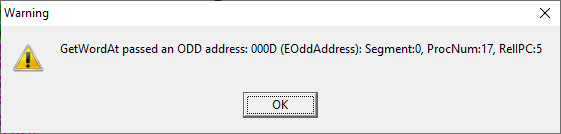
**1/2/2024:**

* **Version IV forgets what the terminal was**
* **When the pSystem RCE occurs (*s[0] := chr(8))*) we end up back in the WAIT loop but the p-System isn’t really waiting for a key**
* ***crttype* contains the terminal type (or does it? no. It is not used)**
* **CrtInfo.TermType should contain the *TermType*. Does it get saved?**
  + ***TfrmFiler.SaveCrtSettings* would like to save the *TermType* but the f*pSysWindow* variable is nil**
  + **Now using the *Sender* parameter to *TfrmFiler.SaveCrtSettings***
* **Exceptions (such as RCE, lead to endless loop)**
* **While parsing *'F:\ndas-i\d7\Projects\pSystem\DebuggerSettings\CSVFilesToLoad-IV.2.2.CSV'*  I am getting:** 
  + **“ is not a valid integer value**
  + **but the file that the data was saved to is CSVFilesToLoad-IV.2.2 (Tue).CSV**
    - **BootParams.VolumesToMount never got updated**
* **I can get into LoadVersion while another p-System is already running**
* **{12} GetWordAt passed an ODD address-- called from SIND when trying to run T0007A**

****

* ***F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\UCSDI5.RAW* is not marked as a *Non-Standard volume* in *CSVFilesToLoad-I.5.CSV***

**1/3/2024:**

* **The *Surface Pro Recover* USB is in the laptop bag**

**1/5/2024:**

* **When I cancel out of a boot sequence, it gets removed from the BootParams**
* **I am getting an ODD address error on both the PM and LB Version 1.4 interpreters at DbgCnt = 88 on a STO instruction**
* **Possible sources of the problem:**
  + **GetDefaultBootParams - try going back to 12/29/2023 version of FilerMain**

**1/6/2023:**

* **I can boot LB I.4b {F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\UCSDI4.RAW} PSYSTEM.EXE dated 5/16/2023 with no problem**
* **But PM I.4b {F:\NDAS-I\d7\Projects\pSystem\Volumes\Linux Vols\system.vol} PSYSTEM.EXE dated 5/16/2023 gets the ODD address error.**
* **Also get the error with LB I.4b {F:\NDAS-I\d7\Projects\pSystem\Volumes\Linux Vols\system.vol} PSYSTEM.EXE dated 5/16/2023 gets the ODD address error.**
* **Can NOT boot LB I.4b {F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\UCSDI4.RAW} PSYSTEM.EXE dated 5/16/2023 -- gets the odd address problem**
* **fpCodesProcTable is nil in TfrmLocalVariables.UpdateDisplay**
* **The first appearance of SetAccDbFileNumber occurs on 20201231**
* **I am comparing to pr20201222\pSystem\Debugger\LocalVariables**
* **TfrmLocalVariables.Create previously had aDatabaseFileName passed in as a parameter**
* **Where is TpCodesProcTable.Create supposed to get the database filename? DEBUGGERSettings.DatabaseToUse**
* **It (ODD address error) has started working again for no obvious reason. Both the LB and PM versions boot and run OK.**
* **I am getting duplicates in the Boot Params list.**
* **Major problems attempting to get the new SurfacePro running**
  1. **Trying to initialize the “New” Surface Pro I had various problems such as:**
     1. **Random stops requiring a reboot**
     2. **Battery indicator flip-flopped from 100% charged to 0% charged**
  2. **I tried to restore a drive image from backup drive. After going through the restore process, I got an error message “Invalid Parameter”**
  3. **I tried to restore the drive image from a different backup image. After the restore process (with no error messages) it would boot to my old operating environment but would turn itself off when I tried to enter a new PIN number.**
  4. **Tried to restore from a “Save Point”. It did not help. The Surface Pro continued to go black randomly.**
  5. **After a “reload from drive image” claimed to have succeeded, the Surface Pro crashes (i.e., goes black) as soon as I try to enter my password.**

**1/9/2024:**

* **The ODD address occurs in LB 1.4 (DbgCnt = 51249) on aUJP** 
  + **(MemRd(WordIndexed(JTab, -1)) + 2 - (MemRd(JTab - disp) + disp)) = $FFFF538B**
  + **JTAB = $CD6E**
  + **Can I use SelRelPtr?**
  + **How is JTAB used?**
  + **Does the problem exist in version 1.5?**
* **Version PM 1.5 appears to have an ODD address problem rather than a RCE at DbgCnt 79390**
* **Can I view the relevant source code where the ODD address error occurs? History shows nothing but SLDC0. FETCH was referencing fOp rather than fOpCode.**
  + ***History* claims that the error occurs in *PASCALSYS.CHECKFIL* (p#17, ipc 5 SIND5). This appears to be FJP 81 88. SIND5 has already been executed 227 times (sometimes--- depends upon the program flow) (DbgCnt = 68898 or other) . This is doing a GetWordAt(13)**
    - **Note that version 1.4 has problems with a FJP while version 1.5 is getting an ODD address error**
    - **The problem occurs in CHECKFIL (CHECKFIL<-FGET<-COMPINIT) but it has already been executed many times.**
    - **SP = 500 ($01EE)**
    - **' @ $01F4 [ 500]: 0001 D360 D372 D384 DBD2 DBE2 0000 0001 0001 0000 0000 0000 0021 03DC 0023 0972 0028 0836 0000 0000 0000 0000 0000 0000 0000'**
    - **'Words @ $0001 [ 1]: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000'**
    - **I seem to be looking at the address=1 rather than $01ee. I changed the breakpoint to display the stack (SP) rather than TOS**
    - **See Reports\Messages (Tue).txt**

**1/10/2024:**

* **When dumping a grid, it would be nice to display the ‘value’ description rather than the word ‘Value’**
* **When crashing on a SIND5, the top word on the stack has been popped and used as an indirect value to pass to WordAt which receiving the value 13**
* **Push(WordAt[WordIndexed(Pop(), fOpCode - SIND\_0)]);**
  + **pops 3 off of the stack**
  + **tries to index off of it by fOpCode - 248 → 253 - 248 → 5 (i.e. SIND5) words**
* **When entering BreakPoint info with an unknown segment name, the SEGNAME gets replaced with a filename.**
* **When the crash occurs, the stack has more stuff pushed onto it (that never got popped off?).** 
  + **Location 492 got the value of 3 in SegNameIdx = 18, CURPROC = 17 (CheckFil). Occurs after SLDO3, DbgCnt = 64883, IPC = 1. This is the first instruction of *CHECKFIL*.**
  + **There are 64886 - 64883 {= 3} instructions from the detection to the crash.**
  + **SLDO3 is used 2142 times**
  + **In INITSCAL the LPA sets the IPC to 10575 which is way out of range (I think)**
  + **How do I find out what the current procedure name is?**
  + **Set a breakpoint in INITSCAL and step down to around IPC 70**
  + **Put a break in LPA and look at the byte stream at the IPC**
  + **InterpII does not have a LPA opcode (OpCode = $D0). It has a S1P opcode (also OpCode = $D0) instead.**
    - **I have renamed S1P to LPA**
    - **The string is in the code stream.**
    - **Its address is on the top of the stack**
    - **It has a length byte**
    - **We point past the length byte**
    - **and push the resulting address onto the stack**
    - **The p-Code that comes from ACCDB appears to have both an LCA followed by an S1P (or LPA).** 
      * **The LCA is the length byte followed by the ASCII char bytes**
    - **After changing I am now getting “Illegal Character in Text” in the Peter Miller version.**

**1/11/2024:**

* **There is no consistency between the various versions of LPA**
  + **The Peter Miller version does not pop anything off of the stack**
  + **PM version of LPA advances the IPC past the string**
* **Why does this *Opstable.Ops[fOpCode]* not work after I leave the *Fetch* routine?**
* ***ProcNamesInDB* is not getting populated**
* **How do I regenerate the p-code in the debugger? Use the *External Decoder Window*.**
* **Remember:**
  + **LoadSegmentName**
  + **SegNamesInDB**
  + **ProcNamesinDB**
  + **format is "ProcName {SegName}"**
* **MVB is doing something to the stack- it is popping both the SRC and the DST address (= 4 bytes)- OK**
* **IDSEARCH is crashing when trying to assign *SymCursor := P^.SYMCUR;***
* **PM V2.0 is now choking when compiling T0005a.text: getting various errors (59, 104**

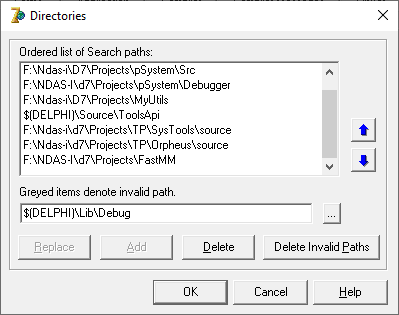
**1/12/2024:**

* **Can I view the Debian folders from XPS-8930? Yes. \\Debian\public**
* **Can I see what's in them on Debian? Yes. I can see *public:***
  + **15sys1.vol**
  + **E4EDITO.VOL**
  + **system.vol**
  + **TEST.VOL**
  + **U134.4\_OS\_SOURCE.vol**
  + **UCSDI4.VOL**
  + **UCSDI5.VOL**
  + **UCSDII.VOL**
  + **utility.vol**
* **I can compile and run the TEST:T0001A program on Debian with no error.**
* **I can compile and run the TEST:T0005A program on Debian with no error. This checks the *IDSEARCH* functionality.**
* **Can I run the gdb debugger?**
* ***system.vol* is a version II system.**
* **I first noticed the odd address error on** **5/27/2023 (see the “*How to create   
  DOSBOOT (2023)”* document)**
* **(Debian) to run the debugger…**
* **Type *code* from the “*dand@debian:~$”* prompt**
* **I want to get things working (?) again for PM 1.4, PM 1.5. Try to “*{$IfDef*” PM way and LB way.**
* **Note that OpCode 166 can be either LSA (V 1.4, V1.5) or LSA (V 2.0).**
* **OpCode 167 has also changed from *LDO* to *LAE***
* **OpCode 169 has also changed from *MVB* (V 1.4, V1.5) to *LDO* (V 2.0).**
* **Put breakpoints on LCA/LSA, LDO/LAE, LCA/LSA and compare to the code executed to the “C” version.**
* **The PM version is always calling LSA- never LCA**
* **LSA gets an assertion error at DbgCnt 457 (CurProc = 3 *{INITUNIT}*, IPC=21)**
* ***LCA 0 '' -- Takes 2 bytes- one for the LCA opcode and one for the length byte (=0)***
  + **I do not see how the *assert()* in the C code could always pass?**
  + **What is in assert.h?**
* **LCA does NOT always start on an even address.**
* **can i determine which compiler VERSION compiled a program?**
* **Look at some version II code and see if LCA/LSA is always an even address**
  + **The version II compiler precedes the LCA (166) with a NOP (215)**
* **See: *COMPINIT* which is called by *COMPILER***
* **I need to distinguish which version of the compiler compiled the .CODE file otherwise assume that if I am running V2, watch for odd addresses. If not don’t worry about them.**
* **Watch how MVB, LDO, LDO, LAE behave**

