Var_Checker

Paul Hickman

COLLABORATORS			
	TITLE : Var_Checker		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Paul Hickman	April 15, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Var_Checker

1.1 AMOSPro Variable Checker Version 1.06

```
                      AMOS Pro Variable Checker Version 1.06


         By Paul Hickman  E-mail: ph@doc.ic.ac.uk
```

1.2 Varchecker / Introduction

```
                                 Introduction

   The  variable checker is an accessory program which scans another
   programs  source  from  the AMOS Pro editor, and reports possible
   errors  in  the  use  of  variable  names  that AMOS Pro does not
   normally  detect.   This helps  debug programs, optimises them by
   allowing  to  remove assignments that are unneccessary, and leads
```

to better programming practice.

Version 1.06 now includes context sensitive on-line help - Just press the ‘Help’ button on any dialog box, or the help key on the keyboard to be taken straight to the relevant Amigaguide help page.

1.3 Varchecker / Requirements

Requirements

- Any version of AMOS Pro. (Doesn’t work with AMOS Creator)
- Enough memory to load the accessory, and the program to check simultaneously.
- Easylife V1.08+ (Supplied)

1.4 Varchecker / Distribution

Distribution Conditions

This progam may be freely distributed, as long as no profit is made from doing so. It may not be put on a disk together with programs for which a profit is made from distributing them.

It may be distributed seperately from the easylife archive, but AMOSPro_EasyLife.lib & Easylife.Library should be included along with this document. However please include a readme file describing what easylife is, and how to get the full distribution with this program (You can crib this from the Readme.Guide).

1.5 Varchecker / Glossary

Glossary

Reference / Referal

A Reference to a variable is where the variable is used in the arguments of an AMOS function, or command where it’s value is read, and not altered. The following are all references to VAR:

```
Print "The Answer is:";VAR
_CALL_THE_PROCUEDRE[1,"Fred",VAR,False]
X=4+VAR*2
End Proc[VAR]
```

Declaration

```
    The  declaration of a variable notifies AMOS that it exists.  All
    Definitions/Assignments  are  also  declarations, as are variable
    names in Shared & Global statements.



Definition / Assignments

    The  definition  / assignment of a variable is where the value of
    the  variable  is  set  by an AMOS command, or the '=' operation,
    without  refering  to  VAR.  The following are all definitions of
    VAR, as they all set a new value of VAR which is not based on any
    previous value of VAR:

  VAR=4
  Read VAR
  Input "Enter Value Of Var:";VAR
  Input #1,VAR;
  Line Input #1,VAR
  Procedure A_PROCEDURE[VAR]


    These  are  not definitions of VAR, as they always require VAR to
    be set (Or implicitly take it to be 0 if it isn't)

  VAR=VAR+4     (Also a reference to VAR)
  Add VAR,6
  Dec VAR
```

1.6 VarChecker / Installation

```
                               Installation

    From V1.02 onwards, Variable Checker is run from the AMOSPro
    user menu ONLY.  This involves the following installation procedure:

 1) Copy the VarChecker.AMOS program to its permanent location, if it
    is not already there (The easylife installer script does this).


 2) Select "Add Option" on the user menu. Enter 'Var Check' as the
    option name.


 3) Select the new Var Check option to assign a program to, then select
    the VarChecker.AMOS program.


 4) In the request that appears, enter the command line of 'CHECK', and
    switch on the options 'Load As Hidden Program' & 'Keep After Run'.
    This is very important. Press OK.


 5) Select the Var Check option again, and press the keys
```

Control-Shift-V.

6) Repeat steps 2-5 to create a second menu option 'Var Report', but this time set the command line to 'REPORT' and the keyboard shortcut to Control-V.

7) Select 'Save Default Configuration' from the Config menu to save the 2 new menu options.

8) To activate the on-line help feature make sure AmosGuide.AMOS is correctly installed as the AMOSPro help file viewer, and that the 'Docs' drawer from Easylife archive 2 is a component of your HELP: assignment. VarChecker will read the path AMOSPro uses to access AmosGuide.

1.7 VarChecker / Usage

Usage

To perform a variable check, load the AMOSPro program to be checked and select its window. Then select the 'Var Check' option from the user menu, or press Control-Shift-V. The variable checker options page will then appear. Most of this page is taken up by 11 switches to enable/disable the various checks that can be made:

```
F1:  Undefined Local Variables

F2:  Undefined Shared / Global Variables

F3:  Unreferenced Local Variables

F4:  Unreferenced Shared / Global Variables

F5:  Unused Shared Variables

F6:  Unused Shared / Global Variable

F7:  Global Variables that are shared

F8:  Variables made Global after use

F9:  Labels that are procedure names

F10: Variable Name Missing After A 'Next' Instruction

ESC: Misuse Of Constant Variables
```

Three other error checks are also performed, which cannot ↩ be disabled. There are:

- A check for variable names which are the same as procedure names. Don’t do this, it is not only confusing to the reader of the program, it is confusing to the variable checker itself!

