TO-FILE ( word "PRSN " who " VISITS BANK " bank-who "." )

  let affected-assets ( L0-assets + L1-assets + L2-assets )
  LOG-TO-FILE ( word "  My P0-assets were ------------- " P0-assets )
  LOG-TO-FILE ( word "  My L0-assets were ------------- " L0-assets )
  LOG-TO-FILE ( word "  My L1-assets were ------------- " L1-assets )
  LOG-TO-FILE ( word "  My L2-assets were ------------- " L2-assets )
  LOG-TO-FILE ( word "  Total affected assets --------- " affected-assets )

  ;; ----------------------------------
  ;; Establish appropriate P0/L0 holdings.
  ;; ----------------------------------
  ;; (PART A) Deposit all cash.
  ASSERT ( P0-assets = L0-assets ) "Bad cash" who
  f-bsvcs-prsn-deposits-cash L0-assets
  LOG-TO-FILE ( word "  My P0-assets are -------------- " P0-assets )
  LOG-TO-FILE ( word "  My L0-assets are -------------- " L0-assets )

  ;; (PART B) Remove required amount of cash.
  f-bsvcs-prsn-withdraws-cash g-p-daily-L0-allocation
  LOG-TO-FILE ( word "  My P0-assets are -------------- " P0-assets )
  LOG-TO-FILE ( word "  My L0-assets are -------------- " L0-assets )

  ;; ----------------------------------
  ;; Establish appropriate L1 holdings.
  ;; ----------------------------------
  ;; (PART A) Deposit all checking into savings.
  LOG-TO-FILE ( word "  My L1-assets are -------------- " L1-assets )
  f-bsvcs-prsn-moves-L1-to-L2 L1-assets
  LOG-TO-FILE ( word "  My L1-assets are -------------- " L1-assets )

  ;; (PART B) Put required amount of money back into L1.
  f-bsvcs-prsn-moves-L2-to-L1 g-p-daily-L1-allocation
  LOG-TO-FILE ( word "  My L1-assets are -------------- " L1-assets )

  ;; ----------------------------------
  ;; Establish appropriate L2 holdings.
  ;; ----------------------------------
  ;; THEORY: This will be different.  Savings cannot be negative.
  ;; A prsn must maintain sufficient money in checking to get
  ;;   throught a typical day (as determined by the standard
  ;;   allocations), and this is done from the savings.  When
  ;;   savings fall below zero, it must be topped up by a bank
  ;;   loan of a standard size.  If the bank has insufficient
  ;;   cash reserves, then it can no longer offer loans, and
  ;;   the prsn becomes insolvent (bankrupt).

  LOG-TO-FILE ( word "  Pre-loan - My L2-assets are --- " L2-assets )
  ;; This routine will determine:
  ;;   - if a loan is needed to top up the L2 assets.
  ;;   - if the bank has excess reserves.
  ;;   - size of the loan.
  ;;   - whether the bank can continue to make loans.
  ;;   - if this agent is solvent or insolvent.
  f-bsvcs-prsn-negotiates-an-L1-loan
  LOG-TO-FILE ( word "  Post-loan - My L0-assets are -- " L0-assets )
  LOG-TO-FILE ( word "  Post-loan - My L1-assets are -- " L1-assets )
  LOG-TO-FILE ( word "  Post-loan - My L2-assets are -- " L2-assets )
  ;; Note, the amount of the loan is placed in the agent's
  ;;   L1 checking account, and is moved to savings the next
  ;;   time the agent visits a bank, using this procedure.
  set affected-assets ( L0-assets + L1-assets + L2-assets )
  LOG-TO-FILE ( word "  Total affected assets --------- " affected-assets )

  ;; End of f-prsn-visits-a-bank
end

;;-----------------------------------------------------------------------------|
;; The CRB supervises the management of reserve deposits.
to f-the-crb-reconciles-with-banks-daily
;; This routine is to be executed by the observer.

  LOG-TO-FILE ( word "" )
  LOG-TO-FILE ( word "CRB RECONCILES RESERVE DEPOSITS" )

  let crb-bank ( one-of crbs ) ;; More efficient this way.
  ask banks
  [
    LOG-TO-FILE ( word "BANK " who )

    LOG-TO-FILE ( word "  L1-loan-assets ---------------- " L1-loan-assets )
    LOG-TO-FILE ( word "  Old settings:" )
    LOG-TO-FILE ( word "  P0-vc-assets ------------------ " P0-vc-assets )
    LOG-TO-FILE ( word "  P0-rr-assets ------------------ " P0-rr-assets )
    LOG-TO-FILE ( word "  P0-er-assets ------------------ " P0-er-assets )
    let ttl-reserves ( P0-vc-assets + P0-rr-assets + P0-er-assets )
    LOG-TO-FILE ( word "  Total reserves ---------------- " ttl-reserves )

    ;; This bank controls limited reserves of cash

    ;; I am going to withdraw all CRB deposits and re-deposit the correct amounts.
    ;;   This is instead of shifing cash from place to place, which gets tricky.
    ;; This handles any negatives that may have occured
    ;;   in the course of business.
    f-cbsvcs-bank-moves-er-to-vc P0-er-assets
    f-cbsvcs-bank-moves-rr-to-vc P0-rr-assets

    ;; Deposit the required reserves first.
    ;; The given required reserve ratio is a percentage.
    ;; We need a numeric factor.  Convert percentage to numeric factor.
    let rr-factor ( g-reserve-requirement-ratio / 100 )
    let needed-rr-deposits floor( L1-loan-assets * rr-factor )
    if( needed-rr-deposits > ttl-reserves )
    [
      set needed-rr-deposits ttl-reserves
    ]
    f-cbsvcs-bank-moves-vc-to-rr needed-rr-deposits
    let remaining-reserves ( ttl-reserves - needed-rr-deposits )

    ;; Now I save some in the vault.
    let my-vc g-minimum-vault-cash
    if( my-vc > remaining-reserves )
    [
      set my-vc remaining-reserves
    ]
    set remaining-reserves ( remaining-reserves - my-vc )

    ;; The rest is excess reserves.
    f-cbsvcs-bank-moves-vc-to-er remaining-reserves
    LOG-TO-FILE ( word "  New settings:" )
    LOG-TO-FILE ( word "  P0-vc-assets ------------------ " P0-vc-assets )
    LOG-TO-FILE ( word "  P0-rr-assets ------------------ " P0-rr-assets )
    LOG-TO-FILE ( word "  P0-er-assets ------------------ " P0-er-assets )
    Set ttl-reserves ( P0-vc-assets + P0-rr-assets + P0-er-assets )
    LOG-TO-FILE ( word "  Total reserves ---------------- " ttl-reserves )
    ifelse( P0-er-assets > 0 )
    [
      set b-bank-can-make-loans 1
      LOG-TO-FILE ( word "  Bank loan dept status - OPEN" )
    ]
    ;; Else
    [
      set b-bank-can-make-loans 0
      LOG-TO-FILE ( word "  Bank loan dept status - CLOSED" )
    ]
  ]

;; end of f-the-crb-reconciles-with-banks-daily
end

;;-----------------------------------------------------------------------------|
;; D6 Process all end-of-day banking activities.
;;-----------------------------------------------------------------------------|
to do-banking
  ;; This routine is to be executed by the observer.

  if( gb-debug-on = 1 )
  [
    ifelse( ( gs-debug-step-chooser = "all" ) or ( gs-debug-step-chooser = "banking"
) )
    [ set gb-debug-flow-on 1 LOG-TO-FILE "" LOG-TO-FILE word "Do-banking: Debug on;
tick = " ticks ]
    [ set gb-debug-flow-on 0 ]
  ]

  f-everybody-visits-their-bank
  ;; The visit to the bank can set prsn or bank bankruptcy flags.

  ;; TODO: also banks and corps, when implemented.  Banks may open savings
  ;;   accounts, as may corps?
  ;; Banks will now have odd reserves, and will need to reconcile them.
  ;; The government records need to be reconciled with bank records.
  ;;   The CRB reconciles reserve deposits with each bank daily.
  f-the-crb-reconciles-with-banks-daily

  ;; Banks may have been exhausted of their last abilities to earn C1-assets.
  ;; This sets a bankruptcy flag for banks.
  f-bsvcs-bank-checked-for-bankruptcy

  ;; Process bankruptcies of prsns.
  let prsn-bankruptcies ( prsns with [b-prsn-is-bankrupt = 1] )
  ask prsn-bankruptcies
  [
    f-bsvcs-process-prsn-bankruptcy
  ]

  ;; Process bankruptcies of banks.
  let bank-bankruptcies ( banks with [b-bank-is-bankrupt = 1] )
  ask bank-bankruptcies
  [
    f-bsvcs-process-bank-bankruptcy
  ]

  ;; TODO: Also corps, when implemented.

;; end of do-banking
end

;;---------------------------------------------------------------------------|
;; D7 - do-post-tick procedure(s)
;;---------------------------------------------------------------------------|
to do-post-tick
  ;; This routine is to be executed by the observer.

  if( gb-debug-on = 1 )
  [
    ifelse( ( gs-debug-step-chooser = "all" ) or ( gs-debug-step-chooser = "post-
tick" ) )
    [ set gb-debug-flow-on 1  LOG-TO-FILE "" LOG-TO-FILE word "Do-Post-tick: Debug
on; tick = " ticks ]
    [ set gb-debug-flow-on 0 ]
  ]

  ;; This code ensures that the number of banks active in the economy
  ;;   matches the numbers implied by the sliders.
  ;; Missing banks are created.  Overages are allowed to fall by
  ;;   attrition, through bankruptcies.
  set g-no-of-prsns-max ( g-no-of-prsns-per-bank * g-no-of-banks-max )
  let no-of-banks ( count banks )
  while[ no-of-banks < g-no-of-banks-max ]
  [
    ;; Create a new bank, and it as an average bank.
    f-bank-is-funded-as-average
    set no-of-banks ( count banks )
  ]

  ;; This code ensures that the number of prsns active in the economy
  ;;   matches the numbers implied by the sliders.
  ;; Missing prsns are created.  Overages are allowed to fall by
  ;;   attrition, through bankruptcies.
  ;; Recompute the expected number of prsns, given that the slider may
```

```
  ;;   have been changed.
  set g-no-of-prsns-max ( g-no-of-prsns-per-bank * g-no-of-banks-max )
  set g-no-of-prsns ( count prsns )
  while[ g-no-of-prsns < g-no-of-prsns-max ]
  [
    ;; Create a new prsn, and fund him/her as an average prsn.
    f-prsn-is-funded-as-average
    set g-no-of-prsns ( count prsns )
  ]

  ;; MANUAL CHANGE FOR DEBUG.
  ;; This is a call to a debug routine which could be suppressed if all is okay.
  ;; This is one of a group of such calls, most of which are between steps in
  ;;   the 'Go' routine.  They are suppressed there, but can be enabled again.
  ;; I have decided to leave this one active, for now.
  ;; It checks all agents, every tick.
  if( frb-agents-are-all-valid = false )
    [ LOG-TO-FILE ( word "Agents failed validity test." ) ]

  ;; Update the aggregates for display in the monitors.
  f-update-aggregates

  display

  LOG-TO-FILE "  Do-post-tick: procedure completed."
end

;;---------------------------------------------------------------------------|
;; A new prsn is created and funded as an average prsn.
to f-prsn-is-funded-as-average
;; This routine is to be executed by the observer.

  ;; TODO: After debugging, suppress this.
  ;; f-force-debug-output-on
  ;; TODO: Remove this if annoying.
  ;; beep

  ;; I am interested in the steady-state distribution of wealth, so I don't
  ;;   want to bias the distribution by adding a new prsn that is either too
  ;;   wealthy or too poor.  Neither do I want to change the MS-1 money supply
  ;;   (I.e. the physical money base).  So, I have this three-step process
  ;;   to construct a new prsn.
  ;; Step 1 - the population is canvassed to determine total wealth.
  ;; Step 2 - the population is taxed to gather sufficient L1-assets.
  ;; Step 3 - the prsn is fashioned as a prsn of average wealth.
  ;;
  ;; The impact of this approach should be that L1-assets are transferred
  ;;   to the prsn, causing the relative distribution to remain the same,
  ;;   but translating/shifting the distribution.  I could do step 2 in two
  ;;   ways:
  ;;   - I could pro-rate the contribution from each prsn.  This would have
  ;;     the effect of making the distribution more compact.  Those with
  ;;     the greatest debt or wealth would experience the greatest movement
  ;;     towards zero wealth, while those with little wealth would not be
  ;;     affected much.
  ;;   OR
  ;;   - I could collect a standard fixed sum from each prsn.  This would
  ;;     have the effect of translating the entire population towards
  ;;     zero wealth.  All would benefit or suffer equally, depending on
  ;;     whether the average wealth was negative or positive respectively.
  ;;
  ;; I have implemented the pro-rated version of Step 2.
  ;; TODO: After debugging, remove this.
  ;; Toggle debug on.
  ;; let old-debug gb-debug-on
  ;; set gb-debug-on 0
  ;; f-toggle-debug
  ;; set gb-debug-show-steps true

  LOG-TO-FILE ( word "Creating a new prsn." )
  ;; STEP 1 - Find the total net worth of all prsns.
  ask prsns [ f-compute-prsn-net-worth ]
  let total-net-worth ( sum [net-worth-priv] of prsns )
  let mean-net-worth ( mean [net-worth-priv] of prsns )
  let current-no-of-prsns ( count prsns )
  ;; Adjust for intended additional prsn.
  let target-net-worth
    ( mean-net-worth * current-no-of-prsns / ( 1 + current-no-of-prsns ) )
  LOG-TO-FILE ( word "  Current no of prsns ----------- " current-no-of-prsns )
  LOG-TO-FILE ( word "  Total net worth of prsns ------ " total-net-worth )
  LOG-TO-FILE ( word "  Target net worth of new prsn -- " target-net-worth )

  let total-collected 0
  let donation-factor 0
  let amount-due      0

  create-prsns 1
  [
    set g-counts-p-births ( g-counts-p-births + 1 )
    f-initialize-new-prsn
    set heading 90
    ;; Move to a random point.
    setxy random-xcor random-ycor
    ;; Although initialization simply adds a bank-who variable to prsn,
    ;;   it effectively opened a checking and savings account.  The
    ;;   money will be moved into its checking account.

    ask other prsns
    [
      ;; Canvass each prsn and collect the appropriate assets (debts?)
      ;; The signs on the numbers are important here.  Either part of the
      ;;   following ratio may be negative.  The effect is that poor prsns
      ;;   with negative net worth will be given a little, while rich prsns
      ;;   with positive net worth will have some taken.
      set donation-factor ( net-worth-priv / total-net-worth )
      set amount-due round( target-net-worth * donation-factor )
      ;; A rounded figure to keep things tidy.
      LOG-TO-FILE ( word "  Net-worth-priv ---------------- " net-worth-priv )
      LOG-TO-FILE ( word "  Donation-factor --------------- " donation-factor )
      LOG-TO-FILE ( word "  Amount-due -------------------- " amount-due )

      ;; Contact other prsn's bank.
      let his-bank ( bank bank-who )
      ;; Mark payment in his check book.  Entry #1.
      set L1-assets ( L1-assets - amount-due )
      ;; Inform his bank that a check was written.  Entries #2 and #3.
      ask his-bank
      [
        set L1-assets ( L1-assets - amount-due )
        set L1-debts ( L1-debts - amount-due )
      ]
      ;; The net worth of the bank does not change.  The net worth of
      ;;   the doner of the cash does change.
      ;; Keep a running record of the donations.
      set total-collected ( total-collected + amount-due )
      ;; Some of the amounts collected may have been negative.
      ;;   That is OK.
    ]  ;; end ask other prsns

    ;; The collection is now done.  The new prsn deposits it into a
    ;;   checking account at his/her bank.
    LOG-TO-FILE ( word "  Total-collected --------------- " total-collected )
    ;; Enter it into the personal check book.  Entry #4.
    set L1-assets ( L1-assets + total-collected )
    ;; Contact the bank
    let my-bank ( bank bank-who )
    ;; Deposit the aggregate check into the checking account.
    ;;   Entries #5 and #6.
    ask my-bank
    [
      set L1-assets ( L1-assets + total-collected )
      set L1-debts ( L1-debts + total-collected )
    ]
    ;; This prsn now has a large pile of money, or a large debt,
    ;;   recorded in their checking account.  They need to either
    ;;   move some to savings and currency, or take out a bank loan
    ;;   to cover the debt and get them back ready for action in the
    ;;   economy.  Either way, they should have average net worth.
    f-prsn-visits-a-bank
    ;; They now have cash, and money in checking and savings accounts,
    ;;   and possibly a bank loan that provides those funds.
  ]
  set g-no-of-prsns ( count prsns )

  ;; TODO: Remove this after debug.
  ;; f-force-debug-output-off

;; end of f-prsn-is-funded-as-average
end

;;-----------------------------------------------------------------------------|
;; A new bank is created and funded as an average bank.
to f-bank-is-funded-as-average
;; This routine is to be executed by the observer.

  ;; TODO: After debugging, suppress this.
  ;; f-force-debug-output-on
  ;; TODO: Remove this if annoying.
  ;; beep

  ;; I am interested in the steady-state distribution of wealth, so I don't
  ;;   want to bias the distribution by adding a new bank that is either too
  ;;   wealthy or too poor.  Neither do I want to change the MS-1 money supply
  ;;   (I.e. the physical money base).  So, I have this nine-step process
  ;;   to construct a new bank:
  ;; Step 1 - Assemble sufficient L1-assets;
  ;; Step 2 - Assemble sufficient P0-assets;
  ;; Step 3 - Assemble sufficient clients.
  ;;
  ;; Each of the above steps has three sub-steps:
  ;; Step A - the population is canvassed to determine total assets.
  ;; Step B - the population is taxed to gather sufficient assets.
  ;; Step C - the bank is fashioned as a bank of average assets.
  ;;
```

```
;; The impact of this approach should be that P0 and L1-assets are transferred
;;   to the bank, causing the relative distribution to remain the same,
;;   but translating/shifting the distribution.  I could do step 2 in two
;;   ways:
;;   - I could pro-rate the contribution from each bank.  This would have
;;     the effect of making the distribution more compact.  Those with
;;     the greatest debt or wealth would experience the greatest movement
;;     towards zero wealth, while those with little wealth would not be
;;     affected much.
;;   OR
;;   - I could collect a standard fixed sum from each bank.  This would
;;     have the effect of translating the entire population towards
;;     zero wealth.  All would benefit or suffer equally, depending on
;;     whether the average wealth was negative or positive respectively.
;;
;; I have implemented the pro-rated version of Step 2.

;; TODO: QQQ After debugging, remove this.
;; Toggle debug on.
let old-debug gb-debug-on
set gb-debug-on 0
f-toggle-debug
set gb-debug-show-steps true

LOG-TO-FILE ( word "Creating a new bank." )
;; STEP 1 - Assemble C1 assets.
;; Step 1A - Canvass population for wealth.
ask banks [ f-compute-bank-net-worth ]
let total-net-worth ( sum [net-worth-priv] of banks )
let mean-net-worth ( mean [net-worth-priv] of banks )
set g-no-of-banks ( count banks )
;; Adjust for intended additional bank.
let target-net-worth
  ( mean-net-worth * g-no-of-banks / ( 1 + g-no-of-banks ) )
LOG-TO-FILE ( word "  Current no of banks ----------- " g-no-of-banks )
LOG-TO-FILE ( word "  Total net worth of banks ------ " total-net-worth )
LOG-TO-FILE ( word "  Target net worth of new bank -- " target-net-worth )

;; Step 1B - Collect the C1-assets.
let total-C1-collected 0
let C1-donation-factor 0
let amount-C1-due      0
let new-bank one-of banks ;; A dummy assignment.

create-banks 1
[
  set g-counts-b-births ( g-counts-b-births + 1 )
  set new-bank ( self )  ;; Create a handle for the new bank.
  LOG-TO-FILE ( word "  Bank <<<" ( [who] of new-bank ) ">>> created." )

  f-initialize-new-bank
  set heading 90
  ;; Move to a random point.
  setxy random-xcor random-ycor

  ask other banks
  [
    ;; STEP 1B - Canvass each bank and collect the appropriate C1-assets.
    ;; The signs on the numbers are important here.  Either part of the
    ;;   following ratio may be negative.  The effect is that poor prsns
    ;;   with negative net worth will be given a little, while rich prsns
    ;;   with positive net worth will have some taken.
    set C1-donation-factor ( net-worth-priv / total-net-worth )
    set amount-C1-due round( target-net-worth * C1-donation-factor )
    ;; Rounded figures to keep things tidy.

    LOG-TO-FILE ( word "  Net-worth-priv ---------------- " net-worth-priv )
    LOG-TO-FILE ( word "  C1-donation-factor ------------ " C1-donation-factor )
    LOG-TO-FILE ( word "  Amount-C1-donated ------------- " amount-C1-due )

    ;; Mark payment in this doner bank's check book.  Entry #1.
    set C1-assets ( C1-assets - amount-C1-due )
    ;; Inform back room that a check was written.  Entries #2 and #3.
    set L1-assets ( L1-assets - amount-C1-due )
    set L1-debts ( L1-debts - amount-C1-due )

    ;; Step 1C - Install the C1-assets in the new bank.
    ;; Inform recipient bank that a check was written.  Entries #4, #5 and #6.
    ask new-bank
    [
      set C1-assets ( C1-assets + amount-C1-due )
      set L1-assets ( L1-assets + amount-C1-due )
      set L1-debts ( L1-debts + amount-C1-due )
    ]
    ;; The net worth of the back room of banks does not change.  The
    ;;   net worth of the front rooms does change.

    ;; Keep a running record of the donations.
    set total-C1-collected ( total-C1-collected + amount-C1-due )
    ;; Some of the amounts collected may have been negative.
    ;;   That is OK.
  ]  ;; end ask other banks

  ;; The collection is now done.
  LOG-TO-FILE ( word "  Total-C1-donated -------------- " total-C1-collected )
  ;; This bank now has a large pile of money, or a large debt,
  ;;   recorded in their checking account.
]  ;; end of create-banks 1

;; The observer takes over again.

set g-no-of-banks ( count banks )

;; STEP 2 - Collect a fair share of physical money (P0).
;; Step 2A - Canvass the banks to determine total P0-assets.
;; This has to be a little different, because between Steps 1A and 2A
;;   the new bank has been created.
ask banks [ f-compute-bank-net-worth ]
let total-P0 0  ;; a dummy declaration.
let mean-P0  0  ;; a dummy declaration.
let no-of-other-banks 0  ;; a dummy declaration.
ask new-bank
[
  ;; This excludes the data for the new-bank, which should be zero
  ;;   in any case.
  set total-P0 ( sum [P0-all-assets] of other banks )
  set mean-P0 ( mean [P0-all-assets] of other banks )
  set no-of-other-banks ( count other banks )
  ;; Adjust for intended additional bank.
]
let target-P0
  floor( mean-P0 * no-of-other-banks / ( 1 + no-of-other-banks ) )
LOG-TO-FILE ( word "  Current no of banks ----------- " g-no-of-banks )
```

```
LOG-TO-FILE ( word "  Total P0-assets of banks ------ " total-P0 )
LOG-TO-FILE ( word "  Target P0-assets of new bank -- " target-P0 )

;; Step 2B - Collect physical P0-assets.
let total-P0-collected 0
let P0-donation-factor 0
let amount-P0-due      0

ask new-bank
[
  ;; This trick excludes the new-bank from making a donation.
  ask other banks
  [
    ;; Canvass each bank and collect the appropriate physical assets (P0).
    ;; The signs on the numbers are all positive here.  The effect is that
    ;;   poor banks with few physical assets will lose a little, while rich
    ;;   banks with large physical assets will lose a lot.

    set P0-donation-factor ( P0-all-assets / total-P0 )
    set amount-P0-due round( target-P0 * P0-donation-factor )
    ;; Rounded figures to keep things tidy.
    LOG-TO-FILE ( word "  P0 all assets ----------------- " P0-all-assets )
    LOG-TO-FILE ( word "  P0-donation-factor ------------ " P0-donation-factor )
    LOG-TO-FILE ( word "  Amount-P0-donated ------------- " amount-P0-due )

    ;; Remove from doner bank.  Entry #1.
    set P0-vc-assets ( P0-vc-assets - amount-P0-due )

    ;; Step 2C - Add the assets to the new bank.
    ;; Add to recipient bank's bank vault.  Entry #2.
    ask new-bank
    [
      set P0-vc-assets ( P0-vc-assets + amount-P0-due )
    ]
    ;; Keep a running record of the donations.
    set total-P0-collected ( total-P0-collected + amount-P0-due )
  ]  ;; end ask other banks
] ;; end ask new-bank

;; The collection is now done.
LOG-TO-FILE ( word "  Total-P0-donated -------------- " total-P0-collected )
;; end of Step 2 - Collect physical assets (P0).