**1/13/2024:**

* **Peter Miller version 2 interpreter programs seem to be working (using either *SYSTEM.VOL* or *UCSDII.RAW*)**
* **BreakPointInfo is getting displayed spuriously**
  + **Displayed during DebuggerLoadFromUnit**
  + **AddBreakPoint2 gets called.**
  + **Debugger.`UpdateDisplay. Sent by a TMemo**
  + **DisplayPCodeToPage → DisplayMemo → DisplayMemoField → SetTextBuf**
* **Peter Miller version I.5 still gets an ODD address error in SIND**
  + **LPA is still going off of the rails**
  + **FetchB is getting a very large value for the *byte count***
  + **The ODD address error is first occurring in the debugger:** 
    - ***MemDumpDW(46705, wt\_Word, 10, ‘’)***
    - ***MemDumpDW(AbsIPC, wt\_Word, 10, ‘’)***
    - **Byte stream before LPA:**
      * **Bytes @ $B6B3 [46771]: D0 A9 08 DA A2 0E 00 9A DA A2 0C 00 9A F5 01 82 AB 0E B9 F6 9F AB 3B E8 FD A1 17 A5 81 CD A6 0E 2A 53 59 53 54 45 4D 2E 50 41 53 43 41 4C AA 28 B9 16**
    - **COMPINIT in the DB has changed?**
      * **For some reason *COMPINIT.INITSCAL* is showing up as *COMPINIT.PROC06*?**
    - ***TfrmSegmentProcName.edtAccessFileNameChange* DOES NOT WORK**
  + **Now I am getting an AV in IDSEARCH on *SymCursor := P^.SYMCUR***
  + **COMPINIT.INITSCAL the p-System debugger does not skip over the LPA at offset 67?**
  + **Does the LB 1.5 ever call LPA?**
  + **Version IV interpreter calls OpCode 208 *LDM***

**1/15/2024:**

* **IDSEARCH called by CSP**
* **When IDSearch is called, the IPC makes a big jump - actually this ok. It is an UJP to an XJP**
* **SYMBUFP = 472, SYMCURSOR = 2, @SYMBUFP^[SYMCURSOR+1] = 475; CH = garbage; SW = garbage**
* **I am uncertain about the location where *SY* is stored? Global variables shows it to be 18 but p-Code thinks that it is 16.**
* **SYMBUFP^[SYMCURSOR+1] contains a ‘w’ “(\*$warning otherwise false\*)”**
* **fWindowsList belongs to FilerSettings**
* **if Global Variables not displaying things correctly? Highlighted variable never changes?**
* **GlobalVar = 47672? After offsets to SymCursor => 47712 ($BA38). (Byte address)**
* **SymCursor is located at local address offset 15 (I think).**
* **Is GlobVar the same as base? NO.**
  + **GlobVar = 47672**
  + **base = 46863**
* ****

**1/16/2024:**

* **Version IV Insists that I halt the system first**
* **So does LB 1.5 if I close the p-System window manually**
* **Even if I cancel out of Load Version, it still tries to boot.**
* **Load Version is not sorting the list properly.**

**1/17/2024:**

* ***TESTDEBUG.CODE* does not show up in the *History* list**
* **Segment 29 (*TESTDEBU*) does show NOT up in the history**
* **I need to display the *SEGTBL***
* **Maybe *GetSegNum* should be returning a *TSegNameIdx* rather than a *word*?**
* **Where do I load the segment names from the loaded segment?**
  + **See: *DumpDebugInfo, SaveSegInfoForFile, CheckForSegDict***
* ***TESTDEBUG* never gets loaded into the debug info**
* ***UNDEF DumpDebugInfo***
* ***History* is NOT showing only the *Calls***
* ***CallHIstOnly -*vs*- CallHistoryOnly***
* ***TESTDEBUG SegTop* is 56058 (or 56060), fn (FileNumber) = 6, FirstBlock = 378**
* **It would be nice to have the *EditText* procedure use the editor specified in the *FilerSettings* rather than always using *NotePad*.**
* **Are the code addresses relative to the beginning of the file or to the beginning of the volume? I think that they indicate the starting block of the segment within the file.**
* **Calling sequence when the *SegmentInfo* gets saved:**
  + ***SaveSegInfoForFile ← CheckForSegDict ← CallIO ← CSPUnitReadWriteCommon ← CSPUnitRead ← CSP***

**1/18/2024:**

1. **When *DumpDebugInfo* is first called for TESTDEBUG.CODE, TheSegTop = 0**
2. **Executing TESTDEBUG.CODE a 2nd time does not even call DumpDebugInfo**
3. **Calling sequence when TESTDEBUG is loaded:**
   * **CSP→ CSPUnitRead→ CSPUnitReadWriteCommon→ CallIO→ CheckForSegDict→ SaveSegInfoForFile→ DumpDebugInfo**
4. **Calling sequence for when *SEGTOP* gets changed:**
   * **CXP→ GetSeg(1, FALSE)--> ReadSeg(“”, 56060)--> UpdateSegStuff(1,1,’USERPROG’,56060,5,379,148)**
     + **I.e., the SEGTOP gets changed when the segment gets loaded**
5. **Here is the culprit:** 
   * **fFilesLoadedList[6].Seginfo[1] = (TheREFCOUNT:0; TheSEGTOP:56060; TheSEGNAME:'USERPROG'; TheCodeAddr:1; TheCODEleng:148)**
6. **The reference count goes to 0 (which then sets the *SEGTOP* to 0)**
   * **RBP→ RNP→ DECREF(1, 56060). Now:**
   * **fFilesLoadedList[6].Seginfo[1] = (TheREFCOUNT:0; TheSEGTOP:56060; TheSEGNAME:'USERPROG'; TheCodeAddr:1; TheCODEleng:148)**
   * **This is effectively erasing the segment from known segments**
7. **After step 5, I should look for any attempt to get a segment/procedure name**
8. **After resetting in step 6, the SEGNAME has been changed to USERPROG and does not get change back**
9. **i changed *SaveSegInfoForFile* to always update the *SEGTOP* rather than only if it is new.**
10. ***SegIdxFromSegTop* is still not finding the segment.**
11. **TheSegTop has been set to 0 again. Does it EVER get changed from 0?**
12. **It would get changed in *UpdateSegStuff* or in *UpdateSegInfo* (which calls *UpdateSegStuff*)**
13. ***UpdateSegStuff* gets called by** 
    * ***ReadSeg* (during initialization)**
    * ***ReadSeg* (when exiting the *Filer* and returning to the OS)**
    * **When executing *testdebug***
14. **TheSegTop does get set to 56060 here:**
    * **CXP→ GETSEG(1, FALSE)--> ReadSeg(“”, 56060)--> UpdateSegStuff(1,1,’USERPROG’, 56060,5,379,148)**
15. ***TheSegTop* gets reset to zero *SaveSegInfoForFile***
16. **Added *FileIsKnown* to *SaveSegInfoForFile* to prevent *SegTop* from getting reset to 0. The breakpoint is still not breaking.**
17. **aCurrentSegName breakpoint in OldDebugger: (only two)**
    * **PASCALSY**
    * **GETCMD**
18. **In *CurrentSegName*, *SEGP* = 56060 yields a segment name ‘USERPROG’**
19. **TheSegTop never got set for TESTDEBUG! Overlapping segments?**

**If I close the p-System window first, I cannot close the debugger window because it insists that I must halt the p-System first**

**1/19/2024:**

* **Seems like *FileIsKnown* gets set too often?**
* **56060 is getting used for SYSTEM.FILER**
* **I am getting overlapping regions. fn = 0, idx = 5**
* ***‘TESTDEBUG’* gets overlaid with ‘*USERPROG’* with *UpdateSegStuff***
* ***CXP→ GETSEG(1)--> ReadSeg(1, FALSE)--> UpdateSegStuff(1,1,’’, ‘USERPROG, 56060,5,379,148)***
* ***LoadVersion.LoadDEBUGGERSettingsFile* destroys any previous copy of *TDEBUGGERSettings* which destroys the TWindowsList. It belongs to the FilerSettings. It should belong to the DEBUGGERSettings.**
* ***XfDEBUGGERSettings* is declared in *pCodeDebugger\_Decl***
* ***TDebuggerSettings* life story**
  + **a local copy is created in *LoadVersion***
  + **it is destroyed by *LoadVersion***
  + **It is created again by *rgVersionClick***
    - **It creates a *TWindowsList* as *fWindowsList***
  + **Another copy is created when TfrmPCodeDebuggerII is created**
* ***fDebugWindowsList* belongs to the debugger -NOT- to the settings files**
* ***DebugWindowsList* is -NOT- the same as WindowsList**
* **TfrmPSysDebugWindow.fDebugger is nil**
* ***DumpDebugInfo* should refresh the DashBoard**
  + **DumpDebugInfo: *frmPCodeDebugger.DebugWindowsList* is unknown**
* ***DumpDebugInfo* belongs to *TUCSDInterpreter* which is defined in *UCSDInterpreter***

**1/20/2024:**

* ***DebugWindowsList* is a public property of *TfrmPCodeDebuggerCustom.* No. It is not.**
* ***DebugWindowsList* belongs to *TfrmPCodeDebugger* which is a child of *TfrmPCodeDebuggerCustom***
* **This is the compile error code in *TUCSDInterpreter*:**

**with frmPCodeDebugger do**

**DebugWindowsList.;**

* ***DumpDebugInfo* should display the *notice* to the proper {renamed from DebugWindowsLIst to} DashboardWindowsList using *UpdateLatest(Notice). -OR-* Maybe the message should be part of the fFilesLoadedList[].SegInfo[] structure**

**1/22/2024:**

* ***SegTop* never gets set to non-zero for TESTDEBUG.CODE. No. It got set to 56060 after I halted the p-System.**
* **Does the system really think that this is USERPROG rather than TESTDEBU? Yes.The segment is listed as USERPROG.**
* **Is the OVERLAP test incorrect because the TOP address is beyondthe memory region?**
* **The second time that I run *TESTDEBUG*, the value for *SEGTOP* is set to 56060.**
* **Free segment info TESTDEBU because of b1: (OldB(55912) <= NewB(55912) <= OldT(56060))**
* **Free segment info TESTDEBU because of b2: (NewB(55912) <= OldT(56060)) <= NewT(56060))**

**1/24/2024:**

* **When I look at “fFilesLoadedList[6].SegInfo[1]” in the dashboard, I see:**

**FN= 6, UnitNr= 5, TheFileName=TESTDEBUG.CODE, TheAbsFileStartingBlock= 378**

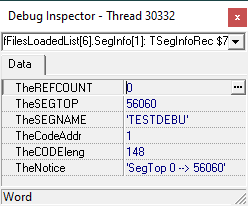
**# SegName, SegTop, Addr, Leng, Blks, 1stB, LastB, Refs, Notice**