- A check for global or shared variables appearing in the format argument lists of a procedure defintion. This definitely should not be done with global variables, or shared variables which are shared with the procedure they appear in the header of. To use a variable which is shared in another procedure is not such a major problem, but should still be avoided.

- A check for procedures which are never called. NOTE:This will not detect calls to procedures via Menu$ "PR" strings.

There are three control buttons at the bottom of the requester:

Stored

Perform a variable check, but to not report all the errors immediately. Instead they are stored in a data bank, and at the end of the check, the last error found in the program is reported, and the check ends.

To display the rest of the errors, select the ’Var Report’ option from the user menu (Or press Control-V). The means you can correct the errors as you go.

NOTE: The errors are reported in order of the line number the occur on, last to first. This is done backwards so you don’t muck up the line numbering of stored errors by inserting new lines when correcting the current error.

NOTE: This will not work properly if you did not select the ’Keep After Run’ option when installing the variable checker.

Interactive

This is similar to the old versions of variable checker - A requester pops up when each error is detected:

- The program listing is moved to the line containing the variable for all errors, including those detected at the end of procedures, and the end of the program.

- A ’Store’ button has been added that will store this error, as if the program were running in stored mode.

- The ’Ignore’ button will forget this error, and continue checking.

- The ’Abort’ button will stop the check, and return to the editor.

NOTE: The first stored error (In any are stored) is not reported at the end of the check in this mode - You must press Control-V.

Help

Displays this page using AmosGuide (Requires AmosGuide to be installed as the AMOSPro help viewer).

Cancel

Guess!

1.8 VarChecker / Checks / Undefined Local Variables

F1: Undefined Local Variables

Detected: As soon as the variable is used.

An undefined local variable is a variable which is not global, or shared, and is refered to in the program, before it has been defined (See Glossary)

This error occurs if the variable is used above the point in the program at which it is defined. This should not be a problem with well structred programs, but those which use Goto/Gosub may find that frequently a variable is defined after the point it is used.

NOTE: No error will occur at this stage, if the variable is a shared or global variable. They are only reported if they are never defined anywhere. This also applies to shared variables when they are used in the main program.

This is the most frequently occuring error, and can be the most dangerous. If possible (practical -it’s always possible), you should never asssume that a variable is uninitialised, and therefore 0, as:

- You might modify the program to use it later, and it will then stop working.
- The program will throw out loads of ’Local Varaible Undefined’ errors, obscuring the times when you have actually mis-spelled a variable name, which is the case when this error being in your program makes it incorrect, and may make it crash.

NOTE: This error will only occur the first time each undeclared variable is used in each procedure/the main program.

1.9 VarChecker / Checks / Undefined Shared & Global Variables

```
F2: Undefined Shared / Global Variables

   Detected: After all lines have been checked

   This   error   occurs   for   variables   that   are  defined  as
   Shared/Global,  and  are  referenced one or more times, but never
   have a value assigned to them anywhere in the program.
```

1.10 VarChecker / Checks / Unreferenced Local Variables

```
F3: Unreferenced Local Variables

   Detected: At the end of a procedure / end of the program.

   This  is the opposite of the 'Undefined Local Variable' - A value
   has  been  assigned  to  a  variable,  but  the variable is never
   refered  to in the arguments of a function / command.  This error
   only occurs within procedures during this stage of checking.


   NOTE:   Sometimes  an  unreferenced variable is not an error e.g.
   A=Dialog  Box(A$,0,B$).   If  you  don't care what the dialog box
   returns, but just want to display it, A will be unreferenced.  In
   such  cases,  A  should be replaced with NULL.  This program does
   not  produce  errors  if NULL is unreferenced.  By forcing you to
   use NULL, this shows other people reading the code that it is not
   used.  (NULL# & NULL$ can be used when appropriate)

   NOTE:   This  error will not occur for variables that are defined
   as loop counters in For...Next loops.
```

1.11 VarChecker / Checks / Unreferenced Shared & Global Variables

```
F4: Unreferenced Shared / Global Variables

   Detected: At the end of the program.

   This   error   occurs   for   variables   which   are  defined  as
   Shared/Global, and have a value assigned to them at some point in
   the  program,  but  are  never  refered  to in the arguments of a
   command or function.

   New  for V1.02:  This check will also detect procedures which are
   never  called.   NOTE:   Procedure  Calls from Amos Menus are NOT
   detected,  so  you may get a procedure never called error message
   in such cases.
```

1.12 VarChecker / Checks / Unused Shared Variables

```
 F5: Unused Shared Variables

    Detected: At the end of a procedure

    This  error  occurs  if a variable is made shared in a procedure,
    but is then not used in that procedure - I.E.  it is not assigned
    to,  or  refered to.  Note the difference between this error, and
    the next - they are not the same.
```