;; The observer takes over again.

;; Step 3 - Now we have to gather some clients from other banks.
;; Step 3A - Determine how many clients there are.
set g-no-of-prsns ( count prsns )  ;; Probably redundant
let target-no-of-clients ( g-no-of-prsns / g-no-of-banks )
let clients-gathered 0

;; Steps 3B and 3C - These will be done together.
let client-factor 0  ;; a dummy declaration.
let clients-due   0  ;; a dummy declaration.

ask new-bank
[
  ask other banks
  [
    set client-factor ( no-of-prsn-clients / g-no-of-prsns )
    ;; Rounded to keep things tidy.
    set clients-due round( target-no-of-clients * client-factor )
```

```
      ;; For each bank I have to randomly select a subset of clients
      ;;   and transfer them to the new bank.
      let other-bank self  ;; Give the bank in control an explicit handle.
      let other-bank-who ( [who] of self )
      let prsn-client-set ( prsns with [bank-who = other-bank-who] )
      ;; Select a random subset of size clients-due.
      set prsn-client-set ( n-of clients-due prsn-client-set )
      ask prsn-client-set
      [
        ;; Ask each prsn to transfer its accounts to the new bank.
        ;; The prsn is a client of other-bank.
        ;; Each transfer requires four entries.  The client's bank book does
        ;;   not need to be changed, but it is the reference that determines
        ;;   the amount of assets to be moved.
        LOG-TO-FILE ( word "  Prsn " who " transferred." )
        let amount-to-move L1-assets  ;; From bank book.
        ask other-bank
        [
          set L1-assets ( L1-assets - amount-to-move )
          set L1-debts ( L1-debts - amount-to-move )
        ]
        ask new-bank
        [
          set L1-assets ( L1-assets + amount-to-move )
          set L1-debts ( L1-debts + amount-to-move )
        ] ;; end of ask new-bank
      ] ;; end of ask prsn-client-set
      LOG-TO-FILE ( word "  No of clients transferred ------ "
        ( count prsn-client-set ) )
      set clients-gathered ( clients-gathered + clients-due )
    ] ;; end of ask other banks
  ]  ;; end of ask new-bank
  LOG-TO-FILE ( word "  Total clients transferred ------ " clients-gathered )

  f-the-crb-reconciles-with-banks-daily
  ;; They now have cash, and assets, and clients.
  set g-no-of-banks ( count banks )

  ;; TODO: Remove this after debug.
  ;; f-force-debug-output-off

;; end of f-bank-is-funded-as-average
end

;;----------------------------------------------------------------------------|
;; COMPUTATION OF NET WORTH OF ALL AGENTS
;;----------------------------------------------------------------------------|

;;----------------------------------------------------------------------------|
;; Compute the net worth of each of the agents.
to f-compute-each-net-worth
;; This routine is to be executed the observer.

  LOG-TO-FILE ( word "Each net worth will be computed. " )
  ask gcras [ f-compute-gcra-net-worth ]
  ask crbs  [ f-compute-crb-net-worth ]
  ask banks [ f-compute-bank-net-worth ]
  ask prsns [ f-compute-prsn-net-worth ]
  ask corps [ f-compute-corp-net-worth ]

;; end of f-compute-each-net-worth
```

```
end

;;----------------------------------------------------------------------------|
;; Compute the net worth of the GCRA (Government Consolidated Revenue Accounts).
to f-compute-gcra-net-worth
;; This routine is to be executed the GCRA.

  set ttl-P0-assets   0 ;; aggregate of all physical assets

  set ttl-publ-assets 0
  set ttl-publ-assets ( ttl-publ-assets + L1-assets )
  ;; ss set ttl-publ-assets ( ttl-publ-assets + L2-assets )

  set ttl-publ-debts  0
  set ttl-publ-debts ( ttl-publ-debts + L1-loan-debts )
  ;; ss set ttl-publ-debts ( ttl-publ-debts + L3-debts )

  set net-worth-publ ( ttl-publ-assets - ttl-publ-debts )

  set ttl-priv-assets   0
  set ttl-priv-debts    0
  set net-worth-priv    0

  ;; Money supply aggregates
  set msi-assets        0 ;; Physical money supply
  set msi-debts         0 ;; Physical money supply
  set msii-assets ttl-publ-assets ;; Logical money supply
  set msii-debts  ttl-publ-debts  ;; Logical money supply
  set msiii-assets      0 ;; Shadow money supply
  set msiii-debts ( S1-L1ip-debts ) ;; Shadow money supply
  ;; TODO: When this is non-suppressed, next line is needed instead.
  ;; ss set msiii-debts ( S1-L1ip-debts + S1-L3ip-debts ) ;; Shadow money supply

;; end of f-compute-gcra-net-worth
end

;;----------------------------------------------------------------------------|
;; Compute the net worth of the CRB (Central Reserve Bank).
to f-compute-crb-net-worth
;; This routine is to be executed the crb.

  set ttl-P0-assets 0
  set ttl-P0-assets ( ttl-P0-assets + P0-assets )
  set ttl-P0-assets ( ttl-P0-assets + P0-rr-assets )
  set ttl-P0-assets ( ttl-P0-assets + P0-er-assets )

  set ttl-publ-assets L0-assets
  set ttl-publ-debts  L0-debts
  set net-worth-publ ( ttl-publ-assets - ttl-publ-debts )

  set ttl-priv-assets 0
  set ttl-priv-assets ( ttl-priv-assets + C1-assets )
  ;; xx set ttl-priv-assets ( ttl-priv-assets + c2-assets )

  set ttl-priv-debts 0
  set ttl-priv-debts ( ttl-priv-debts + S1-rrip-debts )
  set ttl-priv-debts ( ttl-priv-debts + S1-erip-debts )

  set net-worth-priv ( ttl-priv-assets - ttl-priv-debts )

  let shadow-money ( S1-rrip-debts + S1-erip-debts )
```

```
  ;; Money supply aggregates
  set msi-assets ttl-P0-assets         ;; Physical money supply
  set msi-debts  P0-debts              ;; Physical money supply
  set msii-assets ttl-priv-assets      ;; Logical money supply
  set msii-debts 0                     ;; Logical money supply
  set msiii-assets 0                   ;; Shadow money supply
  set msiii-debts shadow-money         ;; Shadow money supply

;; end of f-compute-crb-net-worth
end

;;----------------------------------------------------------------------------|
;; Compute the net worth of a bank.
to f-compute-bank-net-worth
;; This routine is to be executed a bank.

  set ttl-P0-assets 0
  set ttl-P0-assets ( ttl-P0-assets + P0-vc-assets )

  ;; This is totalled differently from ttl-P0-assets because this includes
  ;;   some that are offset by P0-xx-debts.  I.e. some of these assets are
  ;;   not in the posession of the bank, and should not be counted here
  ;;   as that would cause double counting.  But the variable P0-all-assets
  ;;   is intended to include all assets under the control of this bank, and
  ;;   not merely those in its posession.  So I include those in the CRB
  ;;   as part of the P0-all-assets variable, based on this bank's records
  ;;   of its CRB deposits.
  set P0-all-assets 0
  set P0-all-assets ( P0-all-assets + P0-vc-assets )
  set P0-all-assets ( P0-all-assets + P0-er-assets )
  set P0-all-assets ( P0-all-assets + P0-rr-assets )

  set ttl-publ-assets 0
  set ttl-publ-assets ( ttl-publ-assets + L1-assets )
  set ttl-publ-assets ( ttl-publ-assets + L1-loan-assets )

  set ttl-publ-debts 0
  set ttl-publ-debts ( ttl-publ-debts + L1-debts )
  set ttl-publ-debts ( ttl-publ-debts + L2-debts )
  ;; ss set ttl-publ-debts ( ttl-publ-debts + L3-debts )

  set net-worth-publ ( ttl-publ-assets - ttl-publ-debts )

  set ttl-priv-assets 0
  set ttl-priv-assets ( ttl-priv-assets + C1-assets )
  set ttl-priv-assets ( ttl-priv-assets + S1-L1ir-assets )
  ;; xx set ttl-priv-assets ( ttl-priv-assets + c2-assets )
  set ttl-priv-assets ( ttl-priv-assets + S1-rrir-assets )
  set ttl-priv-assets ( ttl-priv-assets + S1-erir-assets )

  ;; TODO: Run a bank like a corp.
  ;; Debts equal assets, excluding receivables, because it is it's
  ;;   own bank.
  set ttl-priv-debts 0
  set ttl-priv-debts ( ttl-priv-debts + S1-L2ip-debts )
  ;; xx set ttl-priv-debts ( ttl-priv-debts + c2-assets )

  set net-worth-priv ( ttl-priv-assets - ttl-priv-debts )

  ;; Money supply aggregates
  set msi-assets  0 ;; Physical money supply
  set msi-assets ( msi-assets + P0-vc-assets )
  set msi-assets ( msi-assets + P0-er-assets )
  set msi-assets ( msi-assets + P0-rr-assets )

  set msi-debts   0 ;; Physical money supply
  set msi-debts ( msi-debts + P0-rr-debts )
  set msi-debts ( msi-debts + P0-er-debts )

  set msii-assets 0 ;; Logical money supply
  set msii-assets ( msii-assets + L1-assets )
  set msii-assets ( msii-assets + L1-loan-assets )
  set msii-assets ( msii-assets + C1-assets )
  ;; xx set msii-assets ( msii-assets + c2-assets )

  set msii-debts  0 ;; Logical money supply
  set msii-debts ( msii-debts + L1-debts )
  set msii-debts ( msii-debts + L2-debts )

  set msiii-assets 0 ;; Shadow money supply
  set msiii-assets ( msiii-assets + S1-L1ir-assets )
  set msiii-assets ( msiii-assets + S1-rrir-assets )
  set msiii-assets ( msiii-assets + S1-erir-assets )

  set msiii-debts 0 ;; Shadow money supply
  set msiii-debts ( msiii-debts + S1-L2ip-debts )


;; end of f-compute-bank-net-worth
end

;;-----------------------------------------------------------------------------|
;; Compute the net worth of a prsn.
to f-compute-prsn-net-worth
;; This routine is to be executed a prsn.

  set ttl-P0-assets P0-assets

  set ttl-publ-assets      0
  set ttl-publ-debts       0
  set net-worth-publ       0

  set ttl-P0-assets        P0-assets

  set ttl-priv-assets 0
  set ttl-priv-assets ( ttl-priv-assets + L0-assets )
  set ttl-priv-assets ( ttl-priv-assets + L1-assets )
  set ttl-priv-assets ( ttl-priv-assets + S1-30day-total-assets )
  set ttl-priv-assets ( ttl-priv-assets + L2-assets )
  set ttl-priv-assets ( ttl-priv-assets + S1-L2ir-assets )
  ;; ss set ttl-priv-assets ( ttl-priv-assets + L3-assets )
  ;; ss set ttl-priv-assets ( ttl-priv-assets + S1-L3ir-assets )
  ;; ss set ttl-priv-assets ( ttl-priv-assets + L4-assets )
  ;; ss set ttl-priv-assets ( ttl-priv-assets + L4-dividend-receivable )

  set ttl-priv-debts 0
  set ttl-priv-debts ( ttl-priv-debts + L1-loan-debts )
  set ttl-priv-debts ( ttl-priv-debts + S1-L1ip-debts )
  set ttl-priv-debts ( ttl-priv-debts + S1-30day-total-debts )

  set net-worth-priv ( ttl-priv-assets - ttl-priv-debts )

  ;; Money supply aggregates
  set msi-assets  0 ;; Physical money supply
```

```
  set msi-assets ( msi-assets + P0-assets )

  set msi-debts   0 ;; Physical money supply

  set msii-assets 0 ;; Logical money supply
  set msii-assets ( msii-assets + L0-assets )
  set msii-assets ( msii-assets + L1-assets )
  set msii-assets ( msii-assets + L2-assets )
  ;; ss set msii-assets ( msii-assets + L3-assets )
  ;; ss set msii-assets ( msii-assets + L4-assets )

  set msii-debts  0 ;; Logical money supply
  set msii-debts ( msii-debts + L1-loan-debts )

  set msiii-assets 0 ;; Shadow money supply
  set msiii-assets ( msiii-assets + S1-30day-total-assets )
  set msiii-assets ( msiii-assets + S1-L2ir-assets )
  ;; ss set msiii-assets ( msiii-assets + S1-L3ir-assets )
  ;; ss set msiii-assets ( msiii-assets + L4-dividend-receivable )

  set msiii-debts 0 ;; Shadow money supply
  set msiii-debts ( msiii-debts + S1-30day-total-debts )
  ;; Somewhat arbitrarily I have decided that L1 loan debts will be
  ;;   considered shadow money.  This is so the only MS-II expansion
  ;;   will come from the principal of the loans themselves.
  set msiii-debts ( msiii-debts + S1-L1ip-debts )

;; end of f-compute-prsn-net-worth
end

;;-----------------------------------------------------------------------------|
;; Compute the net worth of a corp.
to f-compute-corp-net-worth
;; This routine is to be executed a corp.

  set ttl-publ-assets      0
  set ttl-publ-debts       0
  set net-worth-publ       0

  set ttl-P0-assets        P0-assets

  set ttl-priv-assets 0
  set ttl-priv-assets ( ttl-priv-assets + L0-assets )
  set ttl-priv-assets ( ttl-priv-assets + L1-assets )
  set ttl-priv-assets ( ttl-priv-assets + S1-30day-total-assets )
  set ttl-priv-assets ( ttl-priv-assets + L2-assets )
  set ttl-priv-assets ( ttl-priv-assets + S1-L2ir-assets )
  ;; ss set ttl-priv-assets ( ttl-priv-assets + L3-assets )
  ;; ss set ttl-priv-assets ( ttl-priv-assets + L4-assets )

  set ttl-priv-debts 0
  set ttl-priv-debts ( ttl-priv-debts + L1-loan-debts )
  set ttl-priv-debts ( ttl-priv-debts + S1-L1ip-debts )
  set ttl-priv-debts ( ttl-priv-debts + S1-30day-total-debts )
  ;; ss set ttl-priv-debts ( ttl-priv-debts + L3-debts )
  ;; ss set ttl-priv-debts ( ttl-priv-debts + S1-L3ip-debts )
  ;; ss set ttl-priv-debts ( ttl-priv-debts + L4-debts )
  ;; ss set ttl-priv-debts ( ttl-priv-debts + S1-L4dp-debts )

  set net-worth-priv ( ttl-priv-assets - ttl-priv-debts )

  ;; Money supply aggregates
  set msi-assets  0 ;; Physical money supply
  set msi-assets ( msi-assets + P0-assets )

  set msi-debts   0 ;; Physical money supply

  set msii-assets 0 ;; Logical money supply
  set msii-assets ( msii-assets + P0-assets )
  set msii-assets ( msii-assets + L1-assets )
  set msii-assets ( msii-assets + L2-assets )
  ;; ss set msii-assets ( msii-assets + L3-assets )
  ;; ss set msii-assets ( msii-assets + L4-assets )

  set msii-debts  0 ;; Logical money supply
  set msii-debts ( msii-debts + L1-loan-debts )
  ;; ss set msii-debts ( msii-debts + L3-debts )
  ;; ss set msii-debts ( msii-debts + L4-debts )

  set msiii-assets 0 ;; Shadow money supply
  set msiii-assets ( msiii-assets + S1-30day-total-assets )
  set msiii-assets ( msiii-assets + S1-L2ir-assets )

  set msiii-debts 0 ;; Shadow money supply
  set msiii-assets ( msiii-debts + S1-30day-total-debts )
  ;; ss set msiii-assets ( msiii-assets + S1-L3ip-debts )
  ;; ss set msiii-assets ( msiii-assets + S1-L4dp-debts )

;; end of f-compute-corp-net-worth
end

;;-----------------------------------------------------------------------------|
;; BANKING SERVICES
;;-----------------------------------------------------------------------------|
;; THEORY: In this section of the code all of the patterns for types of banking
;;   services have been pulled together in a single place.  This is to enable
;;   consistency in the means of implmenting each type of service, with
;;   the hope that it will make coding, debugging, and maintenance easier, at
;;   a possible cost of performance.
;; Note that it is intentional that none of these routine do range error
;;   checking on the variables affected.  So, for example, a prsn with no money
;;   in a savings account may still move money from there to a checking account.
;; The creation of negatives and their ultimate removal again all gets
;;   resolved in the daily visit to the bank by each agent.  Loans are usually
;;   available to cover net negatives, and, when they are not, bankruptcy
;;   routines sort it all out.
;; The real purpose of these routines is to defend the public trust that
;;   money is properly conserved unless explicitly indicated otherwise.
;; Rather that implementing the complicated issue of linking bank accounts
;;   directly to clients, the clients keep track of the details of their own
;;   accounts, and the banks only keep track of aggregate amounts.  This
;;   simplifies the coding dramatically, and so reduces the chances of coding
;;   error, but it puts the onus on the clients to have their books in order.
;;   These banking routines look after that.

;;-----------------------------------------------------------------------------|
;; A prsn has cash (P0, L0) and deposits into its bank.
to f-bsvcs-prsn-deposits-cash [ amount-to-deposit ]
;; This routine is to be executed a prsn.

  ;; TODO: this routine may work for corps as well.

  ;; Contact the bank.
  let my-bank ( bank bank-who )

  ;; Remove cash from prsn's wallet.
  set L0-assets ( L0-assets - amount-to-deposit )
  set P0-assets ( P0-assets - amount-to-deposit )

  ;; Put the cash into the bank's books (L0) and vault (P0).
  ask my-bank
  [
    set L1-assets ( L1-assets + amount-to-deposit )
    set P0-vc-assets ( P0-vc-assets + amount-to-deposit )
  ]

  ;; Now, adjust the bank's aggregate checking account to reflect
  ;;   the increase in the checkable deposits.
  ask my-bank [ set L1-debts ( L1-debts + amount-to-deposit ) ]
  ;; Finally, adjust the prsn's bankbook to indicate the amount of checkable
  ;;   money available to this prsn, and also to lay a claim on a portion
  ;;   of the aggregate of checkable money in the bank.
  set L1-assets ( L1-assets + amount-to-deposit )

  LOG-TO-FILE ( word "  BSvcs: Amount of P0 deposited - " amount-to-deposit )

;; end of f-bsvcs-prsn-deposits-cash
end

;;-----------------------------------------------------------------------------|
;; A prsn has checkable funds in the bank and withdraws cash (P0, L0).
to f-bsvcs-prsn-withdraws-cash [ amount-to-withdraw ]
;; This routine is to be executed a prsn.

  ;; TODO: this routine may work for corps as well.

  ;; Contact the bank.
  let my-bank ( bank bank-who )

  ;; This is the reversal of a deposit.
  ;; Put cash into prsn's wallet.
  set L0-assets ( L0-assets + amount-to-withdraw )
  set P0-assets ( P0-assets + amount-to-withdraw )

  ;; Get the cash from the bank's books (L0) and vault (P0).
  ask my-bank
  [
    set L1-assets ( L1-assets - amount-to-withdraw )
    set P0-vc-assets ( P0-vc-assets - amount-to-withdraw )
  ]

  ;; Now, adjust the bank's aggregate checking account to reflect
  ;;   the decrease in the checkable deposits.
  ask my-bank [ set L1-debts ( L1-debts - amount-to-withdraw ) ]
  ;; Finally, adjust the prsn's bankbook to indicate the amount of checkable
  ;;   money no longer available to this prsn, and also to release the
  ;;   claim on a portion of the aggregate of checkable money in the bank.
  set L1-assets ( L1-assets - amount-to-withdraw )

  LOG-TO-FILE ( word "  BSvcs: Amount of P0 withdrawn - " amount-to-withdraw )

;; end of f-bsvcs-prsn-withdraws-cash
end

;;-----------------------------------------------------------------------------|
;; A prsn moves money from a checking acct (L1) to a savings acct (L2).
```

```
to f-bsvcs-prsn-moves-L1-to-L2 [ amount-to-move ]
;; This routine is to be executed a prsn.

  ;; TODO: this routine may work for corps as well.

  ;; Contact the bank.
  let my-bank ( bank bank-who )

  ask my-bank
  [
    ;; The bank decreases the aggregator for checkable funds.
    set L1-debts ( L1-debts - amount-to-move )
    ;; The bank increases the aggregator for savings funds.
    set L2-debts ( L2-debts + amount-to-move )
  ]

  ;; The prsn decreases its claim on checkable funds, in its check book.
  set L1-assets ( L1-assets - amount-to-move )
  ;; The prsn increases its claim on savings, in its savings book.
  set L2-assets ( L2-assets + amount-to-move )

  LOG-TO-FILE ( word "  BSvcs: Moved from L1 to L2 ---- " amount-to-move )

;; end of f-bsvcs-prsn-moves-L1-to-L2
end

;;-------------------------------------------------------------------------------|
;; A prsn moves money from a savings acct (L2) to a checking acct (L1).
to f-bsvcs-prsn-moves-L2-to-L1 [ amount-to-move ]
;; This routine is to be executed a prsn.

  ;; TODO: this routine may work for corps as well.

  ;; Contact the bank.
  let my-bank ( bank bank-who )

  ;; This is the reversal of a move of L1 to L2.
  ask my-bank
  [
    ;; The bank increases the aggregator for checkable funds.
    set L1-debts ( L1-debts + amount-to-move )
    ;; The bank decreases the aggregator for savings funds.
    set L2-debts ( L2-debts - amount-to-move )
  ]

  ;; The prsn increases its claim on checkable funds, in its check book.
  set L1-assets ( L1-assets + amount-to-move )
  ;; The prsn decreases its claim on savings, in its savings book.
  set L2-assets ( L2-assets - amount-to-move )

  LOG-TO-FILE ( word "  BSvcs: Moved from L2 to L1 ---- " amount-to-move )

;; end of f-bsvcs-prsn-moves-L2-to-L1
end

;;-------------------------------------------------------------------------------|
;; A bank is checked to determine if it is bankrupt.
to f-bsvcs-bank-checked-for-bankruptcy
  ;; This routine is to be executed by the observer.

  ;; Determine whether the bank is, itself, bankrupt.
  ask banks
  [
    ;; THEORY: If the bank has no means of earning money, it must trust to
    ;;   luck to have its clients deposit more vault cash, which could
    ;;   then be deposited in the CRB to earn interest for its C1-assets.
    ;;   But there will be a steady drain on its C1-assets as its clients
    ;;   go bankrupt for lack of L1-loans.  So this bank is doomed.