**1 TESTDEBU, 56060, 1, 148, 1, 379, 380, 0, SegTop 0 --> 56060**

* **and when I look at it in the Delphi debugger I see:**

**(1, 56060, 'USERPROG', 1, 148, 'SegTop 0 --> 56060')**

**-or-**

****

**But AFTER running TESTDEBUG, the SEGNAME has changed to USERPROG**

* ***TheSegNameIdx* is calling *TErecPtr(@Bytes[SegBase])^* but *TErec* only exists in Version IV?**
  + **I commented out TErecPtr line with no obvious ill effects**
* **Initially 56060 links to “USERPROG” but I don’t see 56060 in the fFileLoadedList at all?**
* **Initially identified as “USERPROG”**
* **Can I use the BPT to identify the new program?**
* **CurrentSegName is the problem. Do I have access to the segment dictionary?**
* **Put a breakpoint in EXECUTE? SegTbl @ 49;**
* **The segment map shows the segment name to be “TESTDEBUG”**
* **SegNameFromSegTop finds a match for aSegTop = 56060 in fn = 4, but my DashBoard says that fn = 4 is FILEHAND - NOT USERPROG which is what if being found.**
* **CSPReleaseSeg is not implemented**

**1/24/2024:**

* **Get rid of the VersionNr parameter to the interpreters and replace it with BootParams**

**1/25/2024:**

* ***fBootParams* is defined in *TUCSDInterpreter***
* **How can I look at the p-Code generated for *TESTDEBU.code?* Using *DISASM.CODE.***
* ***function GetSegNum: TSegNameIdx; virtual; abstract;* -- This used to return a *word* rather than a *TSegNameIdx***
  + ***TSegNameIdx* is defined in *Debug\_Decl*  which should not even be compiled when compiling *FILER.EXE***
    - **I have changed it to return an *integer* rather than a *TSegNameIdx*.**
* **The Peter MIller version correctly runs *TESTDEBUG.CODE***
* **I need to watch the p-Code being generated for TESTDEBUG.CODE. Why is it so long?**
* **The first byte that is accessed from ? is a 2**
* **Overrides initialized:**

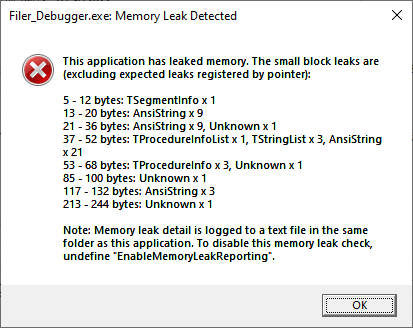
| **OnAddLine** | **AddLine** |
| --- | --- |
| **OnAddLineSeparator** | **AddLineSeperator** |
| **OnGetbyte3** | **GetByteFromMemory** |
| **OnGetWord3** | **GetWordFromMemory** |
| **OnGetJTtab** | **GetJTab** |
| **OnGetBaseAddress** | **GetBaseAddress** |
| **OnGetCPOffset** | **GetCPOffset** |
| **OnGetSegMentBase** | **GetSegmentBaseAddress** |

* **The base address that is passed in is: 112444 ($1B73C)**
* **IpcBase = 56174 ($DB6E). Note: IpcBase \* 2 = 112348 ($1B6DE) (about double)**
* **OnGetByte3 is what gets called by the decoder.**
* **The decoded p-System code offsets are incorrect. *DBBuiilder* needs to be recompiled and the database records for TESTDEBUG.TEXT need to be recreated from “I5-TESTDEBUG-Listing.txt”.**

**1/26/2024:**

* **What causes BreakpointInfo to be compiled?**
* **Where is the base definition for the *Debugger Settings*?**
* **Forms to be created:**
  + **frmFileParameters**
  + **frmUpdateConfirm**
  + **frmProcedureInfo**
* **I need to list the output file when the report is generated**
* **This is the outputfilename that was generated:**
  + **'F:\NDAS-I\d7\Projects\pSystem\Listings\V1.4\EDITOR-E4-VI4.txt\_I.4.CSV'**

**1/29/2024:**

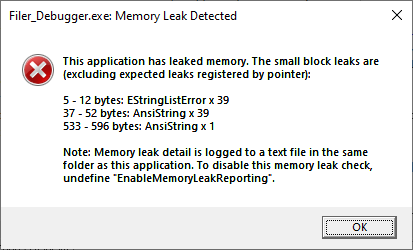
* **“Volume Containing the Code File” is not being prompted?**
  + **Include .RAW files**
* **Memory Leak**
* ****
* **TSegmentInfo - created**
  + **TProcedureInfo - 3 created : NEVER freed**
    - **Where does TProcedureInfo get created?**
      * **Each creates/contains a *TProcedureBody***
      * **Where does it get freed?**
    - **Does the *ProcedureBody* get destroyed?**
    - ***UpdateVolumeInfoTable* does not get called?**
* **Structure:**
  + **TSegmentInfoList is a *TStringList***
  + ***SegmentInfoList* is created in *TfrmBuildDebugDB* (and belongs to it)** 
    - **It gets destroyed when the form is destroyed**
    - **It contains items of TSegmentInfo type**
  + ***TSegmentInfo***
    - * ***SegmentNumber***
      * ***SegmentType***
      * ***Procedures {TProcedureInfoList which is a string list- each object points to a TProcedureInfo}***
    - ***TProcedureInfo***
      * + ***SegmentName***
        + ***SegmentNameFull***
        + ***ProcedureName***
        + ***ProcedureNumber***
        + ***ProcedureNameFull***
        + ***SegmentNumber***
      * ***ProcedureBody (TStringList)***
* **Does *TSegmentInfo* get destroyed?**
* **The *SegmentInfoList* should be destroyed when the form is destroyed.**

**1/30/2024:**

* **needs to write the variables in the order required by the VersionNr!**
  + **When fSegmentInfoList is destroyed, it ought to be destroying the one and only TSegmentInfo?**
* **How come the PM version calls CSP CSPUnloadSegment but the LB does not?**
* **The Global variables are not displaying properly**
* **Local variables appear in the same order as the compiler listing.**
  + **I need to add a function call to my test program**
* ***TWindowsList.FindNamedWindow* is nil**
* **Are old *WindowsList* items being deleted?**
* ***GlobalAddr: Base+MS\_VARW+Offset***
  + **GlobVar = $DB24**
  + **Base = $DB24**
* **MS\_VARW = 5**
* **The actual address for *Global1* is:**
* **The offset for Global1 is 1 (word?)**
* **The address for Global1 is $DB34 (offset = 3 words)**
* **The address for Global2 is $DB36 (offset = 4 words)**
* **When the DEBUGGERSettings are freed, the debugger windows list is destroyed**
  + **The LocalVariables window info is trying to be used AFTER the debugger settings have already been freed.**
  + **Where does the list of *Local Windows* get freed?**
  + **When does the DEBUGGERSettings get freed?**
  + **Confusion between the FILERSettings and the DEBUGGERSettings?**
  + **The Local Variables list is being destroyed by the system shutting down-- not by the debugger being destroyed?**
* **I still have a problem with “Halt the p-System first”**
* **Getting an “Odd Address Error” when I do ^L to bring up the Local Variables debugger window. It is trying to display a FIB. The debugger is trying to display some global FIBS. It can be ignored.**

**1/31/2024:**

* ***fThePSysWindow* is the original. It belongs to *FilerMain*.**
* ***frmPSysWindow* is just a copy stored in the interpreter.**
* ***TfrmPCodeDebugger.FormCloseQuery* is checking *frmPSysWindow***
* **Need to get a notification message back to the interpreter to *nil frmPSysWindow.* I now NIL frmPSysWindow in the pCodeDebugger *Notification* procedure.**
* **Whenever fThePSysWindow is set to NIL, that info must get passed to the interpreter *frmPSysWindow***
* **BuildDebugDB is updating the wrong records? Not true;**
* **Getting a memory leak when is update the database via *BuildDebugDB*.**

****

* **The p-Code is not synchronized to the source code. The p-Code needed to be erase when parsing the listing. I added an “Erase p-Code” checkbox to the Debugger Database Utilities dialog.**
* **LB Version still cannot recognize that the TESTDEBUG.CODE is executing.**
* **PM Version cannot compile -- still gets the exception on IDSearch.**

**2/1/2024:**

* **CSPUnloadSegment is called from segment TESTDEBU(?), CurProc = 42**
* **CspLoadSegment is also called from CurProc = 42**
* **I need to find the source for procedure 42.**
* **Executing TESTDEBUG calls *CspLoadSegment* to load SegNo = 1, the IPC = 101**
* **The LB version 1.5 NEVER calls *CSPGetSeg* or *CSPReleaseSeg*? This would seem to imply that the *InterpII* interpreter is expected to be updating something that the OS is using. *InterpII* does not have the smarts that the *InterpC* has (see, for example, the *load* function).**
* **Regarding IDSEARCH:**
  + **SymBufP = $472**
  + **RetnInfoP = $BA60**
  + **P := TRetnInfoPtr(@Bytes[RetnInfoP])**
* **or should it be -**

**P := TRetnInfoPtr(@Words[RetnInfoP])**

* **I fixed *IDSearch* for the version I.5 system by using the first alternative above “@Bytes[RetnInfoP]”**
* ***IDSearch* is called from *InSymbol***
* **Do I need to move *F:\NDAS-I\d7\Projects\pSystem\Listings\V4.0\PascalCompilerListing.TXT* to the V2.0 folder?**
* **Guess compiler listing level reports this line:**
  + **' 488 9 2:1 0 NEW(INTPTR,SCALAR,STANDARD);'**
  + **as an error because nesting level = 1 and offset = 0;**
  + **This occurs on the first line of every procedure.**
* **These lines error out (ERR\_PROCESSLISTINGLINE):**
  + **' 17 0 1:D 1 '**
  + **' 18 0 1:D 1 (\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)'**
  + **' 19 0 1:D 1 (\***
* **\*)'**
* **IDSEARCH error when I try to compile using Version II**
* **Remember to save the *Erase P-Code* checkbox.**

**2/2/2024:**

* **F:\NDAS-I\d7\Projects\pSystem\Volumes\UCSDII.RAW: comp20.code**
* **I want to DISASSEM this file (COMP20.CODE) under version II and send it to the CONSOLE:**
* **Boot from this volume:**
  + **F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\UCSDII.RAW because it has DISASSEM.CODE**
  + **COMP20.CODE is located here: [UCSDII0:] F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\UCSDII.RAW**
  + **JUNK.RAW is actually UCSD2:**
* **Ugly listing:**
  + **F:\NDAS-I\d7\Projects\pSystem\Temp\I5-TESTDEBUG2-Listing.txt →**

**F:\NDAS-I\d7\Projects\pSystem\Temp\TESTDEBUG.PAS**

* **# 4: SYSTEM: [ 2000] F:\NDAS-I\d7\Projects\pSystem\Volumes\Linux Vols\system.vol**
* **# 5: TEST: [ 1000] F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\TEST.RAW**
* **Z80 is looking for SYSTEM.MICRO or SYSTEM.INTERP**
* **Put a breakpoint at InSymbol, IPC:0**
* **I may be able to move SYSTEM.MICRO from** 
  + **F:\NDAS-I\d7\Projects\pSystem\Volumes\UCSDII.VOL**
  + **- to -**
  + **SYSTEM-II0.RAW**
* **Try to boot the Z80 using:**
  + **F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\SYSTEM-II0.RAW-- This does not work**
* **Cannot disassemble Version II.0 code**