1.13 VarChecker / Checks / Unused Shared & Global Variables

```
 F6: Unused Shared / Global Variables

    Detected: At the end of the program

    This  error  occurs  for variables that are defined as Shared, or
    Global,  and  then  not  used  anywhere in the program, either in
    definitions or references.
```

1.14 VarChecker / Checks / Global Variables that are shared

```
 F7: Global Variables that are shared

    Detected: Immediately

    Making  a global variable shared is a pointless exercise.  Remove
    the  shared  statement  from the offending procedure.  (Using the
    Editor_Enhancer.AMOS Improved Search & Replace of course :-)

    NOTE:  Even if a global variable is shared in several procedures,
    this  error will only be reported once, at the line of the global
    statement.
```

1.15 VarChecker / Checks / Variables Made Global After Use

```
 F8: Variables made Global after use

    Detected: When the variable is made global


    This detects code such as:

        X=4
        Global X

    Doing this can confuse the AMOS compiler, and it is generally bad
    programming  practice.   Move all Global statements to the top of
    your program.  This is the correct way:
        Global X
        X=4

    NOTE:   AMOSPro  allows  you  to  make  variables Global inside a
    procedure.   Don't do this - it will only cause compiler problems
    &  problems  for  people  reading  your  code (Including you in 6
    months time :-)
```

1.16 VarChecker / Checks / Labels That Are Procedure Names

```
 F9: Labels that are procedure names

    Detected: Whenever the label occurs.

   If you have a procedure called FRED, and a line such as:

        FRED: Print "Hello"

   The  Variable  Checker  will  now point this out to you, as it is
   probable that what you actually meant was:

        FRED : Print "Hello"

   Which  means  something  completely different.  This error occurs
   whenever a label matches the name of a procedure.
```

1.17 VarChecker / Checks / Variable Name Missing After A 'Next' Instruction

```
 F10: Variable Name Missing After A 'Next' Instruction

   Detected: Immediately

   This  check  looks at the 'Next' instructions of For...Next loops
   to ensure that the loop counter variable occurs after the 'next'.

   NOTE: Putting it there does not slow things down.
```

1.18 VarChecker / Checks / Misuse Of Constant Variables

```
 ESC: Misuse of Constant Variables

    Detected: Immediately

    AMOS  does  not  have  any concept of named constants, except the
    AMOSPro  equates,  which  is  just  a  hack  to  get around this.
    However,  you  can  of  course  use variables as constants.  This
    check  will  make  sure you obey the following rules when using a
    variable as a constant:

  - The name must begin with a double underscore e.g. __FRED
```

This is to identify constants, against normal variables.

- It must be a global variable.

- You must assign a value to the constant before it is used.

- You must never change the value of a constant.

The variable checker makes sure you do not violate these rules. another utility OPTIMISE.AMOS will replace constants with the actualy values they represent, but you should check them with this program first. If the switch is off, variable names begining with a double underscore are treated like any other variable.

NOTE: Changes to a constants value are not detected if they are not made with the '=' operator. E.g. Add __FRED,6 is an error, but this version of the variable checker won't spot it. See OPTIMISE.Doc for more information on constants.

NOTE: Variable names begining with 'ST_' are also treated as constants by the variable checker. This is the prefix used by the structure compiler accessory when defining structured varaibles.

1.19 Varchecker / Miscellaneuos

Miscellaneous

Other Controls:

- All requesters can be dragged vertically by holding the mouse button down over the title bar.

- Press Control-C to abort a check. Any errors stored before you aborted can still be reported correctly with Control-V

- The first letter of any button in a requester can be used as a shortcut for pressing the button.

Notes:

- If for some reason you save the VarChecker.AMOS program, erase bank 10 first. It holds the stored errors, and must survive between runs of varchecker, but need not be saved.

1.20 VarChecker / Cavats

Cavats

- Arrays are not checked, as AMOS requires that all arrays are dimensioned, before use, and produces an error if you try to refer to an undimensioned array. However, the index parameters are checked for references to other variables.

- The arguments of Goto/Gosub statments are not checked for variable references, as there is no way to distinguish them from label references. The first expression of On <exp> Goto etc. statements is checked properly.

- Commands names that have several words, the last of which is a single letter, and have no arguments may be mis-interpreted as shorter command names with 1 - letter variables as arguements e.g. An example is the AMOS Turbo Plus command 'Scene X'. This program would interpret that as the command 'Scene', followed by a reference to the variable 'X', and produce an error if 'X' is not defined. The program knows about a few commands such as 'Scene X' and ignores them. If you find any others, please E-Mail me.

- Def Fn statements are not checked for references, as they are not interpreted at the point they are defined.