    ;; Ensure the net worth data is up-to-date.
    f-compute-bank-net-worth

    ;; Assume bankrupt as the default, then switch it back if there is
    ;;   some potential to earn interest.
    set b-bank-is-bankrupt 1 ;; The default assumption.
    if( P0-all-assets > g-minimum-vault-cash )
      [ set b-bank-is-bankrupt 0 ]  ;; Can earn money on ER and RR.
    if( L1-loan-assets > 0 )
      [ set b-bank-is-bankrupt 0 ]  ;; Can earn money on L1 loans.
  ]

;; end of f-bsvcs-bank-checked-for-bankruptcy
end

;;-------------------------------------------------------------------------------|
;; A prsn negotiates to take out a bank loan.
to f-bsvcs-prsn-negotiates-an-L1-loan
  ;; This routine is to be executed by a prsn.

  ;; Contact the bank.
  let my-bank ( bank bank-who )

  ;; Loans are given only if savings account is negative.
  ;; This means the agent had insufficient funds to address daily needs for
  ;;   L0 and L1 types of funds.  I.e. all assets have been moved to checking
  ;;   or cash for daily use.
  ifelse( L2-assets < 0 )
  [
    ;; This agent needs to take out a loan.
    LOG-TO-FILE ( word "  Prsn " who " requires a bank loan." )

    ;; Is the bank elligible to provide more loans?
    let bank-loan-flag ( [b-bank-can-make-loans] of my-bank )
    ;; The bank may not have any remaining excess reserves to support a loan.
    ifelse( bank-loan-flag = 0 )
    [
      ;; Case of bank cannot make loans.
      ;; Mark the prsn as bankrupt.
      set b-prsn-is-bankrupt 1
      LOG-TO-FILE ( word "  Bank " bank-who " cannot provide loan." )
      LOG-TO-FILE ( word "  Prsn " who " is now bankrupt." )
    ]
    ;; else
    [
      ;; Case of the prsn needs a loan and the bank can offer one.
      ;; Is the prsn elligible to receive a loan.

      ifelse( L1-loan-debts < ( g-bankruptcy-factor * g-p-standard-loan ) )
      [
        ;; The loan is approved!
        set g-counts-loans ( g-counts-loans + 1 )

        ;;  NOTE: a loan requires four entries - two offsetting double-entries
        ;;    such that the net worth of neither participant changes.
        ;;
        ;; The amount of the loan will be sufficient for two months
        ;;   of daily living.
        LOG-TO-FILE ( word "  Prsn L2-assets ---------- " L2-assets  )
        LOG-TO-FILE ( word "  Prsn L1-assets ---------- " L1-assets  )
        LOG-TO-FILE ( word "  Prsn L1-loan-debts ------ " L1-loan-debts  )
        let amount-of-loan g-p-standard-loan

        ask my-bank
        [
          LOG-TO-FILE ( word "  Bank L1-assets ---------- " L1-assets  )
          LOG-TO-FILE ( word "  Bank L1-loan-assets ----- " L1-loan-assets  )
          LOG-TO-FILE ( word "  Bank L1-debts ----------- " L1-debts  )
          ;; Register the loan as a bank asset.  Entry #1 of 4.
          set L1-loan-assets ( L1-loan-assets + amount-of-loan )
          LOG-TO-FILE ( word "  Amount of loan ---------- " amount-of-loan  )
          ;; Put money into the prsn's loan-related checking account.
          ;;   Entry #2 of 4.
          set L1-debts ( L1-debts + amount-of-loan )
          LOG-TO-FILE ( word "  Bank L1-assets ---------- " L1-assets  )
          LOG-TO-FILE ( word "  Bank L1-loan-assets ----- " L1-loan-assets  )
          LOG-TO-FILE ( word "  Bank L1-debts ----------- " L1-debts  )
        ]
        ;; Prsn records the loan in his checkbook.  Entry #3 of 4.
        set L1-assets ( L1-assets + amount-of-loan )  ;; Good as is.
        ;; Prsn files the loan agreement.  Entry #4 of 4.
        set L1-loan-debts ( L1-loan-debts + amount-of-loan )
        LOG-TO-FILE ( word "  Prsn L1-assets ---------- " L1-assets  )
        LOG-TO-FILE ( word "  Prsn L1-loan-debts ------ " L1-loan-debts  )
      ]  ;; end of ifelse( L1-loan-debts > ( 2 * g-p-standard-loan ) )
      ;; Else prsn is inellible.
      [
        ;; Case of prsn is inelligible.
        ;; Mark the prsn as bankrupt.
        set b-prsn-is-bankrupt 1
        LOG-TO-FILE ( word "  Prsn " who " is inelligible due to debt." )
        LOG-TO-FILE ( word "  Prsn L1-loan-debts ------ " L1-loan-debts  )
        LOG-TO-FILE ( word "  Prsn " who " is now bankrupt." )
      ]  ;; end of case of prsn is inelligible.
    ] ;; end of Bank can make loans.
  ]  ;; end prsn needs a loan.
  ;; Else
  [
    LOG-TO-FILE ( word "  A loan is not required!" )
  ]

;; End of f-bsvcs-prsn-negotiates-an-L1-loan
end

;;-----------------------------------------------------------------------------|
;; A client takes out a loan and places the money in its checkable (L1) account.
to f-bsvcs-client-takes-out-L1-loan [ amount-to-borrow ]
;; This routine is to be executed a prsn, a corp, or the GCRA.

  ;; This version is not used.  See f-bsvcs-prsn-negotiates-an-L1-loan.

  ;; The client and the bank sign a loan agreement in duplicate, and the funds
  ;;   are deposited into the client's checkable (L1) account.  This requires
  ;;   four entries - two of which are segregated in L1-loan variables.

  ;; Contact the bank.
  let the-bank ( bank bank-who )

  ;; The loan is signed in duplicate, and the size recorded by both parties.
  ;; First, the bank registers the loan in an aggregator.  Entry #1.
  ask the-bank [ set L1-loan-assets ( L1-loan-assets + amount-to-borrow ) ]
  ;; Then the client stores the copy of the loan in their own records.
  ;;   Entry #2.
  set L1-loan-debts ( L1-loan-debts + amount-to-borrow )

  ;; Now, the bank makes checkable money available to its client.  Entry #3.
  ask the-bank [ set L1-debts ( L1-debts + amount-to-borrow ) ]
  ;; And the client records the claim to the money in its own check book.
  ;;   Entry #4.
  set L1-assets ( L1-assets + amount-to-borrow )

  LOG-TO-FILE ( word "  BSvcs: L1 loan taken ---------- " amount-to-borrow )

  ;; As a result of this, the bank will need to move some of its reserves
  ;;   from excess reserves to required reserves.  This is handled when the
  ;;   bank and CRB reconcile their books daily.

;; end of f-bsvcs-client-takes-out-L1-loan
end

;;-----------------------------------------------------------------------------|
;; A client makes a payment on an L1 loan from its checkable (L1) account.
to f-bsvcs-client-makes-L1-loan-payment [ amount-to-pay ]
;; This routine is to be executed a prsn, a corp or the GCRA.

  ;; Contact the bank.
  let the-bank ( bank bank-who )

  ;; This is a partial reversal of the routine to take out a loan.
  ;; First, the bank decreases the size of the loan in its aggregator.
  ask the-bank [ set L1-loan-assets ( L1-loan-assets - amount-to-pay ) ]
  ;; Then the client decreases the size of the loan in its own records.
  set L1-loan-debts ( L1-loan-debts - amount-to-pay )

  ;; Now, the bank reduces the checkable money available to its clients.
  ask the-bank [ set L1-debts ( L1-debts - amount-to-pay ) ]
  ;; And the client reduces its claim to the money in its own check book.
  set L1-assets ( L1-assets - amount-to-pay )

  LOG-TO-FILE ( word "  BSvcs: L1 loan paid ----------- " amount-to-pay )

;; end of f-bsvcs-client-makes-L1-loan-payment
end

;;-----------------------------------------------------------------------------|
;; A client is charged daily interest on outstanding amount of L1 loan(s).
to f-bsvcs-client-accrues-daily-interest-on-L1-loan
;; This routine is to be executed a prsn, a corp or the GCRA.

  ;; THEORY: -ptbfs- This causes a flow of money from the real
  ;;   economy to the banking sector because the interest on L1 bank
  ;;   loans is paid by Prsns directly to the Banks.  As such, it is part
  ;;   of the "Prsns to Banks Flows" (ptbfs).  It can be turned off
  ;;   by setting g-iobl to zero.

  if( g-iobl > 0 )
  [
    ;; THEORY: Interest on L1 loans is to be paid by the prsn to the bank.
    ;;   The size of the loan may vary due to new amounts taken out or payments
    ;;   made, so interest is charged and accrued on a daily basis, but only
    ;;   paid on a monthly basis.  This interest is a debt which expands the
    ;;   shadow money supply, as it is basically a loan from the bank to the
    ;;   prsn until it is paid.  There is a hair to be split, here, and I am
    ;;   splitting it this way.  Because this debt is visible to the banks,
    ;;   and really amounts to a bank loan, it should be considered part of the
    ;;   logical money supply (L1) instead of the shadow money supply (S1).
    ;;   But, because I want to focus on L1 loan tracking in this application,
    ;;   I have chosen, somewhat arbitrarily, to include it in S1 until it
    ;;   is paid.

    ;; Contact the bank.
    let the-bank ( bank bank-who )

    ;; The bank only has an aggregate variable for all of the L1 loans of all
    ;;   of its clients.  Only the client's record indicates the size of the
    ;;   loan associated with this client.
    let loan-size L1-loan-debts
    ;; The annual interest on bank loans is in slider g-iobl.
    let annual-interest-due ( loan-size * g-iobl / 100 )
    ;; Prorate this to a daily rate (12 months; 30 days per month).
    let daily-interest-due ( annual-interest-due / ( 12 * 30 ) )

    ;; The bank records the increase in its S1 aggregator for
    ;;   L1 loan interest receivable.
    ask the-bank [ set S1-L1ir-assets ( S1-L1ir-assets + daily-interest-due ) ]
    ;; The client records the increase in its S1 record for interest payable.
    set S1-L1ip-debts ( S1-L1ip-debts + daily-interest-due )

    LOG-TO-FILE ( word "  BSvcs: L1 interest accrued ---- " daily-interest-due )
  ]

;; end of f-bsvcs-client-accrues-daily-interest-on-L1-loan
end

;;-----------------------------------------------------------------------------|
;; A client pays outstanding interest on L1 loan(s) monthly.
to f-bsvcs-client-pays-monthly-interest-on-L1-loan
;; This routine is to be executed a prsn, a corp or the GCRA.

  ;; THEORY: Interest on L1 loans is to be paid by the prsn to the bank.
  ;;   It accrues daily, but is paid in aggregate monthly.
  ;; When interest is accrued, it is stored with 16 (or so) digits after
  ;;   the decimal, but it is paid in dollar units.  I don't want to round
  ;;   away all of the accuracy of the interest payments, since I accrue
  ;;   it daily.  So, I determine the floor of the amount due, pay that,
  ;;   and leave a residual amount to be paid the next month.  By doing it
  ;;   this way, the shadow money supply holds the (not-absolutely precise)
  ;;   fractional debts, but the logical money supply is always accurate
  ;;   with infinite precision to the dollar.
  ;; This may affect the way I compare total interest payments, over time,
  ;;   with total write-offs, over time, but I don't think it will.
  ;; TODO:  I need to watch that.
  ;; Interest collected by the bank represents a change in its corporate
  ;;   net worth.  This income is outside of its role as the guardian of
  ;;   the rule of conservation of money, its public trust, and so must be
  ;;   put into its own corporate checking account (a C1 account) as if
  ;;   it is a client of itself.
  ;; So this payment is a peculiar client-to-client payment mediated by
  ;;   the bank's back room that manages the public trust.  This payment
  ;;   requires a total of six accounting entries, two of which counter-act
  ;;   each other and are suppressed.

  ;; Contact the bank.
  let the-bank ( bank bank-who )


  ;; The bank only has an aggregate variable for all of the interest payable
  ;;   on all loans to its clients.  Only the client's records indicate the
  ;;   size of the accrued interest associated with this client.
  ;; Determine the largest integral dollar amount payable.
  let monthly-interest-paid floor( S1-L1ip-debts )

  ;; Settle the records for the shadow money supply first.
  ;; The client notes the payment, subtracting it from dues accrued,
  ;;   and leaving a residual.
  set S1-L1ip-debts ( S1-L1ip-debts - monthly-interest-paid )
  ;; The bank decreases its aggregator by the same amount.
  ask the-bank [ set S1-L1ir-assets ( S1-L1ir-assets - monthly-interest-paid ) ]

  ;; Now, the client has to actually pay the bill with real money.
  ;; The payment is noted in the client's check book.
  set L1-assets ( L1-assets - monthly-interest-paid )
  ask the-bank
  [
    ;; The front-room corporate comptroller notes the payment in its check book.
    set C1-assets ( C1-assets + monthly-interest-paid )

    ;; The bank's back-room staff who manage the public trust note the payment.
    ;; Two entries are required to note the decreased liability for one client
    ;;   and the increased liability for the other client.  These all happen in
    ;;   an aggregator that is used to track all clients.  So, they cancel each
    ;;   other out, and are suppressed for performance purposes.
    ;; set L1-debts ( L1-debts - monthly-interest-paid )
    ;; set L1-debts ( L1-debts + monthly-interest-paid )
  ]

  LOG-TO-FILE ( word "  BSvcs: L1 interest paid ------- " monthly-interest-paid )

;; end of f-bsvcs-client-pays-monthly-interest-on-L1-loan
end


;;-----------------------------------------------------------------------------|
;; A bank is charged daily interest on outstanding amounts of L2 savings.
to f-bsvcs-client-accrues-daily-interest-on-L2-savings
;; This routine is to be executed a prsn, a corp or the GCRA.

  if( g-iosd > 0 )
  [
    ;; THEORY: Interest on L2 savings is to be paid by the bank to the client.
    ;;   The size of the savings may vary daily due to commercial activity,
    ;;   so interest is charged and accrued on a daily basis, but only
    ;;   paid on a monthly basis.  This interest is a debt which expands the
    ;;   shadow money supply, as it is basically a loan from the client to the
    ;;   bank until it is paid.
    ;;
    ;; The same as for L1 loans, there is a hair to be split, here, and I am
    ;;   splitting it this way.  Because this debt is visible to the banks,
    ;;   and really amounts to a reverse bank loan, it should be considered
    ;;   part of the logical money supply (L1) instead of the shadow money
    ;;   supply (S1).
    ;; But, because I want to focus on L1 loan tracking in this application, I have
    ;;   chosen, somewhat arbitrarily, to include it in S1 until it is paid.
    ;; Contact the bank.
    let the-bank ( bank bank-who )

    ;; The bank only has an aggregate variable for all of the savings of all
    ;;   of its clients.  Only the client's records indicate the size of the
    ;;   savings deposit associated with this client.
    let savings-account-size L2-assets
    ;; The annual interest on bank deposits is in slider g-iosd.
    let annual-interest-due ( savings-account-size * g-iosd / 100 )
    ;; Prorate this to a daily rate (12 months; 30 days per month).
    let daily-interest-due ( annual-interest-due / ( 12 * 30 ) )

    ;; The bank records the increase in its S1 aggregator for
    ;;   savings (L2) interest payable.
    ask the-bank [ set S1-L2ip-debts ( S1-L2ip-debts + daily-interest-due ) ]
    ;; The client records the increase in its S1 record for interest receivable.
    set S1-L2ir-assets ( S1-L2ir-assets + daily-interest-due )

    LOG-TO-FILE ( word "  BSvcs: L2 interest accrued ---- " daily-interest-due )
  ]

;; end of f-bsvcs-client-accrues-daily-interest-on-L2-savings
end

;;-----------------------------------------------------------------------------|
;; A client pays outstanding interest on savings deposits monthly.
to f-bsvcs-client-paid-monthly-interest-on-L2-savings
;; This routine is to be executed a prsn, a corp or the GCRA.

  ;; THEORY: Interest on L2 savings is to be paid by the bank to the client.
  ;;   It accrues daily, but is paid in aggregate monthly.
  ;; When interest is accrued, it is stored with 17 (or so) digits after
  ;;   the decimal, but it is paid in dollar units.  I don't want to round
  ;;   away all of the accuracy of the interest payments, since I accrue
  ;;   it daily.  So, I determine the floor of the amount due, pay that,
  ;;   and leave a residual amount to be paid the next month.  By doing it
  ;;   this way, the shadow money supply holds the (not-absolutely precise)
  ;;   fractional debts, but the logical money supply is always accurate
  ;;   with infinite precision to the dollar.
  ;; This may affect the way I compare total interest payments, over time,
  ;;   with total write-offs, over time, but I don't think it will.
  ;; TODO:  I need to watch that.
  ;; Interest paid by the bank represents a change in its corporate
  ;;   net worth.  This expense is outside of its role as the guardian of
  ;;   the rule of conservation of money, its public trust, and so must be
  ;;   put into its own corporate checking account (a C1 account) as if
  ;;   it is a client of itself.
  ;; So this payment is a peculiar client-to-client payment mediated by
  ;;   the bank's back room that manages the public trust.  This payment
  ;;   requires a total of six accounting entries, two of which counter-act
  ;;   each other and are suppressed.

  ;; Contact the bank.
  let the-bank ( bank bank-who )

  ;; The bank only has an aggregate variable for all of the interest payable
  ;;   on all savings deposits of its clients.  Only the client's records
  ;;   indicate the size of the accrued interest associated with this client.
  ;; Determine the largest integral dollar amount payable.
  let monthly-interest-paid floor( S1-L2ir-assets )

  ;; Settle the records for the shadow money supply first.
  ;; The client notes the payment, subtracting it from dues accrued,
  ;;   and leaving a residual.
  set S1-L2ir-assets ( S1-L2ir-assets - monthly-interest-paid )
  ;; The bank decreases its aggregator by the same amount.
  ask the-bank [ set S1-L2ip-debts ( S1-L2ip-debts - monthly-interest-paid ) ]

  ;; Now, the bank has to actually pay the bill with real money.
  ;; The payment is noted in the client's check book.
  set L1-assets ( L1-assets + monthly-interest-paid )
  ask the-bank
  [
    ;; The front-room corporate comptroller notes the payment in its check book.
    set C1-assets ( C1-assets - monthly-interest-paid )

    ;; The bank's back-room staff who manage the public trust note the payment.
    ;; Two entries are required to note the decreased liability for one client
    ;;   and the increased liability for the other client.  These all happen in
    ;;   an aggregator that is used to track all clients.  So, they cancel each
    ;;   other out, and are suppressed for performance purposes.
    ;; set L1-debts ( L1-debts - monthly-interest-paid )
    ;; set L1-debts ( L1-debts + monthly-interest-paid )
  ]

  LOG-TO-FILE ( word "  BSvcs: L2 interest received --- " monthly-interest-paid )

;; end of f-bsvcs-client-paid-monthly-interest-on-L2-savings
end

;;-----------------------------------------------------------------------------|
;; A prsn pays another prsn for something.  This is a capital exchange.
to f-bsvcs-prsn1-pays-prsn2-by-cash [ prsn2who amount-to-pay ]
;; This routine is to be executed a prsn.

  ;; THEORY:  This is the most simple capital exchange possible, in the
  ;;   real world, but has its minor complications in this program due to
  ;;   the separation of physical and logical money.  The exchange requires
  ;;   four entries.
  ;; Due to the fact that this model does not pay any regard to the goods
  ;;   and services exchanged in reciprocity for the cash exchanged, the
  ;;   money is simply moved from prsn to prsn.  Because this is a cash
  ;;   only transaction, no bank is involved.  As such, the bank has no
  ;;   real visibility into this volume of economic activity, and so it is
  ;;   in some sense part of the shadow economy, but it definitely affects
  ;;   only the physical and logical money, and not shadow money.

  ;; TODO: this routine may also work for corps, as long as the recipient
  ;;   is a prsn.

  ;; Contact prsn2.
  let prsn2 ( prsn prsn2who )

  ;; prsn1 takes the cash out of its wallet.
  set P0-assets ( P0-assets - amount-to-pay )
  set L0-assets ( L0-assets - amount-to-pay )

  ;; prsn2 puts the cash into its wallet.
  ask prsn2
  [
    set P0-assets ( P0-assets + amount-to-pay )
    set L0-assets ( L0-assets + amount-to-pay )
  ]
  LOG-TO-FILE ( word "  BSvcs: Prsn " who " paid Prsn "
    prsn2who " ----------- " amount-to-pay )

;; end of f-bsvcs-prsn1-pays-prsn2-by-cash
end

;;-----------------------------------------------------------------------------|
;; A prsn pays another prsn for something.  This is a capital exchange.
to f-bsvcs-prsn1-pays-prsn2-by-check [ prsn2who amount-to-pay ]
;; This routine is to be executed a prsn.

  ;; THEORY:  This is a simple capital exchange using a check.  In this
  ;;   program due to the involvement of two banks there are some minor
  ;;   wrinkles to be managed.  The exchange requires four entries if it
  ;;   is within a single bank, but six for bank-to-bank exchange.  Only
  ;;   the net worth of the prsns change.
  ;; Due to the fact that this model does not pay any regard to the goods
  ;;   and services exchanged in reciprocity for the cash exchanged, the
  ;;   money is simply moved from prsn to prsn.  Because this is a check
  ;;   only transaction, two banks are involved.  As such, the bank has
  ;;   real visibility into this volume of economic activity and functions
  ;;   entirely within the logical money supply.

  ;; TODO: this routine may also work for corps, as long as the recipient
  ;;   is a prsn.

  ;; Contact my bank.
  let my-bank ( bank bank-who )
  ;; Contact prsn2.
  let prsn2 ( prsn prsn2who )
  ;; Contact bank of prsn2.
  let prsn2-bank ( bank ( [bank-who] of prsn2 ) )

  ;; THEORY: A payment by check requires three double-entry actions, or
  ;;   six entries in total:
  ;;   -- The check books of the two parties in the transactions need to
  ;;      be changed to reflect the transfer of money.  I.e. their L1-assets
  ;;      variables need to be altered.  This changes the net worth of each
  ;;      party to the transaction, which is as expected.
  ;;   -- To match the transfer, the L1-debts variables of the associated
  ;;      banks need to be altered.  But this changes the net worth of the
  ;;      back room of each chartered bank, which is not good.  The assets
  ;;      of each bank need to be altered to match the liabilities of each
  ;;      bank.
  ;;   -- To balance the books within each bank (back room) the L1-assets
  ;;      variables must also be adjusted.  In effect, one bank transfers its
  ;;      obligations to the other bank.
  ;; If both prsns use the same bank, since the L1-assets and L1-debts variables
  ;;      are aggregators for all clients of the bank, the above four actions
  ;;      counter-act each other.  So this works whether the prsns are
  ;;      clients of the same or different banks.


  ;; prsn1 writes the check, recording it in its check book.
  set L1-assets ( L1-assets - amount-to-pay )
  ;; prsn2 accepts the check and indicates an L1 deposit in its check book.
  ask prsn2 [ set L1-assets ( L1-assets + amount-to-pay ) ]

  ;; Now the back rooms of the two banks reconcile their books.
  ask my-bank [ set L1-assets ( L1-assets - amount-to-pay ) ]
  ask my-bank [ set L1-debts ( L1-debts - amount-to-pay ) ]
```

```
  ask prsn2-bank [ set L1-assets ( L1-assets + amount-to-pay ) ]
  ask prsn2-bank [ set L1-debts ( L1-debts + amount-to-pay ) ]

  LOG-TO-FILE ( word "  BSvcs: Prsn " who " paid Prsn "
    prsn2who " --- " amount-to-pay )

;; end of f-bsvcs-prsn1-pays-prsn2-by-check
end

;;-----------------------------------------------------------------------------|
;; BANKING SERVICES
;;-----------------------------------------------------------------------------|
;; All of the routines that perform banking services start with f-cbsvcs-xxx or
;;   or f-bsvcs-xxx or f-bnkrpt-xxx.  They address the activities of the
;;   central reserve bank (the CRB), the chartered banks (front and back room
;;   activities), and all bankruptcy processing.
;; The routines are all gathered here to enable consistency and easy scrutiny.
;;

;; START OF -BSVCS- SUBSECTION.