**2/5/2024:**

* **IDSearch seems to have worked when executing the 5/16/2023 version of FILER.EXE**
* **I am nor getting err 101: “Identifier Declared Twice” when “Program” is parsed.**
* **Changes made to InterpC**
  + **“Temporary” defined**
  + **word IPC → property Ipc**
  + **Using the LB version of of *LPA***
  + **UJP changed to use *ShortInt* of *FetchUB***
  + **The *jump* function may have changed.**
  + **The *fjp* function has changed from**
    - **w := jump(p1);**
    - **-to-**
    - **w := jump(shortint(p1));**
  + **and from**
    - **p1: word;**
    - **-to-**
    - **p1: byte; // 1/9/2024: word --> byte**
  + **CSPUnload segment now has debugging code in it**
  + **I got rid of all of the AccDB stuff**
  + **GetAbsIPC changed from**
    - **result := ByteIndexed(IpcBase) + IPC;**
    - **-to-**
    - **result := IpcBase + IPC;**
  + **After reverting to LB\_VER, the text file appears to be garbled-- however--** 
    - **if I open the file from the filer / notepad, everything looks fine.**
    - **if I open the file in LB version 1.5, everything looks fine**
    - **If I open the file in PM 1.5, everything looks ok**
    - **It appears that PM version was trying to use the VT-52 CRT codes. Changing to the PM CRT codes everything looks OK.**
* **When trying to trace through InSymbol, The OPName is always showing SLDC0;**
  + **But the PCodeDebugger does appear to be tracing through the correct code**
  + **Likewise, the Delphi debugger does seem to be Fetching the correct operations**
  + **but, the debugger is not displaying the correct stuff -- it show the current opname as SLDC0 for every operation**
  + **The *Value* column of the registers Opcode for the OpName is one instruction behind. The *OpCode* numeric value is correct.**
  + **I made MemRdByte a virtual procedure in TUCSDInterpreter**
    - **I override it in InterpC**
      * **It will also need to be overridden in InterpII**
    - **I made GetIpcBase a virtual procedure in TUCSDInterpreter**
    - **I override it in InterpC**
      * **It will also need to be overridden in InterpII**

**2/6/2024:**

* **By changing *GetIPC* to *virtual; abstract* I now get an abstract error when *GetIpcBase* is called. I changed it to directly reference the symbols in InterpC.**
* **I could really use a properly decoded version of the version II compiler.**
* **Ways to go:**
  + **Try to get the decoder to work for PM V2.0**
  + **Use the Z80 system to de-compile the compiler**
    - **DISASSEM.CODE cannot be found-- maybe it was compiled with an incompatible compiler?**
    - **I cannot boot Z80 system from the Debian volume *system* because it needs *SYSTEM.MICRO* or *SYSTEM.INTERP*.**
  + **Try reverting selected pieces of InterpC to the 5/16/2023 version**
  + **Don’t I have some program which can scan .CODE files and update the database?**
    - **Yes. ScanCodeFileandUpdateDB. Comments say that this may not work for word addressed code.**
    - **Most of the decoded procedures appear to have the same decoded data (with names like *PROC06*…)**
  + ***ScanCodeAndUpdateDB* ought to change the default *ACCDB* name whenever the *VersionNr* changes**

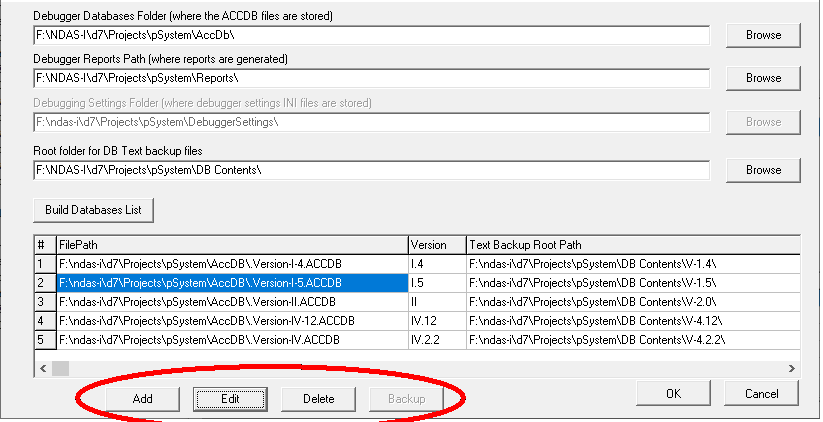
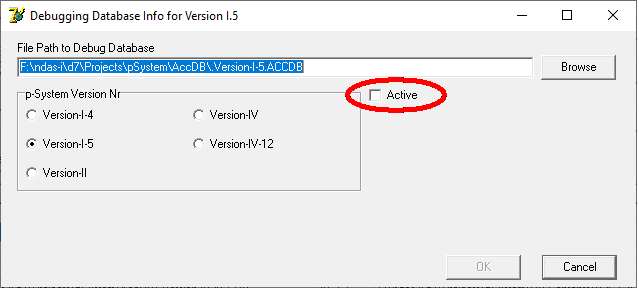
**2/7/2024:**

* **Try compiling with the UCSDII volume. Does it compile OK?**
* **Reference to *\*system.lst.text* in *F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\COMP-II.RAW***
* **In COMP: (COMP-II.RAW) RENAME *SYSTEM.COMPILER* to *OLD.COMPILER***
* **and rename NEW.COMPILER and see if it can compile**
* **‘Release level II.0g’ does not exist in OLD.COMPILER**
* **The NEW.COMPILER has the same problem as the old one: “error 101” on the word “program”**
* **Directory entry 10 in the volume F:\NDAS-I\d7\Projects\pSystem\Volumes\Peter W\360K.VOL always gets a range check error. The file is named “VT100GO”. I deleted it.**
* **The volume F:\NDAS-I\d7\Projects\pSystem\Volumes\COMP.VOL appears to be the compiler NEW.COMPILER.**
* **The volume F:\NDAS-I\d7\Projects\pSystem\Volumes\TEMP.VOL has a file called COMPLIST.TEXT which may be the compiler listing of NEW.COMPILER**
* **I made a copy of *Version-II.accdb* and saved it in the folder F:\NDAS-I\d7\Projects\pSystem\AccDb\Saved**
* **Trying to compile with the compiler on UCSDII0 gives the same problem (error 101)**
* **What version is NEW.COMPILER?**
  + **COMPLIST.TXT is version II.0g**
  + **COMPILER-VersionII-Listing(2) is iI.0g**
  + **COMPILER-VersionII-Listing(3) is iI.0g**
  + **COMPILER-VersionII-Listing(3-Clean) is iI.0g**
  + **COMPILER-VersionII-Listing(4) is iI.0g**
  + **COMPILER-VersionII-Listing(clean) is iI.0g**
  + **Compiler-VersionII-Listing(4).txt is iI.0g - This seems to match what is stored in the db**
    - **Booting with *F:\NDAS-I\d7\Projects\pSystem\Volumes\Linux Vols\system.vol* does seem to have the II.0g compiler as *SYSTEM.COMPILER***
    - **I need to decode the compiler to get the correct p-Code into the database.**
    - **Variables may not be in the correct order**
    - **<ctrl><alt>L Macro to fix listing**
    - **Notes about the database after updates (filtering on updated 2/7/2024)**
      * **COMPINIT looks OK**
      * **DECLARATIONPART OK**
      * **FINISHUP OK**
      * **PUTSYNTAX OK**
      * **UNITPART OK**
      * **WRITELIN OK**
      * **ENTSTDTY OK**
      * **GLOBALSEARCH NO**
      * **INCOMMA OK**
      * **OPENREFFILE OK**
      * **PUTNUM OK**
      * **ENTSTDNAMES ok (but weird code- 1 byte put onto each word of stack)**
      * **GEN0 OK**
      * **GETNEXTBLOCK OK**
      * **INITUNIT no**
      * **INTCHK ok**
      * **TYP OK**
      * **ENTUNDEC OK**
      * **GENLDC OK**
      * **GLOBALSEARCH OK**
      * **INITUNIT OK**
      * **STRGVAR OK**
      * **ADDNAMES OK**
      * **NEWSTMT OK**
      * **PACKABLE OK**
      * **FIELDLIS OK**
      * **BODYPART NO**
      * **GETNAME OK**
      * **INSYMBOL ok?**
      * **MOVE ok**
      * **ALLOCATE ok**
      * **EXIT ok**
      * **ENTSPCPR OK maybe**
      * **ENTSTDPROCS ok**
      * **UNITIO ok maybe**
      * **VARIANTLIST ok**
      * **CONCAT ok**
      * **INITSCAL ok**
      * **LOADADDR ok**
      * **POINTERTYPE ok**
      * **COPYDELETE ok**
      * **INITSETS ok**
      * **SELECTOR ok**
      * **USESDECL ok**
      * **BYTEADDRESS ok**
      * **STR ok**
      * **USE ok**
      * **GETTEXT mostly ok?**
      * **FOUNDUNIT ok?**
      * **GETPUTETC ok**
      * **DECSIZE OK**
      * **SCANDICT ok**
      * **SCANSCAN OK**
      * **BLOCKIO OK**
      * **GENBIG OK**
      * **LIBINFO OK**
      * **COMPTYPES YES**
      * **ADDLIB OK**
      * **ASSIGN OK**
      * **SEGINFO OK**
      * **SIZEOF OK**
      * **COROUTIN OK**
      * **GENJMP OK**
      * **LABELDECLARATION OK**
      * **GENBYTE OK**
      * **CONSTDEC OK**
      * **GENFJP OK**
      * **GENLABEL OK**
      * **TYPEDECL OK**
      * **FINISHSEG OK**