- Procedures which can not be unfolded (E.g. Contain Machine Code), will cause this program to think any code between them and then next procedure header is the code of the folded procedure, and not part of the main program. To prevent this, place these procedures at the end of your program, or follow them immediately by another procedure. It will also generate unreferenced variable errors for the procedure arguments.

- Wildcards in Global/Shared statements are ignored. If you didn't know that you could use them, don't bother finding out how - not only will it stop this program testing your program correctly, it is asking for trouble, with variables "accidently" becoming global. Don't be lazy, and list each one explicitly. It also makes your code easier for others to read.

- Calls to procedures from inside an AMOS Menu definiton are not detected, and the procedure will be reported as never being called unless it is also called from outside the menu

- The constant misuse check will not spot changes made to a constants value by instructions such as Add, Dec, Inc.

1.21 VarChecker / Suggestions For Use

Suggestion For Use:

This program can be used in three ways:

- To find out why a program isn't working.

- To optimise them by removing unrefered to variables.

- To remove all errors found by this program from your code, even

```
    if  they are not "Real Errors" - E.g.  this code would produce an
    error but is not wrong.


  Procedure COUNT_NUMBERS[S$]

     X=1
      While X<Len(S$)
         C=Asc(Mid$(S$,X))
         IF (C=>48) and (C<=57) Then Inc RESULT
         Inc X
      Wend
  End Proc[RESULT]


    However, it is better coding to add to the end of the first line,
    ' : RESULT=0'.   This  will  get  rid  of   the  'Undefined Local
    Variable' error.

    I  recommed you try to make your code produce no errors from this
    program,  so  it  makes  it  less  likely  to  go  wrong if it is
    modified.   This  includes  not  using wildcares in Global/Shared
    statements  (Which  I  only  found  out you could do today, and I
    think it was a rather stupid idea).
```

1.22 VarChecker / Program History

```
      Program History

CHANGES V1.06 ---> V1.05

  - NEW FEATURE: Added context sensitive On-Line help to dialog boxes

  - Check for uncalled procedures was undocumented.

CHANGES V1.05 ---> V1.04

  NOTE: V1.05 was accidentally distributed with V1.04 documentation.
        Sorry.

  - Bug Fix: Error in deleting structures to represent local variables
    fixed. Checking now uses slightly less memory.

  - Improved effeciency of structured variables usage.


CHANGES V1.04 ---> V1.03

  - Bug fix. Uses of shared variables in the main program were treated
    as local variables. This bug was introduced in V1.03.

  - Bug fix: An error in the tree deletion routine caused variables
    to be "forgoten" sometimes.

  - Escape & C keyboard shortcuts swapped in the main requester, so all
```

```
    buttons have their 1st letter as a shortcut

  - NEW FEATURE: Extra check for global/shared variables in procedure
    headers added.

  - In Stored Mode, the first error is not shown until you run the
    program again in report mode. Therefore you can go away and do
    something else whilst it is checking without missing the first
    error report.


CHANGES V1.03 ---> V1.02

  - Switched to using easylife structured variables to store the
    internal data. This lead to a 2-5 times speed increase, and no
    problems with the AMOS variable buffer becoming full. This is
    because a binary tree is used to store variable names.

  - This also removed the limit of 300 variable names, as they are
    not stored in an array anymore.



CHANGES V1.01 ---> V1.02

  - Five new error checks added.

  - Options requester added.

  - Stored mode added.

  - New dialog interface routines.

  - Slight speed improvements. (Stored mode is much faster than
    interactive, as the program listing is not scrolled).



Changes V1.00 ---> V1.01

  - BUG FIX: Did not spot references to variables followed by spaces. This
    happened with variable names immediately before ’and’,’or’,’div’,’mod’
    etc. (Thanks to Ben Marty for spotting that one)
```

1.23 VarChecker / To Do

```
          To Do

  - Remove some of the cavats (DEFN: CAVAT = Bug I don’t want to call a
    bug)

  - Any Ideas?
```

1.24 VarChecker / Cavat Reports :-)

```
        Cavat Reports :-)

   Given  that the AMOS Pro editor itself sometimes cannot read your
   programs  correctly,  I will be amazed if I have managed to cover
   every  possible  case  in this program.  If you find it reports a
   name  that is not a variable, or fails to report an error that it
   should, please E-Mail either the whole program (I never say no to
   free programs), or just the line that goes wrong.

   NOTE:  If you use any extensions other than those listed below in
   code  you  send, either send me the extension as well (preferable
   if  it's  legal),  or send it in AscII form, making clear what is
   command, and what is variable.  I Have:

 AMOS Creator Compiler
 AMOS 3D
 ANOS Turbo / Turbo Plus
 EasyLife (Shock - Horror!)
 Intuition

   NOTE: The Intuition extension I have is the one by Andrew Church.
         There are others, or so I hear.
```