;;-----------------------------------------------------------------------------|
;; The Gov't finds a suitable bank to do business.
to f-bsvcs-gcra-find-bank
  ;; This routine is to be executed by a GCRA.
  ;; This GCRA does not yet have a bank assigned.

  ;; Does this GCRA already have a bank?
  ifelse( bank-who = -1 )
  [
    ;; It does not have a bank.
    ;; Establish a list of potential banks.
    ;; Potential bank must need clients.
    ;; A dummy let statement.
    let bank-list []
    ;; Bank must need GCRA clients.
    set bank-list ( banks with
      [ ( no-of-gcra-clients < 1 ) ] )

    if( any? bank-list )
    [
      let this-bank one-of bank-list
      ;; The search is successful.
      set bank-who ( [who] of this-bank )
      ask this-bank [ set no-of-gcra-clients ( no-of-gcra-clients + 1 ) ]
      LOG-TO-FILE ( word "  Found - " this-bank )
    ]
  ]  ;; End of if( bank-who = -1 )
  ;; Else
  [
    LOG-TO-FILE ( word "  Bank not needed!  Not searching." )
  ]
  ;; End Else
;; End of f-bsvcs-gcra-find-bank
end

;;-----------------------------------------------------------------------------|
;; The CRB finds a suitable chartered bank for its C1 account.
to f-bsvcs-crb-find-bank
  ;; This routine is to be executed by a CRB.
  ;; This CRB does not yet have a bank assigned.
  ;; Does this CRB already have a bank?
  ifelse( bank-who = -1 )
  [
    ;; It does not have a bank.
    ;; Establish a list of potential banks.
    ;; Potential bank must need clients.
    ;; A dummy let statement.
    let bank-list []
    ;; Bank must need CRB clients.
    set bank-list ( banks with
      [ ( no-of-crb-clients < 1 ) ] )

    if( any? bank-list )
    [
      let this-bank one-of bank-list
      ;; The search is successful.
      set bank-who ( [who] of this-bank )
      ask this-bank [ set no-of-crb-clients ( no-of-crb-clients + 1 ) ]
      LOG-TO-FILE ( word "  Found - " this-bank )
    ]
  ]  ;; End of if( bank-who = -1 )
  ;; Else
  [
    LOG-TO-FILE ( word "  Bank not needed!  Not searching." )
  ]
  ;; End Else
;; End of f-bsvcs-crb-find-bank
end

;;-----------------------------------------------------------------------------|
;; Prsns find a suitable bank to do business.
to f-bsvcs-prsn-find-bank
  ;; This routine is to be executed by a prsn.
  ;; This prsn may have a bank already assigned.  Then a new one is assigned.

  LOG-TO-FILE ( word "Prsn " who " finding a bank." )
  ;; Establish a list of potential banks.
  ;; Potential bank must need clients.
  ;; A dummy let statement.
  let bank-list []
  ;; Bank should have available P0-ER-assets.
  set bank-list ( banks with [P0-ER-assets > 0] )

  ifelse( any? bank-list )
  [
    let this-bank one-of bank-list
    ;; The search is successful.
    set bank-who ( [who] of this-bank )
    ask this-bank [ set no-of-prsn-clients ( no-of-prsn-clients + 1 ) ]
    LOG-TO-FILE ( word "  Found - " this-bank )
  ]
  ;; else none have ER available.
  [
    ;; Choose any bank.
    let this-bank one-of banks
    set bank-who ( [who] of this-bank )
    ask this-bank [ set no-of-prsn-clients ( no-of-prsn-clients + 1 ) ]
  ]

  ;; End of f-bsvcs-prsn-find-bank
end
```

```
;;-----------------------------------------------------------------------------|
;; Corps find a suitable bank to do business.
to f-bsvcs-corp-find-bank
  ;; This routine is to be executed by a corp.
  ;; This corp does not yet have a bank assigned.

  ;; Does this corp already have a bank?
  ifelse( bank-who = -1 )
  [
    ;; It does not have a bank.
    ;; Establish a list of potential banks.
    ;; Potential bank must need clients.
    ;; A dummy let statement.
    let bank-list []
    ;; Bank must need corp clients.
    set bank-list ( banks with
      [ ( no-of-corp-clients < g-no-of-corps-per-bank ) ] )

    if( any? bank-list )
    [
      let this-bank one-of bank-list
      ;; The search is successful.
      set bank-who ( [who] of this-bank )
      ask this-bank [ set no-of-corp-clients ( no-of-corp-clients + 1 ) ]
      LOG-TO-FILE ( word "  Found - " this-bank )
    ]
  ]  ;; End of if( bank-who = -1 )
  ;; Else
  [
    LOG-TO-FILE ( word "  Bank not needed!  Not searching." )
  ]
  ;; End Else

  ;; End of f-bsvcs-corp-find-bank
end

;;-----------------------------------------------------------------------------|
;; Any of GCRA, prsn or corp makes a payment on a loan.
to f-bsvcs-agent-makes-a-payment-on-loan
;; This routine is to be executed by a GCRA, prsn or corp.
  ;; Pre-requisite: L1-assets exist, and L1-loan-debts > 0.

  ASSERT ( L1-loan-debts > 0 ) ( "Improper debts." ) who

  LOG-TO-FILE ( word "  Borrower L1 assets --------------- " L1-assets )
  LOG-TO-FILE ( word "  Borrower L1 loan debts ----------- " L1-loan-debts )

  ;; Determine the payment size.
  ;; Pay the least of standard payment, or remaining principal.
  let amount-to-pay g-p-standard-loan-payment
  if( amount-to-pay > L1-loan-debts )
  [
    set amount-to-pay L1-loan-debts
  ]

  ;; Contact the bank.
  let mybank ( bank bank-who )
  ask mybank
  [
    LOG-TO-FILE ( word "  Bank L1 loan assets ------------ " L1-loan-assets )
    LOG-TO-FILE ( word "  Bank L1 debts ------------------ " L1-debts )
    LOG-TO-FILE ( word "  Loan payment ------------------- " amount-to-pay )
    set L1-loan-assets ( L1-loan-assets - amount-to-pay )
    set L1-debts  ( L1-debts  - amount-to-pay )
    LOG-TO-FILE ( word "  Bank L1 loan assets ------------ " L1-loan-assets )
    LOG-TO-FILE ( word "  Bank L1 debts ------------------ " L1-debts )
  ]
  ;; Note the payment in the agent's checkbook.
  set L1-assets ( L1-assets - amount-to-pay )
  ;; Note that the principal on the loan has been reduced.
  set L1-loan-debts ( L1-loan-debts - amount-to-pay )

  LOG-TO-FILE ( word "Borrower L1 assets ----------------- " L1-assets )
  LOG-TO-FILE ( word "Borrower L1 loan debts ------------- " L1-loan-debts )

;; end of f-bsvcs-agent-makes-a-payment-on-loan
end

;;----------------------------------------------------------------------------|
;; Process a prsn that is bankrupt.
to f-bsvcs-process-prsn-bankruptcy
;; This routine is to be executed by a prsn.

  ;; TODO: After debugging, suppress this.
  ;; f-force-debug-output-on
  ;; TODO: Remove this if annoying.
  ;; beep

  ;; PART A - I need to collapse the assets and declare bankruptcy.
  ;; Prsns are bankrupt when they have insufficient funds to get through
  ;;   a standard day, their savings are <= zero and they are unable
  ;;   to take a loan because their bank does not have any excess reserves.
  ;; When they last attempted to get a loan, the bank would have marked a
  ;;   failed loan request as a bankruptcy.
  ;; So, I need to collapse the assets and debts of this prsn, pay off
  ;;   the loan as well as possible, and effect bankruptcy.

  ASSERT ( b-prsn-is-bankrupt = 1 ) "Prsn not bankrupt" who

  ;; This prsn is bankrupt.  I need to address the following:
  ;;   - deposit any cash into the checking account;
  ;;   - withdraw all savings (+ or -) and put into checking account;
  ;;   - resolve all 30-day receivables;
  ;;   - resolve all 30-day payables;
  ;;   - pay all interest payable;
  ;;   - collect all interest receivable;
  ;;   - pay off what can be paid on outstanding loan;
  ;;   - petition for a restart.

  LOG-TO-FILE( word "PRSN " who " is bankrupt." )
  ;; First, deposit cash, and move savings to checking.
  f-bnkrpt-prsn-collapses-cash-and-savings
  ;; Collect all 30-day receivables.
  f-bnkrpt-prsn-collects-all-30day-receivables
  ;; Collect all interest receivable.
  f-bnkrpt-prsn-collects-all-interest-receivable

  ;; Pay all 30-day payables.  Even if there is not enough money.
  ;;   This might run up a negative in L1-assets.
  f-bnkrpt-prsn-pays-all-30day-payables
  ;; Pay all interest payable.
  f-bnkrpt-prsn-pays-all-interest-payable
  ;; Use what assets remain to pay down the loan.
```

```
  f-bnkrpt-prsn-pays-down-loan
  ;; Due to the program structure, the prsn must initiate action
  ;;   to retire the loan, instead of the bank.
  f-bnkrpt-prsn-has-loan-written-off

  ;; TODO: Remove this after debug.
  ;; f-force-debug-output-off

  set g-counts-p-deaths ( g-counts-p-deaths + 1 )
  ;; The prsn has been removed from the model.
  ;; A replacement prsn may be added in the "do-post-tick" routine.
  set g-no-of-prsns ( count prsns )

  ;; The prsn now has zero assets of any kind, and can be removed.
  ;; Die MUST be the last command.
  die
;; end of f-bsvcs-process-prsn-bankruptcy
end

;;----------------------------------------------------------------------------|
;; A prsn collapses cash and savings account into checking account.
to f-bnkrpt-prsn-collapses-cash-and-savings
;; This routine is to be executed by a prsn.

  ;; This is done as part of bankruptcy proceedings.

  ;; Contact the bank.
  let my-bank ( bank bank-who )
  ;; PART A - Disbursement of assets and debts.
  ;; All of their assets are returned to the bank as L1-assets.
  ;; Then the residual of debts, after assets are cancelled, are
  ;;   written off.

  ;; L0 and P0 assets are deposited into the checking account.
  let my-P0-cash P0-assets  ;; note the amount.
  let my-L0-cash L0-assets  ;; note the amount.
  LOG-TO-FILE ( WORD "  Depositing cash assets" )
  LOG-TO-FILE ( word "  Checking account was ---------- " L1-assets )
  LOG-TO-FILE ( word "  Cash assets deposited --------- " my-L0-cash )
  LOG-TO-FILE ( word "  Physical cash deposited ------- " my-P0-cash )
  f-bsvcs-prsn-deposits-cash L0-assets
  LOG-TO-FILE ( word "  Checking account is now ------- " L1-assets )

  ;; There should be no savings, but things may have happened.
  ;; Savings may be positive or negative.
  ;; L2 assets are deposited into the checking account.
  LOG-TO-FILE ( word "  Savings transferred ----------- " L2-assets )
  f-bsvcs-prsn-moves-L2-to-L1 L2-assets
  LOG-TO-FILE ( word "  Checking account is now ------- " L1-assets )

;; end of f-bnkrpt-prsn-collapses-cash-and-savings
end

;;----------------------------------------------------------------------------|
;; A prsn collects ALL of the outstanding 30-day receivables.
to f-bnkrpt-prsn-collects-all-30day-receivables
;; This routine is to be executed by a prsn.

  ;; Contact my bank
  let my-bank ( bank bank-who )

  ;; Collect from everybody except myself.
  ;; The problem to be resolved is this.  The prsn has kept track of who
  ;;   it owes payment to, but not who owes payment to it.  This is
  ;;   for reasons of computer performance in daily activities, but it
  ;;   causes a problem during bankruptcy processing.  I need to canvass
  ;;   all other prsns, ask them what they owe me, then get them to
  ;;   pay now, in advance of the due date.

  LOG-TO-FILE ( word "  Collecting 30-day receivables" )
  let mywho who

  ;; Initialize an aggregator.
  let total-collected 0

  ask other prsns
  [
    let my-receivables ( filter [ mywho = ( item 0 ? ) ] payables-30day )
    set payables-30day ( filter [ mywho != ( item 0 ? ) ] payables-30day )

    ;; Initialize an aggregator.
    let amount-collected 0

    ;; Inter-bank payements by check require six entries.

    if ( ( length my-receivables ) > 0 )
    [
      ;; Contact his bank.
      let his-bank ( bank bank-who )

      ;; Process all of his payables that are due to the bankrupt prsn.
      foreach my-receivables
      [
        let amount-due ( item 2 ? )
        LOG-TO-FILE ( word "  Amount collected -------------- " amount-due )

        ;; Remove from payor's check-book.  Entry #1.
        set L1-assets ( L1-assets - amount-due )
        ;; Remove from bank of payor.  Entries #s 2 & 3.
        ask his-bank [ set L1-debts ( L1-debts - amount-due ) ]
        ask his-bank [ set L1-assets ( L1-assets - amount-due ) ]
        ;; Remove from his tally of total debts.
        set S1-30day-total-debts ( S1-30day-total-debts - amount-due )
        ;; Add to payor's tally of debts paid off under duress.
        set amount-collected ( amount-collected + amount-due )
      ]  ;; end of foreach receivable
      set total-collected ( total-collected + amount-collected )
      LOG-TO-FILE ( word "  Total collected - this prsn --- " amount-collected )
    ]  ;; end of if ( ( length my-receivables ) > 0 )
  ]  ;; end of ask other prsns

  ;; Enter the total collected into the payee's check book.  Entry #4.
  set L1-assets ( L1-assets + total-collected )
  ;; Update the bank's records.  Entries #5 & #6.
  ask my-bank [ set L1-debts ( L1-debts + total-collected ) ]
  ask my-bank [ set L1-assets ( L1-assets + total-collected ) ]
  ;; Update the aggregator.
  set S1-30day-total-assets ( S1-30day-total-assets - total-collected )
  LOG-TO-FILE ( word "  Total collected - all prsns --- " total-collected )
  LOG-TO-FILE ( word "  30day-assets are now ---------- " S1-30day-total-assets )
  LOG-TO-FILE ( word "  Checking account is now ------- " L1-assets )

;; end of f-bnkrpt-prsn-collects-all-30day-receivables
end
```

```
;;---------------------------------------------------------------------------|
;; A prsn collects ALL of the outstanding interest receivable.
to f-bnkrpt-prsn-collects-all-interest-receivable
;; This routine is to be executed by a prsn.

  ;; This would include interest on savings deposits.
  ;; TODO: Also includes interest on bonds, and stocks.  (Not yet implemented.)

  ;; Contact my bank
  let my-bank ( bank bank-who )

  ;; I want to paid an integral amount, but reduce the bank's
  ;;   records by the precise amount.
  let amount-due S1-L2ir-assets
  let amount-paid floor( S1-L2ir-assets )
  LOG-TO-FILE ( word "  Interest due on L2 savings ---- " amount-due )
  LOG-TO-FILE ( word "  Interest rec'd on L2 savings -- " amount-paid )
  let residual ( amount-due - amount-paid )

  ask my-bank
  [
    ;; Take the money from the bank's corporate funds.  Entry #1.
    set C1-assets ( C1-assets - amount-paid )
    ;; Reduce the off-books record of debt by the full amount due.  This
    ;;   effectively discards the fractional residual due.
    set S1-L2ip-debts ( S1-L2ip-debts - amount-due )
    ;; Two counteracting entries suppressed, for performance purposes.
    ;; set L1-debts ( L1-debts - amount-paid ) ;; Remove from bank.  Entry #2.
    ;; set L1-debts ( L1-debts + amount-paid ) ;; Insert to bank.  Entry #3.
  ]
  ;; Record the payment in bank book.  Entry #4.
  set L1-assets ( L1-assets + amount-paid )
  LOG-TO-FILE ( word "  Checking account is now ------- " L1-assets )
  LOG-TO-FILE ( word "  Residual ignored by both ------ " residual )
  set S1-L2ir-assets 0

;; end of f-bnkrpt-prsn-collects-all-interest-receivable
end

;;---------------------------------------------------------------------------|
;; A prsn pays all of the owed payables as part of bankruptcy processing.
to f-bnkrpt-prsn-pays-all-30day-payables
;; This routine is to be executed by a prsn.

  ;; As part of bankruptcy processing, pay all payables.
  LOG-TO-FILE ( word "  Paying 30-day payables" )

  ;; Contact my bank
  let my-bank ( bank bank-who )

  ;; Inter-bank payments by check require six entries.

  let total-paid 0 ;; Initialize an aggregator.

  if ( ( length payables-30day ) > 0 )
  [
    foreach payables-30day
    [
      let payee ( prsn ( item 0 ? ) )
      let amount-due item 2 ?
      ;; Aggregate the total for reporting purposes.
      set total-paid ( total-paid + amount-due )

      ask payee
      [
        ;; Contact his bank.
        let his-bank ( bank bank-who )

        ;; Put the money into his bank book.  Entry #1.
        set L1-assets ( L1-assets + amount-due )
        ;; Record it in his bank records.  Entries #2 & #3.
        ask his-bank [ set L1-debts ( L1-debts + amount-due ) ]
        ask his-bank [ set L1-assets ( L1-assets + amount-due ) ]

        ;; Reduce his record of receivables.
        set S1-30day-total-assets ( S1-30day-total-assets - amount-due )
        LOG-TO-FILE ( word "  Amount paid ------------------- " amount-due )
      ]
      ;; Mark the payment in bankruptee's bank book.  Entry #4.
      set L1-assets ( L1-assets - amount-due )

      ;; Inform the bank of the bankruptee.  Entries #5 & #6.
      ask my-bank [ set L1-debts ( L1-debts - amount-due ) ]
      ask my-bank [ set L1-assets ( L1-assets - amount-due ) ]

      ;; Reduce his record of payables.
      set S1-30day-total-debts ( S1-30day-total-debts - amount-due )

    ]  ;; end of foreach payable

    set S1-30day-total-debts 0   ;; All cleared.
    set payables-30day      []  ;; All cleared.
  ]  ;; end of if ( ( length payables-30day ) > 0 )
  LOG-TO-FILE ( word "  Total of all 30day paydowns --- " total-paid )
  LOG-TO-FILE ( word "  L1-assets post 30day paydowns - " L1-assets )

;; end of f-bnkrpt-prsn-pays-all-30day-payables
end

;;-----------------------------------------------------------------------------|
;; A prsn pays all interest payable.
to f-bnkrpt-prsn-pays-all-interest-payable
;; This routine is to be executed by a prsn.

  ;; This would include interest on bank loans deposits.

  ;; TODO: add log-to-file here and in all .

  ;; Contact my bank
  let my-bank ( bank bank-who )

  ;; Note the amount due.
  let amount-due S1-L1ip-debts
  ;; I want to pay an integral amount, but reduce the bank's
  ;;   records by the precise amount.
  let amount-paid floor( S1-L1ip-debts )
  LOG-TO-FILE ( word "  Interest on bank loan ------------ " amount-paid )
  let residual ( amount-due - amount-paid )

  ;; An intra-bank payment requires only 4 entries, two of which are suppressed.
  ask my-bank
  [
    ;; Put money into the bank's corporate funds.  Entry #1.
    set C1-assets ( C1-assets + amount-paid )
    ;; Change the off-book record by the precise amount, discarding residual.
    set S1-L1ir-assets ( S1-L1ir-assets - amount-due )
    ;; Two counteracting entries suppressed, for performance purposes.
    ;; set L1-debts ( L1-debts + amount-paid ) ;; Insert to bank.  Entry #2.
    ;; set L1-debts ( L1-debts - amount-paid ) ;; Remove from bank.  Entry #3.
  ]
  ;; Record the payment in bankrupt prsn's bank book.  Entry #4.
  set L1-assets ( L1-assets - amount-paid )
  ;; Change the off-book record by the precise amount, discarding the residual.
  set S1-L1ip-debts ( S1-L1ip-debts - amount-due )
  LOG-TO-FILE ( word "  L1-assets after interest paydown - " L1-assets )
  LOG-TO-FILE ( word "  Residual discarded --------------- " residual )

;; end of f-bnkrpt-prsn-pays-all-interest-payable
end

;;-----------------------------------------------------------------------------|
;; A prsn pays down the loan as far as possible.
to f-bnkrpt-prsn-pays-down-loan
;; This routine is to be executed by a prsn.

  ;; This is part of bankruptcy processing.
  ;; The prsn uses whatever resources remain to pay down the loan.
  ;; Note that those resources (in L1-assets) may be positive or
  ;;   negative, and may reduce the loan or add to it.
  ;; Such a payment is within one bank/client relationship, and
  ;;   can be completed with four entries.

  ;; Contact my bank
  let my-bank ( bank bank-who )

  let amount-paid L1-assets

  ask my-bank
  [
    ;; Pay money against the loan.  This brings down the value of
    ;;   the loan.  Entry #1.
    set L1-loan-assets ( L1-loan-assets - amount-paid )
    ;; Debts follow assets.  The net value of the funds in public
    ;;   trust must not change.  So the amount of L1-funds made
    ;;   available to the client must be removed from the client's
    ;;   checking account.  Entry #2.
    set L1-debts  ( L1-debts  - amount-paid ) ;; Insert to bank.
    ;; The net worth of the bank's books has not changed.
  ]
  ;; Record a reduction in the principal of the loan.  Entry #3.
  set L1-loan-debts ( L1-loan-debts - amount-paid )
  ;; Record the payment in bankrupt prsn's bank book.  Entry #4.
  set L1-assets ( L1-assets - amount-paid )
  ;; The net worth of the client has not changed.

  LOG-TO-FILE ( word "  L1-assets after loan paydown ----- " L1-assets )

;; end of f-bnkrpt-prsn-pays-down-loan
end

;;-----------------------------------------------------------------------------|
;; A prsn requests the loan be written off.  The bank agrees.
to f-bnkrpt-prsn-has-loan-written-off
;; This routine is to be executed by a prsn.
```

```
  ;; This is part of bankruptcy processing.
  ;; The prsn asks the bank to forgive the debt.

  ;; The size of the loan is determined by the client's loan record.
  ;; This is because the bank's loan record is an aggregate for all
  ;;   of its loans.
  let amount-written-off L1-loan-debts

  ;; THEORY:  This can be handled two different ways.  Either the bank that
  ;;   has serviced the bankruptee up until now can bear the brunt of the
  ;;   bankruptcy, or the loss can be spread across all banks.  I call this
  ;;   control bank insurance.

  ;; Contact my bank
  let my-bank ( bank bank-who )

  ;; THEORY:  Cancel the debt.  This is tricky.  At this point all of the
  ;;   assets and debts of the bankrupt person have been converted to
  ;;   be part of the loan.  There are no S1, L1, or L2 assets or
  ;;   liabilities other than the L1-loan.  For a single-bank transaction
  ;;   the net change in the back room must be zero, and transactional
  ;;   conservation of money requires that two other offsetting entries
  ;;   must be made.  The client will have the loan written-off, but
  ;;   has no assets for the required offset.  The bank must provide those
  ;;   assets, and so it takes a loss on the loan.
  ;; In double-entry bookkeeping terms:
  ;;   The bank's loan-asset offsets the prsn's loan-debt.
  ;;   The bank's L1-debt should be offset by the prsn's L1-asset.
  ;;   But the prsn has no L1-asset.  It has been stripped away.
  ;;   So, the bank's corporate C1-asset "eats the loss" and is
  ;;   used to settle the loan.  In this option, that loss is spread across
  ;;   all banks.