**2/8/2024:**

* + - * **PUTLABEL**
      * **VARDECLARATION**
      * **BLOCK OK**
      * **PROCDECL OK**
      * **STORE OK**
      * **PARAMETERLIST OK**
      * **COMPILER OK**
      * **STRGTOPA OK**
      * **CALL OK**
      * **CHECKEND OK**
      * **COMMENTER OK**
      * **READ OK**
      * **SCANSTRING**
      * **WRITE OK**
      * **CALLNONSPECIAL OK**
      * **USERLOAD**
      * **FLOATIT OK**
      * **SIMPLEEXPRESSION OK**
      * **TERM OK**
      * **FACTOR OK**
      * **GETROUTINE OK**
      * **MAKEPA OK**
      * **STATEMENT OK**
      * **ASSIGNMENT OK**
      * **GOTOSTATEMENT OK**
      * **COMPOUNDSTATEMENT OK**
      * **IFSTATEMENTOK**
      * **CASESTATEMENT OK**
      * **REPEATSTATEMENT OK**
      * **WHILESTATEMENT OK**
      * **FORSTATEMENT OK**
      * **WITHSTATEMENT OK**
      * **BODY OK**
      * **LOADSEGS OK**
      * **GETROUTINE OK**
      * **REMEMBER DISASM.II.CODE**
      * **I have been working from Listings\V2.0\Compiler-VersionII-Listing.TXT**
      * **There are duplicate records that need to be reconciled**
        + **WRITELINE needs p-Code to be updated**
        + **UNITDECLARATION is missing**
      * **Change generated procedure names like “PROCxx” to use the procedure number**
* **DISASM.II crashes in PUSH() with a range check error- trying to set the SP to 5466811**
* **Similar problem occurring in SLDC when SetSP($FFFF) is called (SLDC, PUSH(32), SetSP($FFFF)**
  + **Call history:**
    - **'102967=SLDC32 (ROUTINE. @725),102966=SLDC32 (ROUTINE. @724),102965=SLDC32 (ROUTINE. @723),102964=SLDC32 (ROUTINE. @722),102963=SLDC32 (ROUTINE. @721),102962=SLDC32 (ROUTINE. @720),102961=SLDC32 (ROUTINE. @719),102960=SLDC32 (ROUTINE. @718),102959=SLDC32 (ROUTINE. @717),102958=SLDC32 (ROUTINE. @716),102957=SLDC32 (ROUTINE. @715),102956=SLDC32 (ROUTINE. @714),102955=SLDC32 (ROUTINE. @713),102954=SLDC16 (ROUTINE. @712),102953=LPA (ROUTINE. @711),102952=LAO (ROUTINE. @708),102951=SRO (ROUTINE. @706),102950=SLDC0 (ROUTINE. @705),102949=FJP (ROUTINE. @655),102948=LAND (ROUTINE. @654)'**
  + **Somehow the SP get set to 0. It gets set to 0 @ IPC = 724 in procedure 21. When WordIndexed adds -1, the SP gets set to $FFFF. This is happening in DISASM.II.CODE.**
  + **When I try to decode from the external window, it goes into an infinite loop of SLDC 0**
  + **The *global variables* window is not displaying the globals values**
  + **TESTDEBUG is not in the II.0 database But it is in the Version I.5 database**
  + **I’ve got some file name that has a ‘?’ in it. It was in procedure 9 in the TESTDEBUG records (‘GLOBAL?’)**
  + **I have multiple(1) copies of the database textfiles. Try to save only the latest copy.**

**2/9/2024:**

* **TDatabaseInfo can probably just be a record rather than a CLASS- not necessary-- I can continue to use the DatabasesList**
* **Q: Where do I get the filename for the DataBasesList**
  + **A: from FileNames DataBaseSettingsFilesFolder + DATABASE\_INI // = 'DATABASE.' + INI\_EXT;**
* **I need to verify the functions in the unit *FILENAME* function as they are expected to. Maybe I can just make sure that the older functions have not been changed (but some of them have been changed)- maybe I can create some temp copies of the old versions and compare the results with the new versions.- None of the old procs have been changed. I think there is no problem here.**
* **Canceling out of the DatabaseList dialog will have saved the list anyway. Fixed?**
* **Reinstate the ability edit DatabaseInfo**
* ****
* ****

**2/12/2024:**

* **When decoding a .CODE file, GetByteAt uses fOnGetByte3(fBaseAddress + p)** 
  + **fBaseAddress = 793**
  + **p = 3**
  + **which calls BytePtr(fBuffer + IPC)^**
    - **where fBuffer points to where the code file was loaded**
    - **and IPC = 797 (= fBaseAddress + p + 1)**
* **When TpCodeDecoderII is created, the following methods get assigned:**
  + **fOnGetWord3**
  + **fOnGetCPOffset**
  + **fOnGetSegmentBase**
  + **fOnAddLine**
  + **fOnAddLineSeparatorProc**
  + **fOnGetByte3**
  + **fGetBaseAddressFunc**
    - **Note1: all of the methods shown in gray have the same address?**
    - **Note2: this decoder is created when the debugger is created**
* **Another TpCodeDecoderII is created in TfrmDecodeWindow. The following methods get assigned:**
  + **fOnGetWord3**
  + **fOnAddLine**
  + **fOnAddLineSeparatorProd**
  + **fOnGetByte3**
  + **The *BaseAddress* passed into the constructor is 0**
* **When decoding from memory (*external decoder window*), *GetByteAt* uses *fOnGetByte3(fBaseAddress + p)***
* **Starting address is defaulted to $FD92 (64914) (this is the ABSIPC) (which looks like a word address)**
  + **fBaseAddress = 2792**
  + **p = 0**
  + **which calls BytePtr(fBuffer + IPC)^**
    - **where fBuffer points to where the code file was loaded**
    - **and IPC = 797 (= fBaseAddress + p + 1)**
* **When decoding from the *external* window, the fOnGetByte3 method. which just calls *GetByteFromMemory* which just uses Bytes[p]**
  + **Bytes at IPC: Bytes @ $FD92 [64914]: 9F CC 36 CD 04 01 C2 2B CA 36 9F D4 03 CD 04 01 CA 36 9F D3 F6 D6 10 00 0E 02 00 00 07 00 1E 00 01 FF 58 0A 32 10 90 12 82 11 44 1B 82 1B 80 1D 32 00**
  + **When entering INITIALIZE, here are the bytes:**
  + **Bytes @ $EC92 [60562]: B6 01 36 9F C3 AB 01 A5 17 A6 03 3F 3F 3F AA 03 A5 19 D7 A6 03 4A 61 6E AA 03 A5 1B D7 A6 03 46 65 62 AA 03 A5 1D D7 A6 03 4D 61 72 AA 03 A5 1F D7 A6**
* ***TfrmBuildDebugDB.ScanCodeFileandUpdateDB***
* **The memory leak occurred when trying to decode all of the p-Code files on SYSTEM.VOL**

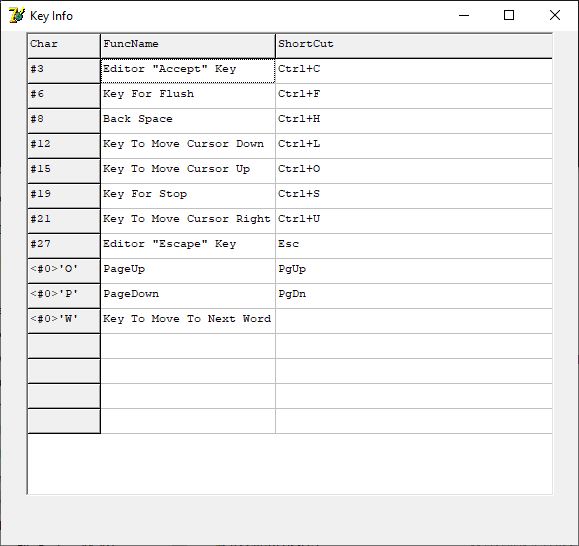
**Memory dump was saved to: *Memory Error Reports/Filer\_Debugger\_MemoryManager\_EventLog(2-12-2024 13-48).txt.***

**Decode .CODE to p-Code also does not seem to be working (sometimes?).**

* **This version: *G:\Location 46 - p-System\[1160] p-System 5-16-2023\pSystem\AccDb\VersionII - UCSDII.accdb*** **appears to have valid p-Code for PASCALSY (10/18/2022) (but probably out-of-date source) although the p-Code and source code for the compiler might be at least partially correct.**
* **Conclusions:**
  + **The external decoder window might be working correctly**
  + **The decode code file code status is unknown**

**2/13/2024:**

* **Booted into Debian on the Surface pro**
* **I had to re-do the Debian network connection:**
  + **Adapter 1**
  + **Enabled**
  + **Bridged adapter**
  + **Marvell Avastar Wireless-AC Network Controller**
  + **promiscuous mode: allow all**
  + **cable Connected: checked**
* **TEST.VOL, when viewed from Debian, appears to be be “K(runched” and has TESTDBII.CODE**
* **When viewed from XPS-8930\Debian\public, none of the changes were visible.**
* **On Debian, I copied TEST.VOL to testx.vol, gave it permissions using**
  + **chmod ugo=rwx testx.vol and then I could see the changes on the windows side.**
* **When I try to execute TESTDBII.CODE using the PM V2.0 interpreter, I get this error:**
  + **“Illegal call to native code from ProcNr = 8, IPC = 195, Linking needed?"**
  + **Possibly attempting to call DECOPS**

****

* **The TESTDEBUG listing compiled on Debian using the PM version II compiler was saved here: F:\NDAS-I\d7\Projects\pSystem\Listings\V2.0\TESTDBLIST-II.TXT**
* ***frmFileParameters* IS NIL after running *Compiler Listing Utilities - Update Database from Listing***
* **When updating the DB for TESTDBII most of the procedure names did not get recorded and the source code is missing or wrong.**
* **The .CODE file is located here: \\Debian\public\testx.vol**
* **All of the procedures are being give a *ProcedureNumber* of 1. Version II listing was not scanned properly.**
* ***frmFileParameters* being freed inappropriately**
* **frmFileParameters may be create both in the *Create* constructor and in the *Create2* constructor which will lead to memory loss**
* **The LSAs do appear to have the correct NOP preceding the string**
* **The source code extraction seems to quit before reaching the “END.” statement**
* **Here is how I got into the p-System (on Debian):**
  + **~/bin/ucsdpsys\_vm -w ~/Downloads/system.vol -w ~/public/testx.vol**
  + **password for Debian is “quito”**

**2/14/2024:**

* **The listing on Debian is being written to TESTDBII.TEXT**
* **The source code is still file TESTDEBUG.TEXT on the Debian version of TEST:**
* **The code is still file TESTDEBUG.CODE on the Debian version of TEST:**
* **The listing has been saved to TESTDBLIST-II.TXT in the LISTINGS folder**
* **When I try to run the II p-System after do the decoding stuff, I get:**
  + **'NIL pointer reference: JTab = 64930, @ IPC: 126'**
* **trying to backup files give an error message:**
  + **Could not find database: F:\ndas-i\d7\Projects\pSystem\AccDB\.Version-II.ACCDB**

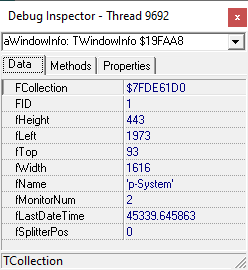
**2/15/2024:**

* **Currently working on INSYMBOL at line 526: GETSTMTLEV := TRUE but *globals* variables list it as FALSE?**
* **IPC = 4: Debugger show STMTLEV as 123, but registers show it as**
* **Maybe only functions reserve the extra 2 words but procedures don’t? Maybe it depends on the number of variables passed?**
  + **How do I know whether I am displaying variables for a *function* or for a *procedure*?**
    - **Maybe from *ParamSize* or *DataSize*?**
    - **function TCPsystemInterpreter.GetEnterIC(JTab: word): word;**
    - **begin**
    - **// WARNING: JTab may be an Word\_Memory address**
    - **result := (JTab-ENTERIC\_OB) - WordAt[JTab-ENTERIC\_OB]; // return EnterIC**
    - **end;**
  + **with fInterpreter as TCPsystemInterpreter do**
  + **begin**
  + **ParamSize := ProcParamSize(JTab);**
  + **DataSize := CurrentDataSize;**
  + **result := WordIndexed(GlobVar, MS\_VARW+2);**
  + ***{+2 reserves 2 words for something? I'm not sure what.***
  + ***Function results maybe?};***
  + **end**
  + **ParamSize always seems to be 0, 2 or 4 (or 6?)**