  LOG-TO-FILE ( word "Loan is being written off." )
  ;; Cancelling a loan requires four entries.
  ;; So, the client is informed that the loan is written off.  Entry #1.
  LOG-TO-FILE ( word "  Checking account is now - " L1-assets )
  LOG-TO-FILE ( word "  Outstanding loan debt --- " L1-loan-debts )
  set L1-loan-debts ( L1-loan-debts - amount-written-off )
  LOG-TO-FILE ( word "  Amount written off ------ " amount-written-off )
  LOG-TO-FILE ( word "  Remaining loan debt ----- " L1-loan-debts )
  ;; Note that there are no client L1 assets remaining to be co-cancelled.
  ;; They have wandered off to the L1-asset accounts of some other prsns.

  ask my-bank
  [
    ;; Bank cancels the loan to this client by reducing its aggregator.
    ;;   Entry #2.
    LOG-TO-FILE ( word "  Bank's loan assets were - " L1-loan-assets )
    set L1-loan-assets ( L1-loan-assets - amount-written-off )
    LOG-TO-FILE ( word "  Bank's loan assets are -- " L1-loan-assets )
    ;; To maintain the back room net worth, an equivalent amount of L1
    ;;   funds available to the economy must be withdrawn from action
    ;;   effectively shrinking the MS-II money supply.  Entry #3.
    set L1-debts  ( L1-debts  - amount-written-off )

    ;; Finally, someone active in the economy needs to cough up the money
    ;;   that has been withdrawn.  The bankrupt client cannot provide it.
    ;;   That money has wandered off to who-knows-where.  So the front room
    ;;   of the bank must provide it out of its C1 corporate accounts.
    ;;   The front room of the bank is a customer of its own back room.  So
    ;;   this amounts to a payment from the corporate bank to the client
    ;;   cancelling its debt.  Entry #4.
    LOG-TO-FILE ( word "  Bank's C1 assets were --- " C1-assets )
    set C1-assets ( C1-assets - amount-written-off )
    LOG-TO-FILE ( word "  Bank's C1 assets are ---- " C1-assets )
  ]

  ;; Prsn takes over again.

  ;; Only invoke insurance if there is a clear loss.
  ;;   Sometimes a prsn goes bankrupt with a minor positive net worth.
  if( amount-written-off > 0 )
  [
    if( gb-bank-insurance = true )
    [
      LOG-TO-FILE ( word "  Banking insurance is on." )
      ;; Bank insurance is turned on.  All banks share the loss.
      ;; At this point, my-bank has born the whole cost.  Now, refund it.
      LOG-TO-FILE ( word "  Amount refunded --------- " amount-written-off )
      ask my-bank [ set C1-assets ( C1-assets + amount-written-off ) ]
      ask my-bank [LOG-TO-FILE ( word "  Bank's C1 assets are ---- " C1-assets ) ]

      ;; Determine the status before the write-off.
      let total-C1-assets ( sum [C1-assets] of banks )
      let donation-factor 0  ;; a dummy declaration.
      let donation        0  ;; a dummy declaration.
      let total-donation  0  ;; a dummy declaration.
      ;; My bank will also make a donation, and receive the donation, to cover
      ;;   its portion of the cost.  This makes the code more simple.
      ask banks
      [
        set donation-factor ( C1-assets / total-C1-assets )
        set donation floor( amount-written-off * donation-factor )
        LOG-TO-FILE ( word "  Bank " who " donated ------- " donation )
        ;; This is an intra-bank cost.  It requires three entries.
        ;; Mark in corporate check books.  Entry #1.
        set C1-assets ( C1-assets - donation )
        ;; Make the back room entries.  Entries #2 and #3.
        set L1-assets ( L1-assets - donation )
        set L1-debts ( L1-debts - donation )
        ;; Keep an aggregate tally.  Includes a self-donation.
        set total-donation ( total-donation + donation )
      ]

      ;; Due to rounding, the total donated (and written off, in each
      ;;   case) may not equal the amount to be written off.  My bank
      ;;   has already taken its share of the lumps given, but it must
      ;;   also handle the residual.
      ask my-bank
      [
        let residual ( amount-written-off - total-donation )
        ;; Mark in corporate check book.  Entry #1.
        set C1-assets ( C1-assets - residual )
        ;; Make back room entries.  Entries #2 and #3.
        set L1-assets ( L1-assets - residual )
        set L1-debts ( L1-debts - residual )
      ]
    ]  ;; end if (gb-bank-insurance = true)
  ]

;; end of f-bnkrpt-prsn-has-loan-written-off
end
```

```
;;----------------------------------------------------------------------------|
;; Process a bank that is bankrupt.
to f-bsvcs-process-bank-bankruptcy
;; This routine is to be executed by a bank.

  ;; TODO: After debugging, suppress this.
  ;; f-force-debug-output-on
  ;; TODO: Remove this if annoying.
  ;; beep

  ;; PART A - I need to collapse the assets and declare bankruptcy.
  ;; Banks are bankrupt when they have insufficient P0-assets to make loans
  ;;   or earn interest from the CRB, and they have no existing L1 loans.
  ;; When they last attempted to issue a loan, the bank would have marked a
  ;;   failed loan request as its own bankruptcy.
  ;; So, I need to collapse the assets and debts of this bank.

  ASSERT ( b-bank-is-bankrupt = 1 ) "Bank not bankrupt" who

  ;; This bank is bankrupt.  I need to address the following:
  ;;   - send GCRA account, if there is one, to another bank;
  ;;   - disperse all client accounts to other banks;
  ;;   - disperse all P0 assets to other banks;
  ;;   - disperse all -tve C1 assets to other banks, who must share the losses;

  LOG-TO-FILE( word "BANK " who " is bankrupt." )

  ;; Send the GCRA to another bank.
  if( no-of-gcra-clients > 0 )
  [
    let new-bank one-of other banks
    let new-bank-who [who] of new-bank
    ask gcras [ set bank-who new-bank-who ]
    LOG-TO-FILE ( word "  GCRA has a new bank ------------- " new-bank-who )
    set no-of-gcra-clients 0
    ask new-bank [ set no-of-gcra-clients ( no-of-gcra-clients + 1 ) ]
  ]

  ;; Send the CRB to another bank.
  if( no-of-crb-clients > 0 )
  [
    let new-bank one-of other banks
    let new-bank-who [who] of new-bank
    ask crbs [ set bank-who new-bank-who ]
    LOG-TO-FILE ( word "  CRB has a new bank -------------- " new-bank-who )
    set no-of-crb-clients 0
    ask new-bank [ set no-of-crb-clients ( no-of-crb-clients + 1 ) ]
  ]

  ;; Disperse other clients to new banks.
  ifelse( no-of-prsn-clients > 0 )
  [
    ;; Get a list of prsns that use this bank.
    let client-list ( prsns with [bank-who = who] )
    LOG-TO-FILE( word "  Client list: " [who] of client-list )
    ;; Get a list of suitable banks.
    let bank-list ( other banks )
    LOG-TO-FILE( word "  Alternate bank list: " [who] of bank-list )
    ask client-list
    [
      ;; Each prsn moves accounts to a new bank.
      ;; P0 assets (currency) does not need to be moved.  It is not in
```

```
    ;;   the bank.
    ;; L1-loans do not need to be moved.  A condition of bankruptcy is
    ;;   this bank has no outstanding loans, and no RR or ER deposits.
    let old-bank ( bank bank-who )     ;; who of bankrupt bank.
    let old-bank-who ( [who] of old-bank )
    let new-bank ( one-of bank-list )  ;; who of some other bank.
    set bank-who ( [who] of new-bank ) ;; bank-to-bank client transfer
    LOG-TO-FILE( word "  Prsn " who " moves from bank "
      old-bank-who " to " bank-who "." )

    ;; Move the assets.  This requires 6 entries.
    ;; No entry is needed in the client's checkbook.
    let L1-to-move L1-assets
    let L2-to-move L2-assets
    LOG-TO-FILE( word "  L1-assets moved --------------" L1-assets )
    LOG-TO-FILE( word "  L2-assets moved --------------" L2-assets )
    ask old-bank
    [
      ;; Entries #1, #2 and #3.
      set L1-assets ( L1-assets - L1-to-move )
      set L1-debts  ( L1-debts  - L1-to-move )
      set L2-debts  ( L2-debts  - L2-to-move )
    ]
    ask new-bank
    [
      ;; Entries #4, #5 and #6.
      set L1-assets ( L1-assets + L1-to-move )
      set L1-debts  ( L1-debts  + L1-to-move )
      set L2-debts  ( L2-debts  + L2-to-move )
    ]

    ;; Cancel any shadow debts.
    ask old-bank
    [
      ;; Remove this client's interest payable on L1-loans.
      set S1-L1ir-assets ( S1-L1ir-assets - S1-L1ip-debts )
      LOG-TO-FILE( word "  S1-L1ip-debts cancelled ------" S1-L1ip-debts )
      ;; Remove this client's interest receivable on L2 savings.
      set S1-L2ip-debts ( S1-L2ip-debts - S1-L2ir-assets )
      LOG-TO-FILE( word "  S1-L2ir-assets cancelled -----" S1-L2ir-assets )
    ]
    set S1-L1ip-debts  0
    set S1-L2ir-assets 0
  ] ;; end of ask client-list
] ;; end of ifelse( no-of-prsn-clients > 0 )
;; else
[
  LOG-TO-FILE( word "  No clients affected." )
]

;; Distribute any C1-assets (whether +ve or -ve).
;; Distribute any P0-assets.
;; So, first, pack up the P0 assets.
f-cbsvcs-bank-moves-rr-to-vc P0-rr-assets
f-cbsvcs-bank-moves-er-to-vc P0-er-assets
let P0-assets-to-move P0-vc-assets

ifelse( P0-assets-to-move > 0 )
[
  LOG-TO-FILE( word "  P0-assets to move -------------- " P0-assets-to-move )
  let no-of-banks ( count banks )
  let one-C1-share floor( C1-assets / ( no-of-banks - 1 ) )
  let C1-residual ( C1-assets - ( one-C1-share * ( no-of-banks - 1 ) ) )
  let one-P0-share floor( P0-vc-assets / ( no-of-banks - 1 ) )
  let P0-residual ( P0-vc-assets - ( one-P0-share * ( no-of-banks - 1 ) ) )
  ;; Give every bank one share of asset/debt of each kind.
  ask other banks
  [
    ;; This is a bank-to-bank check.  It requires six entries.
    ;; Mark in the bank's checkbook.  Entry #1.
    set C1-assets ( C1-assets + one-C1-share )
    ;; Mark in the back room records.  Entries #2 and #3.
    set L1-assets ( L1-assets + one-C1-share )
    set L1-debts ( L1-debts + one-C1-share )
    ;; Add the physical cash to the vault.
    set P0-vc-assets ( P0-vc-assets + one-P0-share )
    LOG-TO-FILE( word "  P0-assets moved to bank " who " - "  one-P0-share )
  ]
  ;; Mark in the back room books.  Entries #4 and #5.
  set L1-assets ( L1-assets - C1-assets )
  set L1-debts ( L1-debts - C1-assets )
  ;; Mark in this bank's check book.  Entry #6.  Assets are gone.
  set C1-assets    0
  set P0-vc-assets 0

  ;; One bank paid a full share when it should only have paid the
  ;;   residual, which may not be a full share.  Correct this.
  ask one-of other banks
  [
    ;; It requires six entries.
    ;; Mark in the bank's checkbook.  Entry #1.
    set C1-assets ( C1-assets - one-C1-share )
    ;; Mark in the back room records.  Entries #2 and #3.
    set L1-assets ( L1-assets - one-C1-share )
    set L1-debts ( L1-debts - one-C1-share )
    ;; Mark in the bank's checkbook.  Entry #4.
    set C1-assets ( C1-assets + C1-residual )
    ;; Mark in the back room records.  Entries #5 and #6.
    set L1-assets ( L1-assets + C1-residual )
    set L1-debts ( L1-debts + C1-residual )
    ;; Adjust the physical cash.
    set P0-vc-assets ( P0-vc-assets - one-P0-share )
    set P0-vc-assets ( P0-vc-assets + P0-residual )
    LOG-TO-FILE( word "  P0-assets change at bank " who " - "
      ( P0-residual - one-P0-share ) )
  ]
] ;; end ifelse( P0-assets-to-move > 0 )
;; else
[
  LOG-TO-FILE( word "  No P0-assets need to move. " )
]

ifelse( ( S1-rrir-assets > 0 ) or ( S1-rrir-assets > 0 ) )
[
  ;; Cancel any interest receivable on ER and RR.  Probably none.
  let crb-bank one-of crbs
  let rrir-to-cancel S1-rrir-assets
  let erir-to-cancel S1-erir-assets
  ask crb-bank
  [
    set S1-rrip-debts ( S1-rrip-debts - rrir-to-cancel )
    LOG-TO-FILE( word "  S1-rrir-assets cancelled -------- "  rrir-to-cancel )
    set S1-erip-debts ( S1-erip-debts - erir-to-cancel )
    LOG-TO-FILE( word "  S1-erir-assets cancelled -------- "  erir-to-cancel )
```

```
    ]
    set S1-rrir-assets 0
    set S1-erir-assets 0
  ] ;; end ifelse( ( S1-rrir-assets> 0 ) or ( S1-rrir-assets > 0 ) )
  ;; else
  [
    LOG-TO-FILE( word "  No interest receivables need be cancelled. " )
  ]

  ;; The bank has been removed from the model.
  ;; A replacement bank may be added in the "do-post-tick" routine.
  set g-no-of-banks ( count banks )

  ;; TODO: Remove this after debug.
  ;; f-force-debug-output-off

  ;; This bank has now been stripped of all assets and debts, and
  ;;   all connections to clients of all kinds.
  set g-counts-b-deaths ( g-counts-b-deaths + 1 )
  ;; Die MUST be the last command.
  die

;; end of f-bsvcs-process-bank-bankruptcy
end

;;-------------------------------------------------------------------------|
;; START OF -CBSVCS- SUB-SECTION.
;;-------------------------------------------------------------------------|
;; These routines involve the Central Reserve Bank (CRB) and its services.
;; THEORY: In this section of the code all of the patterns for types of central
;;   bank services have been pulled together in a single place.  This is to
;;   enable consistency in the means of implmenting each type of service, with
;;   the hope that it will make coding, debugging, and maintenance easier, at
;;   a possible cost of performance.
;; Note that it is intentional that none of these routine do range error
;;   checking on the variables affected.  So, for example, a bank with no cash
;;   in an excess reserve account may still move cash from there to its vault.
;; The creation of negatives and their ultimate removal again all gets
;;   resolved in the daily visit to the CRB by each bank.  If a bank becomes
;;   overextended, a boolian switch is flipped that prevents further action
;;   until clients pay down their loans and the bank is no longer over-extended.
;; The real purpose of these routines is to defend the public trust that
;;   physical money is properly conserved unless explicitly indicated otherwise.
;; Rather that implementing the complicated issue of linking CRB accounts
;;   directly to banks, the banks keep track of the details of their own
;;   accounts, and the CRB only keeps track of aggregate amounts.  This
;;   simplifies the coding dramatically, and so reduces the chances of coding
;;   error, but it puts the onus on the banks to have their books in order.
;;   These central bank routines look after that.


;;-------------------------------------------------------------------------|
;; Distribute the initial endowment of assigned assets to prsns.
to f-cbsvcs-distribute-assets-to-prsns
  ;; This routine is to be executed by the CRB.

  LOG-TO-FILE ( word "" )
  LOG-TO-FILE ( word "Distribution of Money Base by CRB"  )

  ;; Establish CRB endowment by fiat.
   ;; Physical dollars
  set P0-assets ( g-no-of-prsns-max * g-crb-assets-per-prsn )
```

```
;; Logical dollars
set L0-assets P0-assets
;; THEORY: On start, assets must just appear to imply fiat creation.
;; When it is handed out as wages, or, if you wish, as a share
;;   of ownership in the society and economy, a liability is created
;;   for the government, in the person of the CRB.
;; Each cash dollar held, as a personal asset, implies a government-backed
;;   promise to pay in legal tender (gold, or replacement dollars,
;;   or ?? ).
set P0-debts 0
set L0-debts 0
;; I use the code word "debts" to mean "liabilities" just because it
;;   is shorter.  Note that, for banks, these words have somewhat
;;   counter-intituitive meanings.

;; Store the who of the CRB for access by prsns.
let crbwho who
;; Create a handle for the CRB.
let the-crb ( crb crbwho )

ask prsns
[
  ;; Determine how much to give to each prsn.
  let per-person-endowment g-crb-assets-per-prsn

  ;; Put cash into the hands of the prsn.
  ;; $1 cash = ( $1 logical + $1 physical )
  set P0-assets per-person-endowment
  set L0-assets per-person-endowment

  ask the-crb
  [
    ;; THEORY: Adjust CRB's records for each prsn.
    ;; The associated liability is created at the CRB.
    ;; It does not move.  This is part of the "fiat" process of
    ;;   creating valued currency in the economy.
    ;; The ultimate result is currency in the economy that has value
    ;;   because the government guarantees that it can be exchanged
    ;;   for value (in kind, in gold, or in replacement dollars).

    ;; Remove physical and logical $ from CRB assets.
    ;; Logical money is treated as an increase in logical liability.
    set L0-debts ( L0-debts + per-person-endowment )
    ;; Physical money is actually removed from CRB vaults.
    set P0-assets ( P0-assets - per-person-endowment )
  ]
]

;; The prsns deposit some cash, creating checking and savings accounts.
ask prsns [ f-prsn-visits-a-bank ]

;; The currency assets are now all out in the economy, while the
;;   currency liabilities are all in the CRB.

LOG-TO-FILE ( word "  After CRB distribution"  )
LOG-TO-FILE ( word "  CRB P0-assets ------------------ " P0-assets  )
LOG-TO-FILE ( word "  CRB L0-assets ------------------ " L0-assets  )
LOG-TO-FILE ( word "  CRB P0-debts ------------------- " P0-debts  )
LOG-TO-FILE ( word "  CRB L0-debts ------------------- " L0-debts  )
LOG-TO-FILE ( word "  CRB P0-rr-assets --------------- " P0-rr-assets  )
LOG-TO-FILE ( word "  CRB P0-er-assets --------------- " P0-er-assets  )
```

```
  let sum-of-P0 ( sum [P0-assets] of prsns )
  let sum-of-L0 ( sum [L0-assets] of prsns )
  LOG-TO-FILE ( word "  All Prsns P0-assets ------------ " sum-of-P0  )
  LOG-TO-FILE ( word "  All Prsns L0-assets ------------ " sum-of-L0  )

  ;; End of f-cbsvcs-distribute-assets-to-prsns
end

;;----------------------------------------------------------------------------|
;; The GCRA (Govt Consolidated Revenue Accts) are reconciled with banks.
to f-cbsvcs-gcra-reconciles-with-crb-monthly
;; This routine is to be executed by the observer.

  ;; THEORY: The GCRA might deal with a bank for a couple of reasons.
  ;; 1. The CRB must pay interest on reserve deposits, and this must come out
  ;;   of the government consolidated revenue accounts (GCRA).  So interest
  ;;   on both ER deposits and RR deposits must be accounted for.
  ;; 2. TODO: The CRB might loan out reserves to banks that need them, and so
  ;;   may collect interest on those loans, which would go into GCRA.
  ;; 3. TODO: Expenses from gov't buying may exceed income from taxes, and so
  ;    the government may want to address the budget deficit with a normal
  ;;   L1 bank loan from a chartered bank.
  ;; TODO: Only item #1 is implemented so far.

  ;; In all cases, the positive and negative changes in the corporate assets
  ;;   and liabilities of the CRB are reflected in the variable C1-assets.

  ;; Contact the CRB.
  let the-crb ( one-of crbs ) ;; There is only one CRB.

  ;; Contact the chartered bank that holds the CRB's C1 account.
  let bank-of-crb ( bank ( [bank-who] of the-crb ) )

  ask gcras  ;; There is only one GCRA.
  [
    ;; Contact the chartered bank used by the GCRA.
    let gcra-bank ( bank bank-who )

    ;; Move the private (i.e. "corporate") assets and debts from the CRB
    ;;   into the government consolidated revenue accounts.
    let amount-to-transfer ( [C1-assets] of the-crb )

    LOG-TO-FILE ( word "" )
    LOG-TO-FILE ( word "GCRA visits CRB." )
    LOG-TO-FILE ( word "TRANSFER CRB CORP ACCTS TO GCRA" )
    LOG-TO-FILE ( word "  GCRA L1 assets prior to xfer ---- " L1-assets )
    LOG-TO-FILE ( word "  CRB C1 assets prior to xfer ----- " amount-to-transfer )

    ;; NOTE: I use negative assets to record debts.
    ;; This inter-bank payment requires six entries.
    ;; The amount-to-transfer moves from CRB assets to GCRA assets.

    ;; Entry #1.  Add the assets to the check book of the GCRA.
    set L1-assets ( L1-assets + amount-to-transfer )
    ;; Entry #2.  Add the liability to the bank of the GCRA.
    ask gcra-bank [ set L1-debts ( L1-debts + amount-to-transfer ) ]
    ;; Entry #3.  Assets must follow debts.
    ask gcra-bank [ set L1-assets ( L1-assets + amount-to-transfer ) ]
    LOG-TO-FILE ( word "  GCRA L1 assets after xfer ------- " L1-assets )
    ;; At this point the GCRA has the assets, and the net worth of
    ;;  the chartered bank that deals with the GCRA has not changed.
    ;; Remove from the CRB account.
    ask the-crb
    [
      ;; Entry #4.  Remove the assets from the CRB's check book.
      set C1-assets ( C1-assets - amount-to-transfer )
      ;; A payment usually requires six entries.  Two into the bank
      ;;   books of the participants, and four back-room entries by the
      ;;   banks recording the change in assets/liability for the banks.
      ;;   This exchange involves three banks: the CRB and two chartered
      ;;   banks in which the GCRA stores its funds.

      LOG-TO-FILE ( word "   CRB C1 assets after xfer -------- " C1-assets )
    ]

    ask bank-of-crb
    [
      ;; Entry #5.  Record the change in liabilities.
      set L1-debts ( L1-debts - amount-to-transfer )
      ;; Entry #6.  Assets follow liabilities.
      set L1-assets ( L1-assets - amount-to-transfer )
    ]

    ;; The transaction is completed.  The net worth of both chartered bank's
    ;;   back room records has not changed, but the assets have moved from
    ;;   the CRB's C1 account to the GCRA's L1 account.
  ]

;; end of f-cbsvcs-gcra-reconciles-with-crb-monthly
end

;;---------------------------------------------------------------------------|
;; A bank has vault cash (vc) and deposits into its excess reserve (ER)
;;   account at the CRB.
to f-cbsvcs-bank-moves-vc-to-er [ amount-to-move ]
;; This routine is to be executed a bank.

  ;; Contact the CRB.
  let the-crb ( crb crb-who )

  ;; Move the physical cash within the bank's records.
  set P0-vc-assets ( P0-vc-assets - amount-to-move )
  ;; Adjust the phantom account in which assets = liabilities.
  set P0-er-assets ( P0-er-assets + amount-to-move )
  set P0-er-debts  ( P0-er-debts  + amount-to-move )

  ;; Put the physical cash into the CRB's vault as ER (P0-er).
  ask the-crb
  [
    ;; Add it to the aggregate ER amount in the CRB.
    set P0-er-assets ( P0-er-assets + amount-to-move )
  ]

  LOG-TO-FILE ( word "  CBSvcs: Amount of ER deposited -- " amount-to-move )

;; end of f-cbsvcs-bank-moves-vc-to-er
end

;;---------------------------------------------------------------------------|
;; A bank has ER funds in the CRB and withdraws physical cash (P0).
to f-cbsvcs-bank-moves-er-to-vc [ amount-to-move ]
;; This routine is to be executed a bank.

  ;; Contact the bank.
  let the-crb ( crb crb-who )

  ;; This is the reversal of a move vc-to-er.

  ;; Get the physical cash from the CRB's vault as ER (P0-er).
  ask the-crb
  [
    ;; Subract it from the aggregate ER amount in the CRB.
    set P0-er-assets ( P0-er-assets - amount-to-move )
  ]

  ;; Move the physical cash within the bank's records.
  set P0-vc-assets ( P0-vc-assets + amount-to-move )
  ;; Adjust the phantom account in which assets = liabilities.
  set P0-er-assets ( P0-er-assets - amount-to-move )
  set P0-er-debts  ( P0-er-debts  - amount-to-move )

  LOG-TO-FILE ( word "  CBSvcs: Amount of ER withdrawn -- " amount-to-move )

;; end of f-cbsvcs-bank-moves-er-to-vc
end

;;---------------------------------------------------------------------------|
;; A bank has vault cash (vc) and deposits into its required reserve (RR)
;;   account at the CRB.
to f-cbsvcs-bank-moves-vc-to-rr [ amount-to-move ]
;; This routine is to be executed a bank.

  ;; Contact the CRB.
  let the-crb ( crb crb-who )

  ;; Move the physical cash within the bank's records.
  set P0-vc-assets ( P0-vc-assets - amount-to-move )
  ;; Adjust the phantom account in which assets = liabilities.
  set P0-rr-assets ( P0-rr-assets + amount-to-move )
  set P0-rr-debts  ( P0-rr-debts  + amount-to-move )

  ;; Put the physical cash into the CRB's vault as RR (P0-er).
  ask the-crb
  [
    ;; Add it to the aggregate ER amount in the CRB.
    set P0-rr-assets ( P0-rr-assets + amount-to-move )
  ]

  LOG-TO-FILE ( word "  CBSvcs: Amount of RR deposited -- " amount-to-move )

;; end of f-cbsvcs-bank-moves-vc-to-rr
end

;;---------------------------------------------------------------------------|
;; A bank has RR funds in the CRB and withdraws physical cash (P0).
to f-cbsvcs-bank-moves-rr-to-vc [ amount-to-move ]
;; This routine is to be executed a bank.

  ;; Contact the bank.
  let the-crb ( crb crb-who )

  ;; This is the reversal of a move vc-to-rr.

  ;; Get the physical cash from the CRB's vault as RR (P0-rr).
  ask the-crb
  [
    ;; Subract it from the aggregate RR amount in the CRB.
    set P0-rr-assets ( P0-rr-assets - amount-to-move )
  ]

  ;; Move the physical cash within the bank's records.
  set P0-vc-assets ( P0-vc-assets + amount-to-move )
  ;; Adjust the phantom account in which assets = liabilities.
  set P0-rr-assets ( P0-rr-assets - amount-to-move )
  set P0-rr-debts  ( P0-rr-debts  - amount-to-move )

  LOG-TO-FILE ( word "  CBSvcs: Amount of RR withdrawn -- " amount-to-move )

;; end of f-cbsvcs-bank-moves-rr-to-vc
end

;;----------------------------------------------------------------------------|
;; The CRB is charged daily interest on outstanding amounts of ER deposits.
to f-cbsvcs-bank-accrues-daily-interest-on-ER-deposits
;; This routine is to be executed a bank.

  ;; THEORY: -ptbfs- This causes a flow of money from the real
  ;;   economy to the banking sector because the interest on excess
  ;;   reserves is paid by the government to the banks out of the
  ;;   Consolidated Revenue Accounts of the government, which comes out
  ;;   of personal taxes.  As such, it is part of the "Prsns to Banks
  ;;   Flows" (ptbfs).  It can be turned off by setting g-ioer to zero.

  if( g-ioer > 0 )
  [
    ;; THEORY: Interest on ER deposits is to be paid by the CRB to the bank.
    ;;   The size of the deposits may vary daily due to commercial activity,
    ;;   so interest is charged and accrued on a daily basis, but only
    ;;   paid on a monthly basis.  This interest is a debt which expands the
    ;;   shadow money supply, as it is basically a loan from the bank to the
    ;;   CRB until it is paid.
    ;;
    ;;  I note that this makes sense only if the CRB can then loan out
    ;;    any excess physical cash (P0) held in ER deposits to other banks, in
    ;;    place of using fiat powers to create more physical cash (P0, L0) when
    ;;    needed.  In this way the CRB can expand the physical money supply in a
    ;;    fashion similar to the way a chartered bank can expand the logical money
    ;;    supply.  I have NOT implemented this.  In this model, the physical money
    ;;    supply is not expandable by that technique, though it would be easy to
    ;;    add.
    ;;
    ;; The same as for L1 loans, there is a hair to be split, here, and I am
    ;;   splitting it this way.  Because this debt is visible to the banks,
    ;;   and really amounts to a bank loan of sorts, it should be considered
    ;;   part of the logical money supply (L1) instead of the shadow money
    ;;   supply (S1).
    ;; But, because I want to focus on L1 loan tracking in this application, I have
    ;;   chosen, somewhat arbitrarily, to include it in S1 until it is paid.

    ;; Contact the CRB.
    let the-crb ( crb crb-who )

    ;; The CRB only has an aggregate variable for all of the ER deposits of all
    ;;   of its client banks.  Only the bank's records indicate the size of the
    ;;   ER deposit associated with this bank.
    let er-account-size P0-er-assets
    ;; The annual interest on ER deposits is in slider g-ioer.
    let annual-interest-due ( er-account-size * g-ioer / 100 )
    ;; Prorate this to a daily rate (12 months; 30 days per month).
    let daily-interest-due ( annual-interest-due / ( 12 * 30 ) )

    ;; The CRB records the increase in its S1 aggregator for
    ;;   ER deposits (P0-er) interest payable.
    ask the-crb [ set S1-erip-debts ( S1-erip-debts + daily-interest-due ) ]
    ;; The bank records the increase in its S1 record for interest receivable.
    set S1-erir-assets ( S1-erir-assets + daily-interest-due )

    LOG-TO-FILE ( word "  CBSvcs: ER interest accrued ----- " daily-interest-due )

  ]

;; end of f-cbsvcs-bank-accrues-daily-interest-on-ER-deposits
end

;;----------------------------------------------------------------------------|
;; The CRB is charged daily interest on outstanding amounts of RR deposits.
to f-cbsvcs-bank-accrues-daily-interest-on-RR-deposits
;; This routine is to be executed a bank.

  ;; THEORY: -ptbfs- This causes a flow of money from the real
  ;;   economy to the banking sector because the interest on required
  ;;   reserves is paid by the government to the banks out of the
  ;;   Consolidated Revenue Accounts of the government, which comes out
  ;;   of personal taxes.  As such, it is part of the "Prsns to Banks
  ;;   Flows" (ptbfs).  It can be turned off by setting g-iorr to zero.

  if( g-iorr > 0 )
  [
    ;; THEORY: Interest on RR deposits is to be paid by the CRB to the bank.
    ;;   The size of the deposits may vary daily due to commercial activity,
    ;;   so interest is charged and accrued on a daily basis, but only
    ;;   paid on a monthly basis.  This interest is a debt which expands the
    ;;   shadow money supply, as it is basically a loan from the bank to the
    ;;   CRB until it is paid.
    ;;
    ;;  I note that this makes sense only if the CRB can then loan out
    ;;    any excess physical cash (P0) held in ER deposits to other banks, in
    ;;    place of using fiat powers to create more physical cash (P0, L0) when
    ;;    needed.  In this way the CRB can expand the physical money supply in a
    ;;    fashion similar to the way a chartered bank can expand the logical money
    ;;    supply.  I have NOT implemented this.  In this model, the physical money
    ;;    supply is not expandable by that technique, though it would be easy to
    ;;    add.
    ;;
    ;; The same as for L1 loans, there is a hair to be split, here, and I am
    ;;   splitting it this way.  Because this debt is visible to the banks,
    ;;   and really amounts to a bank loan of sorts, it should be considered
    ;;   part of the logical money supply (L1) instead of the shadow money
    ;;   supply (S1).
    ;; But, because I want to focus on L1 loan tracking in this application, I have
    ;;   chosen, somewhat arbitrarily, to include it in S1 until it is paid.

    ;; Contact the CRB.
    let the-crb ( crb crb-who )

    ;; The CRB only has an aggregate variable for all of the RR deposits of all
    ;;   of its client banks.  Only the bank's records indicate the size of the
    ;;   RR deposit associated with this bank.
    let rr-account-size P0-rr-assets
    ;; The annual interest on RR deposits is in slider g-iorr.
    let annual-interest-due ( rr-account-size * g-iorr / 100 )
    ;; Prorate this to a daily rate (12 months; 30 days per month).
    let daily-interest-due ( annual-interest-due / ( 12 * 30 ) )

    ;; The CRB records the increase in its S1 aggregator for
    ;;   RR deposits (P0-rr) interest payable.
    ask the-crb [ set S1-rrip-debts ( S1-rrip-debts + daily-interest-due ) ]
    ;; The bank records the increase in its S1 record for interest receivable.
    set S1-rrir-assets ( S1-rrir-assets + daily-interest-due )

    LOG-TO-FILE ( word "  CBSvcs: RR interest accrued ----- " daily-interest-due )

  ]

;; end of f-cbsvcs-bank-accrues-daily-interest-on-RR-deposits
end

;;----------------------------------------------------------------------------|
;; A client pays outstanding interest on er deposits monthly.
to f-cbsvcs-bank-paid-monthly-interest-on-er-deposits
;; This routine is to be executed by a bank.

  ;; THEORY: Interest on ER deposits is to be paid by the CRB to the bank.
  ;;   It accrues daily, but is paid in aggregate monthly.
  ;; When interest is accrued, it is stored with 17 (or so) digits after
  ;;   the decimal, but it is paid in dollar units.  I don't want to round
  ;;   away all of the accuracy of the interest payments, since I accrue
  ;;   it daily.  So, I determine the floor of the amount due, pay that,
  ;;   and leave a residual amount to be paid the next month.  By doing it
  ;;   this way, the shadow money supply holds the (not-absolutely precise)
  ;;   fractional debts, but the logical money supply is always accurate
  ;;   with infinite precision to the dollar.
  ;; This may affect the way I compare total interest payments, over time,
  ;;   with total write-offs, over time, but I don't think it will.
  ;; TODO:  I need to watch that.
  ;; Interest paid by the CRB represents a change in its corporate
  ;;   net worth.  This expense is outside of its role as the guardian of
  ;;   the rule of conservation of money, its public trust, and so must be
  ;;   put into its own corporate checking account (a C1 account) as if
  ;;   it is a client of itself.
  ;; So this payment is a peculiar client-to-client payment mediated by
  ;;   the two banks' own back rooms that manage the public trust.  This
  ;;   payment requires a total of six accounting entries, one of which is
  ;;   redundant and is suppressed.

  ;; Contact the CRB.
  let the-crb ( crb crb-who )

  ;; Contact the bank that holds the C1 assets of the CRB
  let bank-of-crb ( bank ( [bank-who] of the-crb ) )

  ;; The CRB only has an aggregate variable for all of the interest payable
  ;;   on all ER deposits of its client banks.  Only this bank's records
  ;;   indicate the size of the accrued interest associated with this bank.
  ;; Determine the largest integral dollar amount payable.
  let monthly-interest-paid floor( S1-erir-assets )

  ;; Settle the records for the shadow money supply first.
  ;; The bank notes the payment, subtracting it from dues accrued,
  ;;   and leaving a residual.
  set S1-erir-assets ( S1-erir-assets - monthly-interest-paid )
  ;; The CRB decreases its aggregator by the same amount.
  ask the-crb [ set S1-erip-debts ( S1-erip-debts - monthly-interest-paid ) ]

  ;; Now, the CRB has to actually pay the bill with real money.
  ;; A payment is normally a six-entry event.  Two entries are in the
  ;;   check books of the participating agents, and four are back-room
  ;;   changes in banker's assets/debts.  In this case two banks are involved
  ;;   so it gets confusing.  The two banks must each separate their
  ;;   corporate "check books" from their back-room role to protect the
  ;;   public trust.  The corporate assets are C1-assets.  The back-room
  ;;   banking records are L1-assets/L1-debts.
  ;; The payment is noted in this bank's corporate check book.  Entry #1.
  set C1-assets ( C1-assets + monthly-interest-paid )
  ;; And the money enters the logical money supply in the bank's
  ;;   L1 aggregator by its back room staff.  Entry #2.
  set L1-debts ( L1-debts + monthly-interest-paid )
  ;; Assets must follow debts.  Entry #3.
  set L1-assets ( L1-assets + monthly-interest-paid )

  ask the-crb
  [
    ;; The front-room corporate comptroller notes the payment in its check book.
    ;; Entry #4.
    set C1-assets ( C1-assets - monthly-interest-paid )
    ask bank-of-crb
    [
      ;; Entry #5.
      set L1-debts ( L1-debts - monthly-interest-paid )
      ;; Entry #6.  Assets must follow debts.
      set L1-assets ( L1-assets - monthly-interest-paid )
    ]
    ;; The CRB's assets will be quickly transferred to the GCRA.
  ]

  LOG-TO-FILE ( word "  BSvcs: ER interest received --- " monthly-interest-paid )

;; end of f-cbsvcs-bank-paid-monthly-interest-on-er-deposits
end

;;----------------------------------------------------------------------------|
;; A client pays outstanding interest on rr deposits monthly.
to f-cbsvcs-bank-paid-monthly-interest-on-rr-deposits
;; This routine is to be executed by a bank.

  ;; THEORY: Interest on RR deposits is to be paid by the CRB to the bank.
  ;;   It accrues daily, but is paid in aggregate monthly.
  ;; When interest is accrued, it is stored with 17 (or so) digits after
  ;;   the decimal, but it is paid in dollar units.  I don't want to round
  ;;   away all of the accuracy of the interest payments, since I accrue
  ;;   it daily.  So, I determine the floor of the amount due, pay that,
  ;;   and leave a residual amount to be paid the next month.  By doing it
  ;;   this way, the shadow money supply holds the (not-absolutely precise)
  ;;   fractional debts, but the logical money supply is always accurate
  ;;   with infinite precision to the dollar.
  ;; This may affect the way I compare total interest payments, over time,
  ;;   with total write-offs, over time, but I don't think it will.
  ;; TODO:  I need to watch that.
  ;; Interest paid by the CRB represents a change in its corporate
  ;;   net worth.  This expense is outside of its role as the guardian of
  ;;   the rule of conservation of money, its public trust, and so must be
  ;;   put into its own corporate checking account (a C1 account) as if
  ;;   it is a client of itself.
  ;; So this payment is a peculiar client-to-client payment mediated by
  ;;   the two banks' own back rooms that manage the public trust.  This
  ;;   payment requires a total of six accounting entries, one of which is
  ;;   redundant and is suppressed.

  ;; Contact the CRB.
  let the-crb ( crb crb-who )

  ;; Contact the bank that holds the C1 assets of the CRB
  let bank-of-crb ( bank ( [bank-who] of the-crb ) )

  ;; The CRB only has an aggregate variable for all of the interest payable
  ;;   on all RR deposits of its client banks.  Only this bank's records
  ;;   indicate the size of the accrued interest associated with this bank.
  ;; Determine the largest integral dollar amount payable.
  let monthly-interest-paid floor( S1-rrir-assets )

  ;; Settle the records for the shadow money supply first.
  ;; The bank notes the payment, subtracting it from dues accrued,
  ;;   and leaving a residual.
  set S1-rrir-assets ( S1-rrir-assets - monthly-interest-paid )
  ;; The CRB decreases its aggregator by the same amount.
  ask the-crb [ set S1-rrip-debts ( S1-rrip-debts - monthly-interest-paid ) ]

  ;; Now, the CRB has to actually pay the bill with real money.
  ;; A payment is normally a four-entry event.  Two entries are in the
  ;;   bank books of the participating agents, and two are back-room
  ;;   changes in banker's debts.  In this case two banks are involved
  ;;   so it gets confusing.  The two banks must each separate their
  ;;   corporate "bank books" from their back-room role to protect the
  ;;   public trust.  The corporate assets are C1-assets.  The back-room
  ;;   banking records are L1-debts.  It requires six entries.
  ;; The payment is noted in the bank's corporate check book.  Entry #1.
  set C1-assets ( C1-assets + monthly-interest-paid )
  ;; And the money enters the logical money supply in the bank's
  ;;   L1 aggregator by its back room staff.  Entry #2.
  set L1-debts ( L1-debts + monthly-interest-paid )
  ;; And assets follow debts, in the bank back room.  Entry #3.
  set L1-assets ( L1-assets + monthly-interest-paid )

  ask the-crb
  [
    ;; The front-room corporate comptroller notes the payment in its check book.
    ;; Entry #4.
    set C1-assets ( C1-assets - monthly-interest-paid )
    ask bank-of-crb
    [
      ;; Entry #5.
      set L1-debts ( L1-debts - monthly-interest-paid )
      ;; Entry #6.  Assets must follow debts.
      set L1-assets ( L1-assets - monthly-interest-paid )
    ]
    ;; The CRB's assets will be quickly transferred to the GCRA.
  ]

  LOG-TO-FILE ( word "  BSvcs: RR interest received --- " monthly-interest-paid )

;; end of f-cbsvcs-bank-paid-monthly-interest-on-rr-deposits
end

;; END OF -CBSVCS- SUBSECTION.
```

```
;;-----------------------------------------------------------------------------|
;; START OF THE -BTPFS- SUBSECTION
;;-----------------------------------------------------------------------------|
;; THEORY: This is a special part of the banking services section which is not
;;   really about banking services, so much, as it is about flows of money
;;   from the banking sector to the non-banking sector.  In general money flows
;;   to the banking sector through interest on ER and RR deposits, and through
;;   interest on L1 loans.  It flows from the banking sector through
;;   bankruptcies and interest on savings deposits.  Bankruptcies are a very
;;   difficult thing to manage.  They cause great instability, and public
;;   policy governing bankruptcies is a key source of bias in all wealth
;;   distributions.  In particular, the debts of failed agents must be covered
;;   by one bank or many banks, and assets for replacement agents must be
;;   gathered from many agents.  The way this is done may bias the wealth
;;   distributions of both prsns and banks.
;;
;; The routines that start with f-btpfs-xxx are "banks-to-prsn-flows" special
;;   routines that can be toggled on to provide additional flows from the
;;   banking sector to the non-banking sector, in addition to the
;;   default "bankruptcies" channel.

;;-----------------------------------------------------------------------------|
;; Government collects a tax from banks, distributes to prsns.
to f-btpfs-government-special-monthly-transfer
;; This routine is to be executed by the observer.

  ;; THIS ROUTINE IS PART OF THE BANKS-TO-PRSNS-FLOWS (-btpfs-) REGIME.
  ;; As such, it is an adjunct to the standard -bnkrpt- regime.

  ;; THEORY: In basic mode there is a flow of money from prsns to banks, and
  ;;   the only means for money to return to the non-financial sector is
  ;;   via over-extended loans causing prsns to go bankrupt, and the bank
  ;;   must cover the costs.
  ;; This causes a problem because I then need to find funds to re-constitute
  ;;   the bankrupt prsn as a prsn of average net worth, and there is nowhere
  ;;   to obtain the cash.  So, this routine is one way in which some cash
  ;;   can be returned to the non-banking sector.
  ;; It is controlled by the switch in the User Interface
  ;;   gb-btpfs-monthly-taxes.

  ;; The government collects a tax from each bank removing all remaining
  ;;   C1 assets and distributes it directly and evenly to all prsns.
  ;;   Excess goes into the GCRA.

  if( gb-btpfs-monthly-taxes = true )
  [
    ask gcras
    [
      ;; Identify the bank of the GCRA.
      ;; The GCRA is not a bank.  It keeps its accounts in a commercial bank.
      let gcra-bank ( bank bank-who )

      let taxes-due 0        ;; Initialize a working variable.
      let all-taxes-paid 0   ;; initialize an aggregate to collect all taxes paid.

      ;; This routine proceeds in two steps:
      ;;   STEP 1 - all banks are stripped of all C1 assets, going into the GCRA.
      ;;   STEP 2 - the proceeds are distributed evenly to all prsns.

      ;; STEP 1 - COLLECT THE TAXES.
      ;; This functions like a prsn-to-prsn check, and requires six entries.
      ;;   Two in client's check books.  Four in bank back room records.
```

```
ask banks
[
  LOG-TO-FILE ( word "BANK " who " PAYS TAXES" )
  LOG-TO-FILE ( word "  Bank C1-assets -------------------- " C1-assets )
  set taxes-due C1-assets

  ;; Taxes are paid by bank-to-bank check.
  ;; Remove taxes from bank's bankbook.  Entry #1.
  set C1-assets ( C1-assets - taxes-due )
  ;; Remove the taxes from the bank's checking account. Entry #2.
  set L1-debts ( L1-debts - taxes-due )
  ;; Assets follow debts.  Entry #3.
  set L1-assets ( L1-assets - taxes-due )
  ;; Record the amount as paid, for later entry to GCRA bankbook.
  ;; At this point the net change in prsn-bank is zero.
  set all-taxes-paid ( all-taxes-paid + taxes-due )
  LOG-TO-FILE ( word "  Taxes paid ------------------------ " taxes-due )
  LOG-TO-FILE ( word "  Bank C1 assets after payment ------ " C1-assets )
] ;; end of ask banks

LOG-TO-FILE ( word "  GCRA L1 assets before collection -- " L1-assets )
LOG-TO-FILE ( word "  Total of all taxes collected ------ " all-taxes-paid )

;; Government adjusts its own bankbook.  Entry #4.
set L1-assets ( L1-assets + all-taxes-paid )
;; Add the money to the gov't checking account.  Entry #5.
ask gcra-bank [ set L1-debts ( L1-debts + all-taxes-paid ) ]
;; Assets follow debts.  Entry #6.
ask gcra-bank [ set L1-assets ( L1-assets + all-taxes-paid ) ]
;; At this point the net change in gcra-bank is zero.
LOG-TO-FILE ( word "  GCRA L1 assets after collection --- " L1-assets )

;; STEP 2 - PAY TO PRSNS.
;; Determine the payment to each prsn.
let payout floor( all-taxes-paid / g-no-of-prsns )
;; So, due to the use of 'floor' the entire payout will be less than
;;   or equal to all-taxes-paid.  The residual will remain in the GCRA.

;; Initialize an aggregator.
let total-dole-paid 0

;; This functions like a prsn-to-prsn check, and requires six entries.
;;   Two in client's check books.  Four in bank back room records.
ask prsns
[
  ;; Contact prsn's bank
  let prsns-bank ( bank bank-who )

  LOG-TO-FILE ( word "Prsn " who " RECEIVES DOLE" )
  LOG-TO-FILE ( word "  Prsn L1-assets before dole -------- " L1-assets )

  ;; Dole is paid by bank-to-bank check.
  ;; Add dole to prsn's bankbook.  Entry #1.
  set L1-assets ( L1-assets + payout )
  ;; Adjust checking account. Entry #2.
  ask prsns-bank [ set L1-debts ( L1-debts + payout ) ]
  ;; Assets follow debts.  Entry #3.
  ask prsns-bank [ set L1-assets ( L1-assets + payout ) ]
  ;; Record the amount as paid, for later entry to GCRA bankbook.
  ;; At this point the net change in prsn-bank is zero.
  set total-dole-paid ( total-dole-paid + payout )
  LOG-TO-FILE ( word "  Taxes paid ------------------------ " taxes-due )
```

```
        LOG-TO-FILE ( word "  Prsn L1 assets after payment ------ " L1-assets )
      ] ;; end of ask banks

      LOG-TO-FILE ( word "  GCRA L1 assets before payments ---- " L1-assets )
      LOG-TO-FILE ( word "  Total of all dole paid ------------ " total-dole-paid )

      ;; Government adjusts its own bankbook.  Entry #4.
      set L1-assets ( L1-assets - total-dole-paid )
      ;; Add the money to the gov't checking account.  Entry #5.
      ask gcra-bank [ set L1-debts ( L1-debts - total-dole-paid ) ]
      ;; Assets follow debts.  Entry #6.
      ask gcra-bank [ set L1-assets ( L1-assets - total-dole-paid ) ]
      ;; At this point the net change in gcra-bank is zero.
      LOG-TO-FILE ( word "  GCRA L1 assets after payments ----- " L1-assets )
    ] ;; end of ask gcras
  ] ;; end of if ( gb-btpfs-monthly-taxes = true )
;; end of f-btpfs-government-special-monthly-transfer
end

;;----------------------------------------------------------------------------|
;; Banks buy using checks.
to f-btpfs-banks-buy-using-checks
;; This routine is to be executed by the observer.