**2/16/2024:**

* **BaseRegister = $E831**
* **fWindowType = ftLocal**
* **The local variables seem to be displaying correctly in *InterM1*.**
* **I don’t think that MemDump of bytes is working correctly- does not agree with the ASCII MemDump**
* **Global2 (char) is located at $EC15**
* **if lblVarBase is set to $EC11, things seem to line up correctly**
  + **$EC11 is GlobVar ($EC0C) + 5 == GlobVar + MS\_VARW**
* **MemDump is not displaying floating point numbers correctly**
* **I am not getting a breakpoint on TESTDEBU.TESTDEBU?**
* **When I first enter TESTDEBU.TESTDEBU, The Global Variables window claims that the global variables ar located at $EDA2 (but the debugger registers window claims that GLOBVAR is located $EC0d, a difference of $0195**
* **The first call to OldDebugger lists the CurrentSegName as TESTDEBU**
  + **I get the first break at TESTDEBU IPC = 0 when entering the main program**
  + **I get a BPT call when**
* **When I first enter TESTDEBU.INTERM1, The Global Variables window claims that the global variables are located at $EDA2 (but the debugger registers window claims that GLOBVAR is located $EC0C, a difference of $0196? What is the value of CurrentDataSize? $CC06?**
* **Q. What am I trying to do?** 
  + **A1. Figure out why I am not getting a breakpoint on each TESTDEBU procedure.**
  + **A2. Figure out why the *Global Variables* window has such a weird value for *Variables*.**
* ***BaseRegister* = $EC0C → $EC11 (when *MS\_VARW* is added in)**
* **The value for GlobVar displayed in the debugger will be a *word* address.**
* **History:**
  + **Enters TESTDEBU**
  + **Steps into INTERM1**
  + **Steps into InnerProc but does not step out of InnerProc? When the RNP is called, control does not return to the calling procedure? ReturnOps = [173, 193, 214].** 
    - **Note1: NextOpCode = 0;**
    - **Note2: NextOpcode := Bytes[AbsIPC] // ⇐ this may not be correct**
    - **Note3: SameProcedure() = false**
* **When I dump HexBytes the address used is $EC13**
* **When I dump ASCII the address used is $1D826**
* **DashBoard memory dump does not seem to dump either the Bytes column OR the ASCII column - example: $EEB0**
  + **The *words* column is using *word addressing***
  + **The first row of the bytes column seems to be OK**
* **Using the dashboard messes up the pSystem size and location**

**2/17/2024:**

****

**Width should be 660, Height should be 468**

**The settings in FILER.INI are incorrect.**

**2/19/2024:**

* **GetNextOpcode uses**
  + **with fInterpreter as TUCSDInterpreter do**
  + **result := Bytes[AbsIPC]; AbsIPC=60572**
* **but FETCH uses FetchUB, which does**
  + **result := MemRdByte(IpcBase, IPC); IpcBase+IPC=60572**
* **the memory at that location appears to be:**
  + **@ $EC9C [60572]: 06 CC 05 CE 07 AD 00 02 0E 00 00 00 07 00 24 00 06 01 D5 45 CE 06 B6 01 03 A6 08 47 6C 6F 62 61 6C 31 3D D7 00 CD 00 13 9E 00 B6 01 03 EA 00 CD 00 0D'**
* **or, possibly:**
  + **' @ $0001D938 [121144]: 98 2D 00 00 A4 2E 40 00 5F 4D 40 00 3A 51 40 00 7A D5 49 00 96 E7 49 00 6C E3 49 00 A8 B7 42 00 DD B9 42 00 C1 AE 42 00 93 BC 42 00 FC BB 42 00 24 33'**
* **What is the byte sequence of the code for InnerProc?**
* ***NextOpCode* is always wrong.**
* **I moved GetNextOpCode into the interpreter and it seems to be working OK now.**
* **pCodeDebugger line 6046 - use NextOpCode instead**

**2/20/2024:**

* **TESTDEBUG.TEXT: Floating point number not properly displayed in the debugger**
* **External decoder window fails to decode when using option 2.**
* **The string assignment is occurring in COMPINIT**
  + **SEGTABLE[SEG].SEGNAME := ID;**
  + **The Src is located at 112950**
  + **The Dst is located at 113624**
  + **The SegTable ought to start 4 bytes before this, i.e., at 113620**
  + **SegDictFormat, TI5SegTblPtr**
* **The Scan function is not making any sense. When I MemDump the memory, I see:**
  + **' @ $0001C469 [115817]: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80'**
* **but when i look at Bytes[Addr+i]**
* **I get $53, $59, $53, $54 … = ‘SYSTEM.STARTUP’**
* **The first call is scanning ‘SYSTEM.STARTUP’ looking for a colon (‘:’)**
* **The second call is scanning ‘SYSTEM.STARTUP’ looking for a left-bracket (‘[’)**
* **The 3rd call is scanning ‘SYSTEM:SYSTEM.ASSMBLER’ looking for a ‘:’ and then ‘[‘, etc.**
* **After *IDSEARCH* has been called, it ought to have updated:**
* **SYMCUR : integer; { CURSRANGE }**
* **SY : TSYMBOLTypeII; { needs to be on a word boundary }**
* **fill0 : byte;**
* **OP : TOperator; { needs to be on a word boundary }**
* **fill1 : byte;**
* **RETTOK : TAlpha; { needs to be on a word boundary }**
* **After scanning ‘program’,** 
  + ***SymCursor* is 29.**
  + ***SY* is *PROGSY***
  + ***Op* is *NOOP***
  + ***RetTok* is ‘PROGRAM ‘**
* **When the debugger trys to display ‘PROGRAM’ it is only displaying ‘PROG’ (wt\_Alpha)**
  + **I don’t find any code to display wt\_Alpha? wt\_Ascii seems to be trying to handle wt\_Alpha.**
  + ***ParamType* is *wt\_Alpha***
  + ***AliasType* is *wt\_Ascii***
  + **Where is ParamSize = 4 coming from?**
    - **It is coming from *GetWatchTypeSize* because it assumes that a Word count is required. This is correct(?) for wt\_Integer,**
  + **wt\_Ascii, wt\_Alpha still only display 4 characters rather than 8**
* **main.c is using 2 word reals**

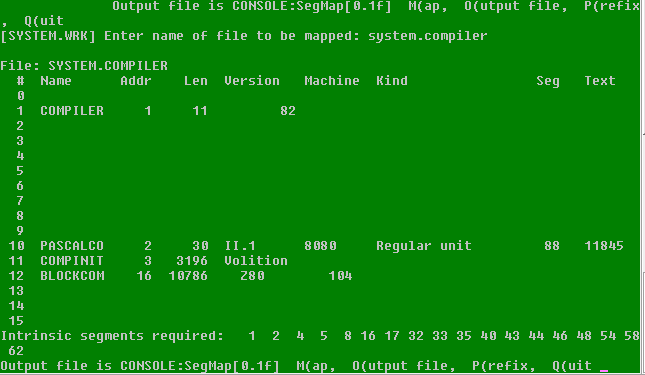
**2/21/2024:**

* **Compiler gets error 104 (*undeclared identifier)* after reading “integer”**
* **This occurs after the call to IdSearch for “INTEGER”**
* **The integer type should be declared in the compiler procedure ENTSTDNAMES**
* **PM\_VER -vs- LB\_VER: LPA is not working correctly**
* **The SLDC’s following an *LPA* aren’t really loading anything onto the stack*,* they simply place a sequence of bytes after the LPA that is a string constant.**
* **Changing LPA to the PM\_VER and changing the code to Push(IpcBase + (Ipc div 2)) seems to have fixed the compiler.**
* **LocalVariables not displaying floating point numbers correctly?**
* **The compiler type *TYPES* should be taking 9 words.**
* **lclCPptr : ^hexbytes[18] @ 8; is not displaying 18 bytes, only 1**
* **Executing *TESTDEBUG* when compiled on the PM Version 2.0 compiler, gives a RCE on the *JUMP* instruction**
* **When I try to execute the TESTDEBUG.CODE that was compiled by PM 2.0 on Delphi, I get a NIL pointer reference.**
  + **I renamed that version to TESTDBX.CODE**
  + **When i recompiled it to TESTDEBUG.CODE after compiling on Debian, it executes without error.**
  + **The two compiled versions are the same length (4 blocks)**
  + **Copied file TESTDBX.CODE to F:\NDAS-I\d7\Projects\pSystem\Temp\TEST\TESTDBX.CODE**
  + **Copied file TESTDEBUG.CODE to F:\NDAS-I\d7\Projects\pSystem\Temp\TEST\TESTDEBUG.CODE**
  + **The two code files do not look similar at all? Perhaps the compiler being used is not the same one? Yes. It was two different versions of the compiler.**
  + **I am going to try and use the compiler “\\Debian\public\system.vol” for both. This shows a date of 5-Oct-22. Debian will not agree since it overwrites the date from the OS. TESTDB2.CODE executes correctly on both OSs.**
    - **TESTDB2.CODE was compiled on Debian (PASCAL Compiler [II.0.A.1])**
    - **TESTDB3.CODE was compiled on Delphi (PASCAL Compiler [II.0.A.1])**
    - **The two code files now look (nearly?) identical. Here are the differences (reported to *C:\temp\Differences20240221.txt*):**
    - **BINARY FILE DIFFERENCES @ 2/21/2024 3:50:07 PM**
      * **File #1: F:\NDAS-I\d7\Projects\pSystem\Temp\TESTDB2.CODE**
      * **File #2: F:\NDAS-I\d7\Projects\pSystem\Temp\TESTDB3.CODE**
      * **#. Addr: Bytes**
      * **---. ----: -----**
      * **1. 206: D9**
      * **: E0**
      * **2. 226: D9**
      * **: E0**

**2/22/2024:**

* **To run the same version of *system.vol* as Delphi:**
  + **~/bin/ucsdpsys\_vm -w ~/public/system.vol -w ~/public/testx.vol**
* **DISASM.II.CODE requires that #4: must be the default directory**
* **The only difference between TESTDB2.CODE and TESTDB3.CODE is that the stored value for PI is different.**
* **While executing TESTDEBUG.CODE:**
  + **The listing matches**
  + **The executing program matches**
  + **(q) But the p-Code doesn’t? How is that possible?**
  + **(a) Because the system was executing a previous version := TESTDEBUG.CODE-- I.E. It had not been saved!**
* **Still having problems when trying to execute a 2nd time without completely exiting to the outside world**
* **Still cannot compile using PM VersionI.4**