  ;; THIS ROUTINE IS PART OF THE BANKS-TO-PRSNS-FLOWS (-btpfs-) REGIME.
  ;; As such, it is an adjunct to the standard -bnkrpt- regime.

  ;; THEORY: In basic mode there is a flow of money from prsns to banks, and
  ;;   the only means for money to return to the non-financial sector is
  ;;   via over-extended loans causing prsns to go bankrupt, and the bank
  ;;   must cover the costs.
  ;; This causes a problem because I then need to find funds to re-fashion
  ;;   the bankrupt prsn as a prsn of average net worth, and there is nowhere
  ;;   to obtain the cash.  So, this routine is one way in which some cash
  ;;   can be returned to the non-banking sector.
  ;; It is controlled by the switch in the User Interface
  ;;   gb-btpfs-daily-purchases.

  ;; Each prsn canvasses its own bank for a $1 purchase per prsn per tick,
  ;;   coming out of its corporate funds, unless those C1 funds are drained.
  ;;   You might think of this as administrative costs for building, personnel
  ;;   and supplies.

  if ( gb-btpfs-daily-purchases = true )
  [
    ;; Initialize a grand aggregator.
    let grand-total-spent 0

    LOG-TO-FILE ( word "  " )
    LOG-TO-FILE ( word "Do-buy-sell: Banks purchase daily supplies" )

    ask prsns
    [
      let amount-to-spend 1

      ;; Contact the prsn's bank so money can be sent.
      let prsns-bank ( bank bank-who )

      ;; Payment by inter-bank check requires six entries.

      let go-flag ( [C1-assets] of prsns-bank )
      if( go-flag > 0 )
      [
        ;; Bank records the aggregate of all payments in its own corporate
        ;;   check book.  Entry #1.
        ask prsns-bank [ set C1-assets ( C1-assets - amount-to-spend ) ]
        ;; The bank settles all check in it back-room records.  Entries #2 and #3.
        ;; ask prsns-bank [ set L1-assets ( L1-assets - amount-to-spend ) ]
        ;; ask prsns-bank [ set L1-debts ( L1-debts - amount-to-spend ) ]

        ;; Prsn receives the money and enters it in their own check book.  Entry #4.
        set L1-assets ( L1-assets + amount-to-spend )
        ;; Their bank records the check with two entries - #5 and #6.
        ;; ask prsns-bank [ set L1-assets ( L1-assets + amount-to-spend ) ]
        ;; ask prsns-bank [ set L1-debts ( L1-debts + amount-to-spend ) ]

        ;; Increment the aggregator.
        set grand-total-spent ( grand-total-spent + amount-to-spend )

        ;; The private net worth of the bank has been reduced by total-spent.
        ;; The private net worth of each prsn has increased by amount-to-spend.
        ;; The net worth of public funds in trust (in the bank's back rooms)
        ;;   has not changed.
      ] ;; end of if( go-flag > 0 )
    ] ;; end ask prsns
    LOG-TO-FILE ( word "  All banks have spent this tick -- " grand-total-spent )
  ] ;; end if ( gb-btpfs-daily-purchases = true )
;; end of f-btpfs-banks-buy-using-checks
end

;;-----------------------------------------------------------------------------|
;; SECTION E – DRAWING AND MAINTENANCE PROCEDURE(S)
;;-----------------------------------------------------------------------------|

;;-----------------------------------------------------------------------------|
;; Dump all of the data to debug file, or to control centre.
to f-dump-all-agent-data
  ;; This routine is to be executed by the observer.

  ;; Dump the GCRA data
  f-dump-gcras-data
  f-dump-crbS-data
  f-dump-bankS-data
  f-dump-prsnS-data

  ;; TODO: Corps not implemented yet.
  ;; f-dump-corpS-data

  ;; End of f-dump-all-agent-data
end

;;-----------------------------------------------------------------------------|
;; Dump all GCRA data to debug file, or to control centre.
to f-dump-gcras-data
  ;; This routine is to be executed by the observer.

  ;; Dump the GCRA data
  ask gcras
  [
    f-dump-gcra-data
  ]

  ;; End of f-dump-gcras-data
end
```

```
;;-----------------------------------------------------------------------------|
;; Dump the data of one calling GCRA to debug file, or to control centre.
to f-dump-gcra-data
  ;; This routine is to be executed by the GCRA.

  LOG-TO-FILE ( word "  " )
  LOG-TO-FILE ( word "DUMP GCRA who# <<< " who " >>>" )
  LOG-TO-FILE ( word "bank-who ------------------ " bank-who )
  LOG-TO-FILE ( word "L1-assets ----------------- " L1-assets )
  ;; LOG-TO-FILE ( word "L1-debts ------------------ " L1-debts )
  LOG-TO-FILE ( word "L1-loan-debts ------------- " L1-loan-debts )
  LOG-TO-FILE ( word "S1-L1ip-debts ------------- " S1-L1ip-debts )
  ;; ss LOG-TO-FILE ( word "L3-debts ------------------ " L3-debts )
  ;; ss LOG-TO-FILE ( word "S1-L3ip-debts ------------- " S1-L3ip-debts )
  LOG-TO-FILE ( word "ttl-P0-assets ------------- " ttl-P0-assets )
  LOG-TO-FILE ( word "ttl-publ-assets ----------- " ttl-publ-assets )
  LOG-TO-FILE ( word "ttl-publ-debts ------------ " ttl-publ-debts )
  LOG-TO-FILE ( word "ttl-priv-assets ----------- " ttl-priv-assets )
  LOG-TO-FILE ( word "ttl-priv-debts ------------ " ttl-priv-debts )
  LOG-TO-FILE ( word "net-worth-publ ------------ " net-worth-publ )
  LOG-TO-FILE ( word "net-worth-priv ------------ " net-worth-priv )

  ;; End of f-dump-gcra-data
end

;;-----------------------------------------------------------------------------|
;; Dump all the CRB data to debug file, or to control centre.
to f-dump-crbs-data
  ;; This routine is to be executed by the observer.

  ;; Dump the CRB data
  ask crbs
  [
    f-dump-crb-data
  ]

  ;; End of f-dump-crbs-data
end

;;-----------------------------------------------------------------------------|
;; Dump the data of the calling CRB to debug file, or to control centre.
to f-dump-crb-data
  ;; This routine is to be executed by the CRB.

  LOG-TO-FILE ( word "  " )
  LOG-TO-FILE ( word "DUMP CRB who# <<< " who " >>>" )
  LOG-TO-FILE ( word "L0-assets ----------------- " L0-assets )
  LOG-TO-FILE ( word "P0-assets ----------------- " P0-assets )
  LOG-TO-FILE ( word "L0-debts ------------------ " L0-debts )
  LOG-TO-FILE ( word "P0-debts ------------------ " P0-debts )
  LOG-TO-FILE ( word "P0-rr-assets -------------- " P0-rr-assets )
  LOG-TO-FILE ( word "P0-er-assets -------------- " P0-er-assets )
  LOG-TO-FILE ( word "S1-rrip-debts ------------- " S1-rrip-debts )
  LOG-TO-FILE ( word "S1-erip-debts ------------- " S1-erip-debts )
  LOG-TO-FILE ( word "C1-assets ----------------- " C1-assets )
  ;; xx LOG-TO-FILE ( word "c2-assets ----------------- " c2-assets )
  LOG-TO-FILE ( word "ttl-P0-assets ------------- " ttl-P0-assets )
  LOG-TO-FILE ( word "ttl-publ-assets ----------- " ttl-publ-assets )
  LOG-TO-FILE ( word "ttl-publ-debts ------------ " ttl-publ-debts )
  LOG-TO-FILE ( word "ttl-priv-assets ----------- " ttl-priv-assets )
  LOG-TO-FILE ( word "ttl-priv-debts ------------ " ttl-priv-debts )
  LOG-TO-FILE ( word "net-worth-publ ------------ " net-worth-publ )
  LOG-TO-FILE ( word "net-worth-priv ------------ " net-worth-priv )

  ;; End of f-dump-crb-data
end

;;----------------------------------------------------------------------------|
;; Dump all bank data to debug file, or to control centre.
to f-dump-banks-data
  ;; This routine is to be executed by the observer.

  ;; Dump the bank data
  ask banks
  [
    f-dump-bank-data
  ]

  ;; End of f-dump-banks-data
end

;;----------------------------------------------------------------------------|
;; Dump the data of the calling bank to debug file, or to control centre.
to f-dump-bank-data
  ;; This routine is to be executed by a bank.

  LOG-TO-FILE ( word "  " )
  LOG-TO-FILE ( word "DUMP BANK who# <<< " who " >>>" )
  LOG-TO-FILE ( word "b-bank-can-make-loans ----- " b-bank-can-make-loans )
  LOG-TO-FILE ( word "b-bank-is-bankrupt -------- " b-bank-is-bankrupt )
  LOG-TO-FILE ( word "L1-assets ----------------- " L1-assets )
  LOG-TO-FILE ( word "L1-loan-assets ------------ " L1-loan-assets )
  LOG-TO-FILE ( word "L1-debts ------------------ " L1-debts )
  LOG-TO-FILE ( word "S1-L1ir-assets ------------ " S1-L1ir-assets )
  LOG-TO-FILE ( word "L2-debts ------------------ " L2-debts )
  LOG-TO-FILE ( word "S1-L2ip-debts ------------- " S1-L2ip-debts )
  ;; ss LOG-TO-FILE ( word "L3-assets ----------------- " L3-assets )
  LOG-TO-FILE ( word "P0-vc-assets -------------- " P0-vc-assets )
  LOG-TO-FILE ( word "P0-rr-assets -------------- " P0-rr-assets )
  LOG-TO-FILE ( word "P0-er-assets -------------- " P0-er-assets )
  LOG-TO-FILE ( word " " )
  LOG-TO-FILE ( word "no-of-prsn-clients -------- " no-of-prsn-clients )
  LOG-TO-FILE ( word "no-of-corp-clients -------- " no-of-corp-clients )
  LOG-TO-FILE ( word "no-of-gcra-clients -------- " no-of-gcra-clients )
  LOG-TO-FILE ( word "no-of-crb-clients --------- " no-of-crb-clients )
  LOG-TO-FILE ( word "S1-rrir-assets ------------ " S1-rrir-assets )
  LOG-TO-FILE ( word "S1-erir-assets ------------ " S1-erir-assets )
  LOG-TO-FILE ( word "C1-assets ----------------- " C1-assets )
  ;; xx LOG-TO-FILE ( word "c2-assets ----------------- " c2-assets )
  LOG-TO-FILE ( word "ttl-P0-assets ------------- " ttl-P0-assets )
  LOG-TO-FILE ( word "ttl-publ-assets ----------- " ttl-publ-assets )
  LOG-TO-FILE ( word "ttl-publ-debts ------------ " ttl-publ-debts )
  LOG-TO-FILE ( word "ttl-priv-assets ----------- " ttl-priv-assets )
  LOG-TO-FILE ( word "ttl-priv-debts ------------ " ttl-priv-debts )
  LOG-TO-FILE ( word "net-worth-publ ------------ " net-worth-publ )
  LOG-TO-FILE ( word "net-worth-priv ------------ " net-worth-priv )

  ;; End of f-dump-bank-data
end

;;----------------------------------------------------------------------------|
;; Dump all prns data to debug file, or to control centre.
to f-dump-prsns-data
  ;; This routine is to be executed by the observer.

  ;; Dump the prsn data
  ask prsns
  [
    f-dump-prsn-data
  ]

  ;; End of f-dump-prsns-data
end

;;----------------------------------------------------------------------------|
;; Dump all one prns's data to debug file, or to control centre.
to f-dump-prsn-data
  ;; This routine is to be executed by a prsn.

  LOG-TO-FILE ( word "  " )
  LOG-TO-FILE ( word "DUMP PRSN who# <<< " who " >>>" )
  LOG-TO-FILE ( word "b-prsn-is-bankrupt -------- " b-prsn-is-bankrupt )
  LOG-TO-FILE ( word "Bank-who ------------------ " bank-who )
  LOG-TO-FILE ( word "P0-assets ----------------- " P0-assets )
  LOG-TO-FILE ( word "L0-assets ----------------- " L0-assets )
  LOG-TO-FILE ( word "L1-assets ----------------- " L1-assets )
  LOG-TO-FILE ( word "L1-loan-debts ------------- " L1-loan-debts )
  LOG-TO-FILE ( word "S1-L1ip-debts ------------- " S1-L1ip-debts )
  LOG-TO-FILE ( word "30day payables total ------ " S1-30day-total-debts )
  LOG-TO-FILE ( word "30day receivables total --- " S1-30day-total-assets )
  foreach payables-30day
  [
    LOG-TO-FILE ?
  ]
  LOG-TO-FILE ( word "L2-assets ----------------- " L2-assets )
  LOG-TO-FILE ( word "S1-L2ir-assets ------------ " S1-L2ir-assets )
  ;; ss LOG-TO-FILE ( word "L3-corpwho ---------------- " L3-corpwho )
  ;; ss LOG-TO-FILE ( word "L3-assets ----------------- " L3-assets )
  ;; ss LOG-TO-FILE ( word "S1-L3ir-assets ---- " S1-L3ir-assets )
  ;; ss LOG-TO-FILE ( word "L4-corpwho ---------------- " L4-corpwho )
  ;; ss LOG-TO-FILE ( word "L4-assets ----------------- " L4-assets )
  ;; ss LOG-TO-FILE ( word "L4-dividend-receivable ---- " L4-dividend-receivable )
  LOG-TO-FILE ( word "ttl-P0-assets ------------- " ttl-P0-assets )
  LOG-TO-FILE ( word "ttl-publ-assets ----------- " ttl-publ-assets )
  LOG-TO-FILE ( word "ttl-publ-debts ------------ " ttl-publ-debts )
  LOG-TO-FILE ( word "ttl-priv-assets ----------- " ttl-priv-assets )
  LOG-TO-FILE ( word "ttl-priv-debts ------------ " ttl-priv-debts )
  LOG-TO-FILE ( word "net-worth-publ ------------ " net-worth-publ )
  LOG-TO-FILE ( word "net-worth-priv ------------ " net-worth-priv )

  ;; End of f-dump-prsn-data
end

;;----------------------------------------------------------------------------|
;; Dump all corp data to debug file, or to control centre.
to f-dump-corps-data
  ;; This routine is to be executed by the observer.

  ;; Dump the corp data
  ask corps
  [
    f-dump-corp-data
  ]

  ;; End of f-dump-corps-data
```

```
end

;;-----------------------------------------------------------------------------|
;; Dump all one corp's data to debug file, or to control centre.
to f-dump-corp-data
  ;; This routine is to be executed by a corp.

  LOG-TO-FILE ( word "  " )
  LOG-TO-FILE ( word "DUMP CORP who# <<< " who " >>>" )
  LOG-TO-FILE ( word "b-corp-is-bankrupt -------- " b-corp-is-bankrupt )
  LOG-TO-FILE ( word "Bank-who ------------------ " bank-who )
  LOG-TO-FILE ( word "P0-assets ----------------- " P0-assets )
  LOG-TO-FILE ( word "L0-assets ----------------- " L0-assets )
  LOG-TO-FILE ( word "L1-assets ----------------- " L1-assets )
  LOG-TO-FILE ( word "L1-debts ------------------ " L1-debts )
  LOG-TO-FILE ( word "L1-loan-debts ------------- " L1-loan-debts )
  LOG-TO-FILE ( word "S1-L1ip-debts ------------- " S1-L1ip-debts )
  LOG-TO-FILE ( word "30day payables total ------ " S1-30day-total-debts )
  LOG-TO-FILE ( word "30day receivables total --- " S1-30day-total-assets )
  foreach payables-30day
  [
    LOG-TO-FILE ?
  ]
  LOG-TO-FILE ( word "L2-assets ----------------- " L2-assets )
  LOG-TO-FILE ( word "S1-L2ir-assets ------------ " S1-L2ir-assets )
  ;; ss LOG-TO-FILE ( word "no-of-bond-clients -------- " no-of-bond-clients )
  ;; ss LOG-TO-FILE ( word "L3-assets ----------------- " L3-assets )
  ;; ss LOG-TO-FILE ( word "L3-debts ------------------ " L3-debts )
  ;; ss LOG-TO-FILE ( word "S1-L3ip-debts ------------- " S1-L3ip-debts )
  ;; ss LOG-TO-FILE ( word "no-of-stock-clients ------- " no-of-stock-clients )
  ;; ss LOG-TO-FILE ( word "L4-assets ----------------- " L4-assets )
  ;; ss LOG-TO-FILE ( word "L4-debts ------------------ " L4-debts )
  ;; ss LOG-TO-FILE ( word "S1-L4dp-debts ------- " S1-L4dp-debts )
  LOG-TO-FILE ( word " " )
  LOG-TO-FILE ( word "ttl-P0-assets ------------- " ttl-P0-assets )
  LOG-TO-FILE ( word "ttl-publ-assets ------- " ttl-publ-assets )
  LOG-TO-FILE ( word "ttl-publ-debts -- " ttl-publ-debts )
  LOG-TO-FILE ( word "ttl-priv-assets ------ " ttl-priv-assets )
  LOG-TO-FILE ( word "ttl-priv-debts - " ttl-priv-debts )
  LOG-TO-FILE ( word "net-worth-publ ---------- " net-worth-publ )
  LOG-TO-FILE ( word "net-worth-priv --------- " net-worth-priv )

  ;; End of f-dump-corp-data
end

;;-----------------------------------------------------------------------------|
;; Update the values of global aggregate numbers.
to f-update-aggregates
  ;; This routine is to be executed by the observer.

  ;; Although this is a display-only routine, it may implicitly call the PRNG and
  ;;   so may have an effect on the trajectory of the model.  In a standard 'go'
  ;;   run it is called only once per tick, before graphs are updated.  If you
  ;;   use the one-step debug buttons, it is called once after each step, so
  ;;   debug runs that use those buttons will not replicate a real run.

  ;; Re-calculate all net worth statements.
  f-compute-each-net-worth

  ;; Update all aggregates.
  ;; In the following I use "debts" to mean "liabilities".
  ;; Money supplies
  set g-msi-ttl-assets (sum [msi-assets] of turtles) ;; Money supply I, Physical
money supply.
  set g-msii-ttl-assets (sum [msii-assets] of turtles) ;; Money supply II, Logical
money supply.
  set g-msiii-ttl-assets (sum [msiii-assets] of turtles) ;; Money supply III, Shadow
money supply.
  set g-msi-ttl-debts (sum [msi-debts] of turtles) ;; Money supply I, Physical money
supply.
  set g-msii-ttl-debts (sum [msii-debts] of turtles) ;; Money supply II, Logical
money supply.
  set g-msiii-ttl-debts (sum [msiii-debts] of turtles) ;; Money supply III, Shadow
money supply.
  set g-msi-net ( g-msi-ttl-assets - g-msi-ttl-debts )
  set g-msii-net ( g-msii-ttl-assets - g-msii-ttl-debts )
  set g-msiii-net ( g-msiii-ttl-assets - g-msiii-ttl-debts )

  ;; Money Categories - by money supply.
  ;; MS-I - The money base - Physical money supply.
  set g-msi-prsn-P0-cash (sum [P0-assets] of prsns) ;; cash in circulation - assets
  set g-msi-corp-P0-cash (sum [P0-assets] of corps) ;; cash in circulation - assets
  set g-msi-bank-vc (sum [P0-vc-assets] of banks) ;; bank vault cash - assets
  set g-msi-bank-rr-assets (sum [P0-rr-assets] of banks) ;; bank required reserves -
debts
  set g-msi-bank-er-assets (sum [P0-er-assets] of banks) ;; bank excess reserves -
debts
  set g-msi-bank-rr-debts (sum [P0-rr-debts] of banks) ;; bank required reserves -
debts
  set g-msi-bank-er-debts (sum [P0-er-debts] of banks) ;; bank excess reserves -
debts
  set g-msi-crb-L0-assets (sum [L0-assets] of crbs) ;; money base endowment
  set g-msi-crb-P0-assets (sum [P0-assets] of crbs) ;; money base endowment
  set g-msi-crb-L0-debts (sum [L0-debts] of crbs) ;; money base endowment
  set g-msi-crb-P0-debts (sum [P0-debts] of crbs) ;; money base endowment
  set g-msi-crb-rr (sum [P0-rr-assets] of crbs) ;; CRB required reserves - assets
  set g-msi-crb-er (sum [P0-er-assets] of crbs) ;; CRB excess reserves - assets

  ;; MS-II - The logical money supply.
  set g-msii-prsn-L0-cash (sum [L0-assets] of prsns) ;; cash in circulation,
overlaps with MS-I.
  set g-msii-corp-L0-cash (sum [L0-assets] of corps) ;; cash in circulation,
overlaps with MS-I.
  set g-msii-crb-C1-assets (sum [C1-assets] of crbs) ;; privatecorp level assets
  ;; xx set g-msii-crb-c2-assets (sum [c2-assets] of crbs) ;; private corp level
assets

  set g-msii-gcra-L1-assets (sum [L1-assets] of gcras) ;; govt checking assets
  ;; set g-msii-gcra-L1-debts (sum [L1-debts] of gcras) ;; govt checking debts
  set g-msii-gcra-L1-loan-debts (sum [L1-loan-debts] of gcras) ;; govt loan debts
  ;; xx set g-msii-gcra-L2-assets (sum [L2-assets] of gcras) ;; govt savings assets
  ;; ss set g-msii-gcra-L3-debts (sum [L3-debts] of gcras) ;; govt bond debts

  set g-msii-bank-L1-assets (sum [L1-assets] of banks) ;; bank checking assets
  set g-msii-bank-L1-loan-assets (sum [L1-loan-assets] of banks) ;; bank checking
assets
  set g-msii-bank-L1-debts (sum [L1-debts] of banks) ;; bank checking debts
  set g-msii-bank-L2-assets (sum [L2-assets] of banks) ;; bank savings assets
  set g-msii-bank-L2-debts (sum [L2-debts] of banks) ;; bank savings debts
  ;; ss set g-msii-bank-L3-assets (sum [L3-assets] of banks) ;; bank bond assets
  set g-msii-bank-C1-assets (sum [C1-assets] of banks) ;; private L1 checking assets
  ;; xx set g-msii-bank-c2-assets (sum [C1-assets] of banks) ;; private L2 savings
assets
```

```
  set g-msii-prsn-L1-assets (sum [L1-assets] of prsns) ;; prsn checking assets
  set g-msii-prsn-L1-loan-debts (sum [L1-loan-debts] of prsns) ;; prsn loan debts
  set g-msii-prsn-L2-assets (sum [L2-assets] of prsns) ;; prsn savings assets
  ;; ss set g-msii-prsn-L3-assets (sum [L3-assets] of prsns) ;; prsn bond assets
  ;; ss set g-msii-prsn-L4-assets (sum [L4-assets] of prsns) ;; prsn bond assets

  set g-msii-corp-L1-assets (sum [L1-assets] of corps) ;; corp checking assets
  set g-msii-corp-L1-loan-debts (sum [L1-loan-debts] of corps) ;; corp loan debts
  set g-msii-corp-L2-assets (sum [L2-assets] of corps) ;; corp savings assets
  ;; ss set g-msii-corp-L3-assets (sum [L3-assets] of corps) ;; corp bond assets
  ;; ss set g-msii-corp-L3-debts (sum [L3-debts] of corps) ;; corp bond debts
  ;; ss set g-msii-corp-L4-assets (sum [L4-assets] of corps) ;; corp bond assets
  ;; ss set g-msii-corp-L4-debts (sum [L4-debts] of corps) ;; corp bond debts

  ;; MS-III - The shadow money supply.
  set g-msiii-crb-S1-rrip-debts (sum [S1-rrip-debts] of crbs) ;; CRB interest
payable on rr - debts
  set g-msiii-crb-S1-erip-debts (sum [S1-erip-debts] of crbs) ;; CRB interest
payable on er - debts
  set g-msiii-gcra-S1-L1ip-debts (sum [S1-L1ip-debts] of gcras) ;; govt interest
payable on loan - debts
  ;; ss set g-msiii-gcra-S1-L3ip-debts (sum [S1-L3ip-debts] of gcras) ;; govt
interest payable on bonds - debts
  set g-msiii-bank-S1-L1ir-assets (sum [S1-L1ir-assets] of banks) ;; bank interest
receivable on loans - assets
  set g-msiii-bank-S1-L2ip-debts  (sum [S1-L2ip-debts] of banks) ;; bank interest
payable on savings - debts
  set g-msiii-bank-S1-rrir-assets (sum [S1-rrir-assets] of banks) ;; bank interest
receivable on rr - assets
  set g-msiii-bank-S1-erir-assets (sum [S1-erir-assets] of banks) ;; bank interest
receivable on er - assets
  set g-msiii-prsn-S1-L1ip-debts  (sum [S1-L1ip-debts] of prsns) ;; prsn total 30day
payables - debts
  set g-msiii-prsn-S1-L1tp-debts  (sum [S1-30day-total-debts] of prsns) ;; prsn
total 30day payables - debts
  set g-msiii-prsn-S1-L1tr-assets (sum [S1-30day-total-assets] of prsns) ;; prsn
total 30day receivables - assets
  set g-msiii-prsn-S1-L2ir-assets (sum [S1-L2ir-assets] of prsns) ;; prsn interest
receivable on savings - assets
  ;; ss set g-msiii-prsn-S1-L3ir-assets (sum [S1-L3ir-assets] of prsns) ;; prsn
interest receivable on bonds - assets
  ;; ss set g-msiii-prsn-S1-L4dr-assets (sum [L4-dividend-receivable] of prsns) ;;
prsn dividend receivable on stocks - assets
  set g-msiii-corp-S1-L1tp-debts (sum [S1-30day-total-debts] of corps) ;; corp total
30day payables - debts
  set g-msiii-corp-S1-L1tr-assets (sum [S1-30day-total-assets] of corps) ;; corp
total 30day receivables - assets
  set g-msiii-corp-S1-L2ir-assets (sum [S1-L2ir-assets] of corps) ;; corp interest
receivable on savings - assets
  ;; ss set g-msiii-corp-S1-L3ip-assets (sum [S1-L3ip-debts] of corps) ;; corp
interest payable on bonds - debts
  ;; ss set g-msiii-corp-S1-L4dp-assets (sum [S1-L4dp-debts] of corps) ;; corp
dividend payable on stocks - debts

  ;; Public funds in trust vs Private funds
  set g-crb-P0-assets (sum [ttl-P0-assets] of crbs) ;; In public trust
  set g-crb-publ-assets (sum [ttl-publ-assets] of crbs) ;; In public trust
  set g-crb-priv-assets (sum [ttl-priv-assets] of crbs) ;; Profit/Loss related
  set g-crb-publ-debts (sum [ttl-publ-debts] of crbs) ;; In public trust
  set g-crb-priv-debts (sum [ttl-priv-debts] of crbs) ;; Profit/Loss related
  set g-crb-publ-net-worth (sum [net-worth-publ] of crbs) ;; In public trust
  set g-crb-priv-net-worth (sum [net-worth-priv] of crbs) ;; Profit/Loss related
```

```
  set g-gcra-P0-assets (sum [ttl-P0-assets] of gcras) ;; In public trust
  set g-gcra-publ-assets (sum [ttl-publ-assets] of gcras) ;; In public trust
  set g-gcra-priv-assets (sum [ttl-priv-assets] of gcras) ;; Profit/Loss related
  set g-gcra-publ-debts (sum [ttl-publ-debts] of gcras) ;; In public trust
  set g-gcra-priv-debts (sum [ttl-priv-debts] of gcras) ;; Profit/Loss related
  set g-gcra-publ-net-worth (sum [net-worth-publ] of gcras) ;; In public trust
  set g-gcra-priv-net-worth (sum [net-worth-priv] of gcras) ;; Profit/Loss related

  set g-bank-P0-assets (sum [ttl-P0-assets] of banks) ;; In public trust
  set g-bank-publ-assets (sum [ttl-publ-assets] of banks) ;; In public trust
  set g-bank-priv-assets (sum [ttl-priv-assets] of banks) ;; Profit/Loss related
  set g-bank-publ-debts (sum [ttl-publ-debts] of banks) ;; In public trust
  set g-bank-priv-debts (sum [ttl-priv-debts] of banks) ;; Profit/Loss related
  set g-bank-publ-net-worth (sum [net-worth-publ] of banks) ;; In public trust
  set g-bank-priv-net-worth (sum [net-worth-priv] of banks) ;; Profit/Loss related

  set g-prsn-P0-assets (sum [ttl-P0-assets] of prsns) ;; In public trust
  set g-prsn-publ-assets (sum [ttl-publ-assets] of prsns) ;; In public trust
  set g-prsn-priv-assets (sum [ttl-priv-assets] of prsns) ;; Profit/Loss related
  set g-prsn-publ-debts (sum [ttl-publ-debts] of prsns) ;; In public trust
  set g-prsn-priv-debts (sum [ttl-priv-debts] of prsns) ;; Profit/Loss related
  set g-prsn-publ-net-worth (sum [net-worth-publ] of prsns) ;; In public trust
  set g-prsn-priv-net-worth (sum [net-worth-priv] of prsns) ;; Profit/Loss related

  set g-corp-P0-assets (sum [ttl-P0-assets] of corps) ;; In public trust
  set g-corp-publ-assets (sum [ttl-publ-assets] of corps) ;; In public trust
  set g-corp-priv-assets (sum [ttl-priv-assets] of corps) ;; Profit/Loss related
  set g-corp-publ-debts (sum [ttl-publ-debts] of corps) ;; In public trust
  set g-corp-priv-debts (sum [ttl-priv-debts] of corps) ;; Profit/Loss related
  set g-corp-publ-net-worth (sum [net-worth-publ] of corps) ;; In public trust
  set g-corp-priv-net-worth (sum [net-worth-priv] of corps) ;; Profit/Loss related

;;-----------------------------------------------------------------------------|
  ;; To ensure that the PRNG is called whether or not plots are displayed, the
  ;;   calculations needed for the histogram plots which invoke the PRNG
  ;;   implicitly should be carried out here where they will happen every tick.

;;-----------------------------------------------------------------------------|
  ;; Setup for Histograms "Net Worth of Agents" in Panel 01 and
  ;;                      "Net Worth of Prsns and Banks" in Panel 05.
  let prsn-nws ( [net-worth-priv] of prsns ) ;; a list
  let bank-nws ( [net-worth-priv] of banks ) ;; a list
  set g-agents-nw-xaxis-min ( min sentence prsn-nws bank-nws ) ;; a number
  set g-agents-nw-xaxis-min ( 1000 * floor( g-agents-nw-xaxis-min / 1000 ) )
  if( g-agents-nw-xaxis-min > 0 ) [ set g-agents-nw-xaxis-min 0 ]

  set g-agents-nw-xaxis-max ( max sentence prsn-nws bank-nws ) ;; a number
  set g-agents-nw-xaxis-max ( 1000 * ceiling( g-agents-nw-xaxis-max / 1000 ) )

  if ( g-agents-nw-xaxis-max < ( g-agents-nw-xaxis-min + 1000 ) )
  [
    set g-agents-nw-xaxis-max ( g-agents-nw-xaxis-max + 1000 )
  ]

  ;; Setup for histogram "Net Worth of Prsns" in Panel 06.
  set g-prsns-nw-xaxis-min ( min prsn-nws ) ;; a number
  set g-prsns-nw-xaxis-min ( 1000 * floor( g-prsns-nw-xaxis-min / 1000 ) ) ;; a
number
  set g-prsns-nw-xaxis-max ( max prsn-nws ) ;; a number
  set g-prsns-nw-xaxis-max ( 1000 * ceiling( g-prsns-nw-xaxis-max / 1000 ) ) ;; a
number
```

```
  if ( g-prsns-nw-xaxis-max < ( g-prsns-nw-xaxis-min + 1000 ) )
  [
    set g-prsns-nw-xaxis-max ( g-prsns-nw-xaxis-min + 1000 )
  ]

  ;; Setup for histogram "Net Worth of Banks" in Panel 06.
  set g-banks-nw-xaxis-min ( min bank-nws ) ;; a number
  set g-banks-nw-xaxis-min ( 1000 * floor( g-banks-nw-xaxis-min / 1000 ) ) ;; a
number
  set g-banks-nw-xaxis-max ( max bank-nws ) ;; a number
  set g-banks-nw-xaxis-max ( 1000 * ceiling( g-banks-nw-xaxis-max / 1000 ) ) ;; a
number
  if ( g-banks-nw-xaxis-max < ( g-banks-nw-xaxis-min + 1000 ) )
  [
    set g-banks-nw-xaxis-max ( g-banks-nw-xaxis-min + 1000 )
  ]

  ;; Setup for histogram "P0 Assets of Banks" in Panel 06.
  set g-banks-P0-xaxis-min ( min [P0-all-assets] of banks ) ;; a number
  set g-banks-P0-xaxis-min ( 1000 * floor( g-banks-P0-xaxis-min / 1000 ) ) ;; a
number
  set g-banks-P0-xaxis-max ( max [P0-all-assets] of banks ) ;; a number
  set g-banks-P0-xaxis-max ( 1000 * ceiling( g-banks-P0-xaxis-max / 1000 ) ) ;; a
number
  if ( g-banks-P0-xaxis-max < ( g-banks-P0-xaxis-min + 1000 ) )
  [
    set g-banks-P0-xaxis-max ( g-banks-P0-xaxis-min + 1000 )
  ]

  ;; Setup for line graph "Bank P0 Assets - (Min, Mean, Max)" in Panel 07.
  set g-banks-P0-all-assets-min  ( min  [P0-all-assets] of banks ) ;; a number
  set g-banks-P0-all-assets-mean ( mean [P0-all-assets] of banks ) ;; a number
  set g-banks-P0-all-assets-max  ( max  [P0-all-assets] of banks ) ;; a number

  ;; Setup for line graph "Mean Net Worth" in Panel 07.
  set g-max-net-worth-priv-prsns ( max [net-worth-priv] of prsns )   ;; What it
says.
  set g-mean-net-worth-priv-prsns ( mean [net-worth-priv] of prsns ) ;; What it
says.
  set g-min-net-worth-priv-prsns ( min [net-worth-priv] of prsns )   ;; What it
says.

  set g-max-net-worth-priv-banks ( max [net-worth-priv] of banks )   ;; What it
says.
  set g-mean-net-worth-priv-banks ( mean [net-worth-priv] of banks ) ;; What it
says.
  set g-min-net-worth-priv-banks ( min [net-worth-priv] of banks )   ;; What it
says.


;;---------------------------------------------------------------------------|
  ;; Setup for Plot "AAAAAA"

  ;; This log entry may come from any step during debug operations.
  LOG-TO-FILE "  Do-aaa: All aggregates updated."
end

;;---------------------------------------------------------------------------|
;; DEBUG AND DEBUG LOG FILE MANAGEMENT FUNCTIONS
;;---------------------------------------------------------------------------|

;;---------------------------------------------------------------------------|
```

```
;; Construct a CSV data file name.
to-report fr-construct-file-name [ type-string ]
  ;; This routine is to be executed by the observer.
  ;;
  ;; Date-string format "01:19:36.685 PM 19-Sep-2002"
  let date-string date-and-time
  let file-name ( word "CmLab_" type-string "_" )
  ;; Append the year as yy.
  set file-name word file-name ( substring date-string 25 27 )
  ;; Append the month as Mmm.
  set file-name word file-name fr-convert-mmm-mm ( substring date-string 19 22 )
  ;; Append the day as dd.
  set file-name word file-name ( substring date-string 16 18 )
  ;; Append a dash.
  set file-name word file-name "_"

  ;; Append the hour as hh.
  set file-name word file-name fr-convert1224 ( substring date-string 0 2 ) (
substring date-string 13 15 )
  ;; Append the minute as mm.
  set file-name word file-name ( substring date-string 3 5 )
  ;; Append the second as ss.
  set file-name word file-name ( substring date-string 6 8 )
  ;; Append the .csv extension.
  set file-name word file-name ".csv"

  report file-name
end

;;---------------------------------------------------------------------------|
;; Open a log file for debug output.
to f-open-log-file
  ;; This routine is to be executed by the observer.

  ;; Ensure previous log file is closed.
  if ( is-string? gs-log-file-name )
  [
    if ( file-exists? gs-log-file-name )
    [
      file-close-all
    ]
  ]

  ;; Date-string format "01:19:36.685 PM 19-Sep-2002"
  let date-string date-and-time
  set gs-log-file-name "CmLab_Log_"
  ;; Append the year as yy.
  set gs-log-file-name word gs-log-file-name ( substring date-string 25 27 )
  ;; Append the month as Mmm.
  set gs-log-file-name word gs-log-file-name fr-convert-mmm-mm ( substring date-
string 19 22 )
  ;; Append the day as dd.
  set gs-log-file-name word gs-log-file-name ( substring date-string 16 18 )
  ;; Append a dash.
  set gs-log-file-name word gs-log-file-name "_"

  ;; Append the hour as hh.
  set gs-log-file-name word gs-log-file-name fr-convert1224 ( substring date-string
0 2 ) ( substring date-string 13 15 )
  ;; Append the minute as mm.
  set gs-log-file-name word gs-log-file-name ( substring date-string 3 5 )
  ;; Append the second as ss.
  set gs-log-file-name word gs-log-file-name ( substring date-string 6 8 )
  ;; Append the .txt extension.
  set gs-log-file-name word gs-log-file-name ".txt"

  file-open gs-log-file-name
  file-show "Log File for a CmLab (NetLogo) Model."
  file-show word "File Name: " gs-log-file-name
  file-show word "File opened at:" date-and-time
  file-show ""

  ;; Send a message directly to the command centre.
  ifelse ( file-exists? gs-log-file-name )
  [
    show word gs-log-file-name " opened."
  ]
  [
    show word gs-log-file-name " not opened."
  ]
end

;;---------------------------------------------------------------------------|
;; Convert month in text form to digital form.
to-report fr-convert-mmm-mm [ mmm ]
  ;; This routine is to be executed by the observer.
  ;; It converts a string in the form mmm ( alpha text ) to the form mm ( digit-text
).

  let mm "00"
  if( mmm = "Jan" ) [ set mm "01" ]
  if( mmm = "Feb" ) [ set mm "02" ]
  if( mmm = "Mar" ) [ set mm "03" ]
  if( mmm = "Apr" ) [ set mm "04" ]
  if( mmm = "May" ) [ set mm "05" ]
  if( mmm = "Jun" ) [ set mm "06" ]
  if( mmm = "Jul" ) [ set mm "07" ]
  if( mmm = "Aug" ) [ set mm "08" ]
  if( mmm = "SeP" ) [ set mm "09" ]
  if( mmm = "Oct" ) [ set mm "10" ]
  if( mmm = "Nov" ) [ set mm "11" ]
  if( mmm = "Dec" ) [ set mm "12" ]
  report mm
end

;;---------------------------------------------------------------------------|
;; Convert hour in 12 format to 24 hour format.
to-report fr-convert1224 [ hh ampm ]
  ;; This routine is to be executed by the observer.
  ;; It converts a string in 12 hour format to 24 hour format.

  let hour read-from-string hh
  if( ampm = "PM" ) [ set hour ( hour + 12 ) ]

  let dd ( word "00" hour )
  let d2 last dd
  set dd but-last dd
  let d1 last dd
  set dd ( word d1 d2 )
  report dd
end

;;---------------------------------------------------------------------------|
;; Close a log file for debug output.
to f-close-log-file
  ;; This routine is to be executed by the observer.

  let b-filename-exists 0
  if ( is-string? gs-log-file-name )
  [
    if ( file-exists? gs-log-file-name )
    [
      set b-filename-exists 1
    ]
  ]

  ifelse( b-filename-exists = 1 )
  [
    ;; Ensure the file is selected.
    file-open gs-log-file-name

    ;; Stanp it.
    LOG-TO-FILE word "File closed at: " date-and-time

    ;; Flush the buffers.
    file-flush

    ;; Close it.
    file-close-all

    ;; Note sent to command centre.
    show word gs-log-file-name " closed."

    ;; Revert to dummy name.
    set gs-log-file-name "dummyname"
  ]
  [
    if( gs-log-file-name = "dummyname" )
      [ show "No log file is open.  Cannot close it." ]
  ]
end

;;---------------------------------------------------------------------------|
;; Select an already opened log file.
to f-select-log-file
  ;; This routine is to be executed by the observer.

  ifelse ( file-exists? gs-log-file-name )
  [
    ;; Ensure the file is selected.
    file-open gs-log-file-name

    ;; Ensure it is open for writing.
    LOG-TO-FILE ""
    LOG-TO-FILE "SELECTED"
  ]
  [
    show word gs-log-file-name " is not open.  Cannot select it."
  ]
end

;;---------------------------------------------------------------------------|
;; Change the debug mode from on to off, or vice versa.
to f-toggle-debug
  ;; This routine is to be executed by the observer, and is activated by a
  ;;   button.
  ifelse( gb-debug-on = 1 )
  [
    ;; Debug is On, turn it Off.
    ;; Close the file before turning debug logging off.
    f-close-log-file
    set gs-debug-status "0 (Off)"  ;; This appears in the monitor.
    set gb-debug-on 0              ;; But this controls the debug feature.
  ]
  [
    ;; Debug is Off, turn it On.
    set gs-debug-status "1 (On)"   ;; This appears in the monitor.
    set gb-debug-on 1              ;; But this controls the debug feature.
    ;; The switches, if needed, are reset manually by the user.
    ;; Open the log file after turning debug logging on.
    f-open-log-file
  ]
  ;; end of f-toggle-debug
end

;;-----------------------------------------------------------------------------|
;; Toggles debug on.  Used as a sieve.
to f-force-debug-output-on
;; This routine can be executed by anybody.

  if( gb-debug-on = 1 )
  [
    f-toggle-debug  ;; Turn it off.
  ]

  if( gb-debug-on = 0 )  ;; A certainty, now!
  [
    f-toggle-debug                     ;; Set flag on, opens debug file.
    set gs-debug-step-chooser "all"    ;; Opens for all steps.
    set gb-debug-flow-on 1             ;; Turns on LOG-TO-FILE flows.
    set gb-debug-show-steps true       ;; Directs flows to screen also.
  ]
;; end of f-force-debug-output-on
end

;;-----------------------------------------------------------------------------|
;; Toggles debug off.
to f-force-debug-output-off
;; This routine can be executed by anybody.

  if( gb-debug-on = 1 )
  [
    f-toggle-debug  ;; Turn it off.
  ]

;; end of f-force-debug-output-off
end

;;-----------------------------------------------------------------------------|
to f-regulate-debug-switches
  ;; This routine is to be performed by the observer.

  ;; There are certain combinations of debug switch settings which are meaning-
  ;;   less when in debug mode.  Rather than placing this logic here and there
  ;;   throughout the application, this routine has the logic to ensure that
  ;;   the debug switches remain in a meaningful configuration.
```

```
  if(gb-debug-on = 0 )
  [
    ;; The debug feature is turned off.  All switches should be set to default
    ;;   positions, which is 'Off', or zero, or false.
    set gb-debug-show-steps false
  ]

end

;;-----------------------------------------------------------------------------|
;; 'Show' a string in a debug log.
to LOG-TO-FILE [ log-this-string ]
  ;; This routine may be executed by any agent.
  ;; It should be invoked as a debug routine only, and would not be used for
  ;;    normal output.  It sends output to the debug log file, or, optionally,
  ;;    also to the command centre.


  f-regulate-debug-switches

  ;; gb-debug-on is a global Boolean and has value 1 (true) or 0 (false).
  if( gb-debug-on = 1 )
  [
    ;; gb-debug-flow-on is declared as a global Boolean variable, and its value
    ;;   is 0 ( false ) or 1 ( true ) and is set on or off at the beginning of each
    ;;   function ( each do-step ).  It is controlled by the chooser that selects
'all'
    ;;   or a specific do-function.
    ;;
    ;; When it is 'on' you can assume the debug log file exists and is open for
    ;;   write.

    if( gb-debug-flow-on = 1 )
    [
      file-show log-this-string
      if( gb-debug-show-steps = true )
      [
        show log-this-string
      ]
    ]
  ]
end

;;-----------------------------------------------------------------------------|
;; This replicates the effect of an 'ASSERTION' in C++
to ASSERT [ error-test error-string error-who ]
;; This routine can be run by any agent.

if( error-test = false )
[
  show ( word error-test " " error-string " " error-who )
  ;; Cause a run-time error and display a message.
  error ( word "Agent: " error-who " - " error-string )
]

end


;;-----------------------------------------------------------------------------|
;; Check whether the agents are all valid.
to-report frb-agents-are-all-valid
;; This routine can be run by the observer.
```

```
  let b-agents-are-all-valid true

  ;; TODO: fix this.
  if( gb-debug-on = 1 )
  [
    ;; Do the check only if debug is on.

    ;; Check the GCRAs.
    ask gcras
    [
      if( frb-gcra-is-valid = false ) [ set b-agents-are-all-valid false ]
    ]

    ;; Check the crbs.
    ask crbs
    [
      if( frb-crb-is-valid = false ) [ set b-agents-are-all-valid false ]
    ]

    ;; Check the banks.
    ask banks
    [
      if( frb-bank-is-valid = false ) [ set b-agents-are-all-valid false ]
    ]

    ;; Check the prsns.
    ask prsns
    [
      if( frb-prsn-is-valid = false ) [ set b-agents-are-all-valid false ]
    ]

    ;; Check the corps.
    ask corps
    [
      if( frb-corp-is-valid = false ) [ set b-agents-are-all-valid false ]
    ]
  ]

  report b-agents-are-all-valid
end

;;----------------------------------------------------------------------------|
;; Check whether a GCRA is valid.
to-report frb-gcra-is-valid
;; This routine can be run by a GCRA.

  let b-gcra-is-valid true

  report b-gcra-is-valid
end

;;----------------------------------------------------------------------------|
;; Check whether a crb is valid.
to-report frb-crb-is-valid
;; This routine can be run by a crb.

  let b-crb-is-valid true

  report b-crb-is-valid
end

;;----------------------------------------------------------------------------|
;; Check whether a bank is valid.
to-report frb-bank-is-valid
;; This routine can be run by a bank.

  let b-bank-is-valid true

  report b-bank-is-valid
end

;;----------------------------------------------------------------------------|
;; Check whether a prsn is valid.
to-report frb-prsn-is-valid
;; This routine can be run by a prsn.

  let b-prsn-is-valid true

  report b-prsn-is-valid
end

;;----------------------------------------------------------------------------|
;; Check whether a corp is valid.
to-report frb-corp-is-valid
;; This routine can be run by a corp.

  let b-corp-is-valid true

  report b-corp-is-valid
end

;; ---------------------------------------------------------------------------|
;; END OF all CODE
;; ---------------------------------------------------------------------------|
```