**2/23/2024:**

* **Can I make use of SelfRelPtr? This is probably irrelevant.**
* **Version PM 1.4 (*F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\UCSDI4.RAW*) gets an RCE in "*jump*" when trying to compile. The *jump* has already succeeded 1048 times. Before the crash:**
  + **ODS: jtab=CD6E, Disp=000E, WordIndexed(JTab, -1)=CD6C, MemRd(JTab - Disp)=AD17**
* **JTab (CD6E) is out of line with all previous values.**
* **MemRd(JTab - Disp) ($AD17) is also out of line with all previous values (~$00CD)**
* **I put a breakpoint on $CD6E in SetJtab. This is occurring in CLP. The call stack doesn't give any useful information (everything is listed as Unknown\_53138, etc). The history shows that It appears to be occurring following a call to RBP in PASCALSYS.FINIT.**
* **Problems with debugger display:**
  + **FINIT: STO stores a 1 at address $699D (which should be the address of EOF?)**
  + **But the global variables are at $CF80 and the FIB is at $D336**
  + **I get a RCE when the debugger tries to display the segment number**
* **JTab set badly shortly after exiting RBP in FINIT. DbgCnt = 54568.**
  + **Last opcode was probably CLP**
  + **Crash occurs following the 3rd (2nd?) call to FINIT following the *C(ompile* command. Last opcode is CLP.**
  + **I am getting the JTAB (Delphi) breakpoint before the UpdateSegInfo (DiskAddr = 0) breakpoint.**
  + **Trying to boot with the *“F:\NDAS-I\d7\Projects\pSystem\Volumes\Linux Vols\system.vol”* using the LB 1.4 system goes immediately to an Odd address error.**
  + **CSP\_LoadSegment & CXP both call CspLoadSegment**
  + **The files are actually being loaded from:**
    - **F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\UCSDI4.RAW**
  + ***SEGMAP* only shows the compiler having 1 (?) segment which is called “*compiler”***
  + **Mounting F:\NDAS-I\d7\Projects\pSystem\Volumes\V1-5\UTILS.SVOL so that I can get SEGMAP.CODE**
  + **Trying to execute SEGMAP.CODE (after “Cleaning for pre-V4 code) gives:**
  + **Execute what file? #5:SEGMAP**
    - **No proc in seg-table**
    - **S# 0, P# 42, I# 1143**
    - **Type <space> to continue**
  + **i cannot compile SEGMAP.TEXT on version I.4 because it wants to US*E* *SCREENOPS*.**
  + **Control characters for version I.4: Everything must be in upper case:**
    - **Right ^D**
    - **Left <CR>?**
    - **Up ^E**
    - **Down ^R**
    - **Editor Accept: ^Z**
  + **Edited SEGMAP.TEXT on ZEM2010\UTILS.RAW to get rid of (“\_*”) illegal in Version 1.4 and comment out the SC*\_Screenops procedures. When I do this, SEGMAP appears to have 4 segments:**
  + ****
  + **but my version of SEGMAP does not see the last segments (10, 11, 12)**
  + **The code length displayed in the SEGMAPped file appears to be given in bytes rather than words.**
  + **The *Major Version* and *MachineType* appear to be nonsense.**
  + **The report was written to: F:\NDAS-I\d7\Projects\pSystem\Reports\SEGMAP-UCSDI4-SYSTEM.COMPILER (5).CSV**

**2/24/2024:**

* **When SYSTEM.COMPILER gets loaded, the segment info is not being retained.**
* **I am booting F:\NDAS-I\d7\Projects\pSystem\Volumes\Linux Vols\system.vol.**
  + **1st block checked is 292 which is in SYSTEM.PASCAL?**
  + **Appears that I am actually booting F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\UCSDI4.RAW**
  + **Now loading SYSTEM.COMPILER**
    - **Appears to have gobbedly-gook for segment names *(DISKINFO.SEGNAMES[n])*? (But when I did a SEGMAP on this (?) file it seemed OK?**
    - **When I break in SaveSegInfoForFile (when the compiler is being loaded)**
      * **DiskInfo looks OK**
      * **SEG\_NAME IS INCORRECT**
      * **SEG\_MISC IS INCORRECT**
      * **SEG\_TEXT/TEXTADDR is incorrect**
    - **When I look at the raw file using UEDIT32, the *segnames* look ok.**
    - **The UNITREAD that reads SYSTEM.COMPILER is only reading the 1st 64 bytes! I.e., it is only reading the *disk\_info*.**

**2/25/2024:**

* ***TUCSDInterpreter.ValidDictionary* is called by *SaveSegInfoForFile***
* ***ValidDictionary* is also called by *OverrideSysCall* in InterpII. ValidDictionary is where I want to put my overriding read of the segment dictionary.**
* **CSPUnitReadWriteCommon (in InterpC) → CallIO → UNITBL[UNUM].Driver.Dispatcher(UREQ, UBLK, ULEN, Bytes[UBUF], control)**
* **Is the XPS-8930 account for connections actually** [**dhdorrough@hotmail.com**](mailto:dhdorrough@hotmail.com) **rather than** [**dhdorrough2@hotmail.com**](mailto:dhdorrough2@hotmail.com) **? Yes. I think so. Password duluth03**

**2/26/2024:**

* **The segment information for the compiler is not getting loaded.**
* ***ValidDictionary* does not think that it is a valid segment dictionary**
* **When I am in *ValidDictionary*, there seems to be an invalid exit from ValidDictionary followed by a re-entry to *ValidDictionary.***
  + **CODELENG is in bytes-- not in blocks!**
* **The segment dictionary for the compiler has been loaded by DbgCnt = 36079**
  + **SaveSegInfoForFile is trying to use the invalid DICT info.**
* **DbgCnt = 36528 → 45859 when the compiler crashes:**
  + **'Dynamic CallStack:**
    - COMPINIT.COMPINIT @ 2,
    - COMPINIT.COMPINIT @ 0,
    - PASCALCO.COMPINIT @ 0,
    - COMPILER.COMPINIT @ 0,
    - PASCALSY. @ 0,
    - PASCALSY.COMPINIT @ 0'
  + **History: '**
    - **45859=CLP (COMPINIT.COMPINIT @1),**
    - **45858=CXP (PASCALCO.RLOCSEG @25),**
    - **45857=RBP (PASCALSY.FINIT @82),**
    - **45856=UJP (PASCALSY.FINIT @73),**
    - **45855=STO (PASCALSY.FINIT @72),**
    - **45854=SLDC0 (PASCALSY.FINIT @71),**
    - **45853=INC (PASCALSY.FINIT @69),**
    - **45852=SLDO4 (PASCALSY.FINIT @68),**
    - **45851=STO (PASCALSY.FINIT @67),**
    - **45850=LDCN (PASCALSY.FINIT @66),**
    - **45849=SLDO4 (PASCALSY.FINIT @65),45848=FJP (PASCALSY.FINIT @63),45847=LESI (PASCALSY.FINIT @62),45846=SLDC0 (PASCALSY.FINIT @61),45845=SLDO1 (PASCALSY.FINIT @60),45844=FJP (PASCALSY.FINIT @35),45843=LOR (PASCALSY.FINIT @34),45842=EQUI (PASCALSY.FINIT @33),45841=NGI (PASCALSY.FINIT @32),45840=SLDC2 (PASCALSY.FINIT @31)'**
    - **No attempt to read TESTDEBUG.PAS has been made yet**

**2/27/2024:**

* **As soon as I try to step into *INITSCALARS* from *COMPINIT*, I get the JTAB crash.**
* **NewSeg = $CF92**
* **JTab getting set at crash: $CD6E**
* **Where does SEGP get set to $CFC4? Or is it $CF92?**
* **Does PM 1.5 work ok?**
  + **When I try to compile using PM 1.5, I get an odd address error in COMPINIT**
* **Why is there a 32 byte difference in the value of NewSeg inside of *call*?**
  + **In CLP SEGP = $CF92**
  + **On entry to *call*, NewSeg = $CF92**
  + **Now I do not see $CFC4?**
* **LB 1.4 compiles fine**
* **Segmap of SYSTEM.COMPILER shows that *BLOCKCOM* wants 21572 bytes for code**
* **I am not finding the source code for *BLOCKCOM* anywhere**
* **No evidence that *BLOCKCOM* segment ever gets loaded**
* **In version I.4, the *FILER* seems to be baked into *SYSTEM.PASCAL***
* **Why does my debugger think that it has loaded procedure proc#17? “COMPINIT.” I find no evidence of a procedure #17.**
* **There is a call to *S1P*? Is *S1P* the same as *LPA*?**
* **There is a call to RLOCSEG**
* **Version II compiles OK. F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\SYSTEM-II0.RAW**
* **The JTAB looks like this:**
  + **' @ $ED8B [60811]: ProcNr=1, LexLevel=0, EnterIC=$1DB14, ExitIC=$1DB12, ParamSize=0, DataSize=0, LastCode=0'**
  + **We fetch the word BEFORE the JTAB**
  + **Add 2 to it**
* **When I try to run the PM 1.4 version, I now get a very immediate Odd address error on a STO opcode (DbgCnt = 88)?**
  + **2/23/2024: F:\NDAS-I\d7\Projects\pSystem\Volumes\Linux Vols\system.vol It Booted and halted**
    - **2/27/2024: OK until sometime today**
  + **The LB 1.4 version using F:\NDAS-I\d7\Projects\pSystem\Z80EM2010\UCSDI4.RAW boots fine**
  + **The LB version also gets an odd address error at the same point (DbgCnt = 88)**
    - **I saved the volume as *system-saved.vol* in *F:\NDAS-I\d7\Projects\pSystem\Volumes\Linux Vols***
    - **Running a version from this morning has the same problem?**

**2/29/2024:**

* **Possible solutions to crash**
  + **Anti-Virus problem- turned off real-time threat protection - still gets odd address error at DbgCnt = 88**
  + **Mismatch between interpreter and boot disk- I don’t think so.**
  + **Compare all sources for changes**
    - **InterpC.pas - no obvious problem**
    - **FilerMain.pas “”**
    - **pCodeDebugger\_Decl.pas “”**
    - **pCodeDebugger.pas “”**
    - **Filer\_Debugger.pas “”**
    - **Interp\_Const.pas “”**
    - **InterpII.pas “”**
  + **Put a break in ValidDictionary, SaveSegInfoForFile**
  + **UNDEF PM\_VER? no fix**
  + **Try to completely rebuild “”**
  + **The LB 1.4 boots OK**
  + **Trying to boot the PM 1.4 version with LB\_VER set, causes a RCE when OldDebugger is called**
  + **The PM 2.0 boots correctly and compiles properly**
  + **Try PM 1.5**
  + **Maintaining the BootList is not working. Delete a BootInfo does not get saved**
    - **MaintainBootFilesList calls TfrmLoadVersion which returns a ModalResult = 2 which is cancel**
    - **Somehow, the ModalResult of mrSave (12) gets changed to a ModalResult of mrCancel (2)?**
  + **Why does Delphi think that “F:\NDAS-I\Source Code\Other Sources\pr20240223\pSystem\Src\FilerMain.pas(4648)” is an open file? Because it was even though I had closed it many times. I think that it had a breakpoint set it in and the folder has been used to open an old version of the file.**
  + **Currently I cannot boot. Now I am getting an *mrCancel* on the Boot button? Calling Close in *btnBootClick* was setting the modal result to *mrCancel*. That explains the previous problem.**

**3/1/2024:**

* **I can boot from Z80EM2010\UCSDI4.RAW using the 1/5/2024 Filer\_Debugger.exe executable.**
* **And now, for no obvious reason, UCSDI4.raw is booting fine again? Maybe this had something to do with being run on “Leap Day” (Feb 29)?**
* **I am always getting two *Halted* messages whenever the 1.4 system is exited?**
* **Problem:** 
  + **Q: What calls CSPUnloadSegment?**
    - **A: RBP calls, Ret(0) calls CSPUnloadSegment.**
  + **Q: What calls CSPLoadSegment?**
    - **A: It is called by CXP**
* **I don’t see where the compiler gets loaded in InterpII. It was only requesting the 1st 64 bytes of the segment dictionary hence not being recognized as a segment dictionary.**
* **Halting the system:**
  + **SYSHALT called from Fetch**
  + **HaltPsys called from SYSHALT**
  + **HaltPsys raises ESYSTEMHALT**
  + **Which generates the message ‘'p-System halted: SYSHALT'’**
  + **Fetch catches the ESYSTEMHALT exception and calls**
  + **FinalException sets frmPCodeDebugger.fExceptionMessage to** 
    - **'p-System halted: SYSHALT (ESYSTEMHALT): Segment:6, ProcNum:1, RelIPC:24'**
  + **which sets fLastError so**
    - **'SYSHALT, p-System halted: SYSHALT (ESYSTEMHALT): Segment:6, ProcNum:1, RelIPC:24'**
* **Files that I have open:**
  + **Delphi: InterpC.PAS**
  + **Debugger: Dashboard 0**
  + **UEDIT32.EXE:**
    - **F:\NDAS-I\d7\Projects\pSystem\Temp\UCSDI4-RAW directory listing.txt**
    - **f:\ndas-i\projects\psystem\temp\JumpLog.txt**
  + **LibreOffice Calc**
    - **SEGMAP-UCSDI4-SYSTEM.COMPILER.CSV**
    - **SEGMAP-UCSDI4-SYSTEM.PASCAL (1).CSV**
  + **MS Access**
    - **Version-I-4.accdb**

**3/2/2024:**

* **PM 1.4 and LB 1.4 are not hitting the first UJP concurrently**
* **Both are the same until the month names have been abbreviated**
* **LB 1.4 takes and FJP in FINIT but PM 1.4 does not**
* **LB 1.4: FJP calls UJP directly. PM 1.4 calls *jump()* rather than calling UJP.**
  + **UJP calls *jump* this way:**
    - **w := jump(ShortInt(FetchUB()))**
  + **FJP calls *jump* this way:**
    - **w := jump(ShortInt(p1))**
* **Reminder: the crash occurs at over DbgCnt > 45000**
* **Version running on SurfacePro is not mounting TESTX**
* **LoadVersion is not always mounting the subsidiary volumes**
* **LB 1.4 doesn’t even show the compiler as having been loaded**
  + **It is seen being loaded by DumpDebugInfo but is wiped out**
* **Windows cannot find *UEDIT32.exe* (needed to be changed in the Filer settings). Maybe, if the the editor is not located where it is supposed to be, I could just execute {FileName}.TXT rather than UEDIT32.EXE {FileName}**
* **Maybe I should not be backing up .INI files to the *SurfacePro*? Or maybe I should mark them as *Read Only?***
* **If I were to mark them as *Read only* the *SurfacePro* would err out when it tried to update them.**
* **Trying to put the LB 1.4 breakpoint into UJP and the PM 1.4 breakpoint into *jump.*** 
  + **Ok to DbgCnt 133, 145, 205, 265, 325, 374, 386, 425, 503, 531, 559, 591, 623, 651, 679, 707, 739, 771, 803, 835 -- goes off the rails somewhere near here**
    - **The PM 1.4 dynamic call stack does not seem to properly display the location within the caller.**
    - **The PM 1.4 dynamic call stack seems to display the IPC+1 compared to PM**
    - **The LB 1.4 variable *A* is a *byte***
    - **The PM 1.4 variable *disp* is a *shortint***
  + **Switched to a breakpoint on a a negative *A* or *disp***
    - **At DbgCnt 475: RelIPC for both is 6. OK until DbgCnt >~ 923 goes off the rails somewhere near here**
* **MemDumpDW(0, wt\_DynamicCallStack) does not work correctly on InterpC.**
* **On SurfacePro, UEDIT32.exe has been moved to C:\My Private Desktop\Development\UltraEdit\UEDIT32.EXE**

**3/4/2024:**

* **The Browse button in Filer Settings seems to be looking for a .TXT file.**
* **Call stack is strange: 'Dynamic CallStack: COMPINIT.COMPINIT @ 2, COMPINIT.COMPINIT @ 2, PASCALCO.COMPINIT @ 27, COMPILER.COMPINIT @ 6, PASCALSY. @ 95, PASCALSY.COMPINIT @ 8'**
* **Put a breakpoint at COMPINT 27**
* **I Get the JTAB breakpoint at DbgCnt = 44296 as soon as I try to step into INITSCALARS. This is occurring in the CLP opcode. It is occurring on an attempt to CLP 17.**
* **None of the compiler listings that I have show INITSCALARS as procedure number 17. It is always seen as procedure number 7 (or 9).**
* **Trying to compile using PM 1.5 gets an odd address error when trying to compile TESTDEBUG.TEXT in the SIND opcode.** 
  + **'Dynamic CallStack: PASCALSY.CHECKFIL @ 5, COMPINIT.PROC07 @ 10580, COMPINIT.PROC01 @ 2, PASCALCO.PROC01 @ 51, Unknown\_56270.PROC01 @ 6, PASCALSY.PREFIXER @ 101, PASCALSY.PROC01 @ 8'**
* **Trying to compile using LB 1.5 seems to work fine.**

**3/5/2024:**

* **Re: compiling with PM 1.5: Odd address error**
  + **History**
    - **77266=CXP (COMPINIT.INITSCAL @10578),**
    - **77221=CLP (COMPINIT.COMPINIT @1),**
    - **77220=CXP (PASCALCO.PASCALCO @49),**
    - **… Once again it seems to occur immediately on the call to INITSCAL**
    - **The IPC is absurd**
  + **Call stack:**
    - **PASCALSY.CHECKFIL @ 5,**
    - **COMPINIT.FGET @ 10580,**
    - **COMPINIT.PASCALSY @ 2,**
    - **PASCALCO.PASCALSY @ 51,**
    - **Unknown\_56270.PASCALSY @ 6,**
    - **PASCALSY.PREFIXER @ 101,**
    - **PASCALSY.PASCALSY @ 8'**
  + **Possible culprits:**
    - **LPA - 1st use 2 DbgCnt = 72690**
      * **Getting called with a ByteCount = 10504 (= 20+ blocks)**
      * **In the p-Code, the LPA follows an LCA 8 ‘ ‘ and does NOT have any reason to jump over a huge block of 10504 bytes**
      * **This was the problem. I created two different versions of LPA-- LPA1 to be called with Word\_Address false and LPA2 to be called with Word\_Address true,**
* **Version 1.4 CSPUnitReadWriteCommon seems to be treating UNITREAD/UNITWRITE as if UNITREAD/UNITWRITE has fewer parameters - i.e., popping the UNUM off but thinking that it is UBUF.**
* **Currently trying to compile an run TEST.TEXT program- still getting an error when I try to run it. Maybe this is the wrong compiler for version 1.4?**

**3/6/2024:**

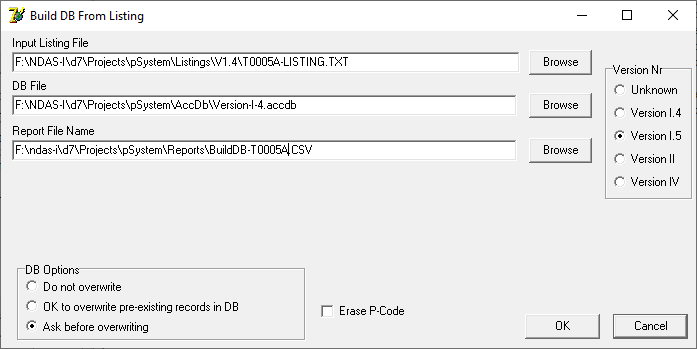
* **I want to try to find more version 1.4 compilers**
  + **So far, the only obvious version 1.4 volume is UCSDI4.VOL**
  + **Replacing the 1.4 compiler with the 1.5 compiler has the same problem with RCE in CallIO**
* **I can compile a basic program by**
  + **Rename SYSTEM.COMPILER to PASCAL.COMPILER**
  + **Rename BASIC.COMPILER to SYSTEM.COMPILER**
  + **Change the file type of SYSTEM.COMPILER to *.CODE***
  + **Edit the the and compile it**
  + **But when I try to run it, I get an *ASSERTion* failure in *CspLoadSegment.***

**3/7/2024:**

* **Z80 system compiles and runs T0001A**
* **It also compiles and runs my simple TEST.TEXT program**
* **UNITWRITE works fine when the system is writing to #1**
* **Make a list of SYSTEM.PASCAL file sizes and version numbers**
  + **Saved to: F:\NDAS-I\d7\Projects\pSystem\Reports\System.pascal-FileDates.xlsx**
* **Does PM Version II work? Yes.**
* **PM 1.5 does not work**
* **LB 1.4 DOES work**
* **Maybe some previous instruction is screwing up the stack? Try to “seed” the UNITWRITE parameters**
* **Deleting a file from the “Files to Load List” does not work**
* **The LB 1.4 debugger does not show any files**
* **Stack before each opcode in TEST.TEXT:**
  + **LAO: ' @ $01F4 [ 500]:**
  + **LCA: ' @ $01F2 [ 498]: D776**
  + **SAS: ' @ $01F0 [ 496]: D7CD D776**
  + **SLDC2: ' @ $01F2 [ 498]:**
  + **LAO: ' @ $01F2 [ 498]: 0002**
  + **SLDC1: ' @ $01F0 [ 496]: D776 0002**
  + **IXS: ' @ $01EE [ 494]: 0001 D776 0002**
  + **SLDC10: ' @ $01EE [ 494]: 0001 D776 0002**
  + **SLDC0: ' @ $01EC [ 492]: 000A 0001 D776 0002**
  + **SLDC0: ' @ $01EA [ 490]: 0000 000A 0001 D776 0002**
  + **CSP: ' @ $01E8 [ 488]: 0000 0000 000A 0001 D776 0002**
* **IXS is not doing the index?**
  + **Should be popping two things but only pushing one?**

**3/8/2024:**

* **Working on T0005A.TXT.**
* **Listing Saved to: F:\NDAS-I\d7\Projects\pSystem\Listings\V1.4\T0005A-LISTING.TXT**

****

* **Added to database F:\NDAS-I\d7\Projects\pSystem\AccDb\Version-I-4.accdb**
* **Executing SYSTEM.WRK.CODE. When trying to run, operating system calls it *USERPROG***
* **Decoder tries to call *GetBaseAddress* which is *abstract*.**
* **I am only getting the abstract error when I use *Decode (1) USERPROG.USERPROG*-- NOT when I use *Decode Specified Memory…***
* **There seems to be a phantom SYSTEM.WRK.CODE which has a code address in an empty region of TEST: (at block 536)**
* **BaseRegister: 'InterpII not yet implemented'**

**3/11/2024:**

* **Abstract error occurs because InterpII did not implement GetBaseAddress**
* **T0005A runs from SYSTEM.WRK.CODE which is located at block 812**
* **Does T0005A work with version 1.5 and/or 2.0?**
* **Seems like SYMBUFP and SYMCURSOR must be correct since it is finding the correct string.**
* **Getting s listindex(0) out of bounds when I try to merge in the external decoder window**
* **No database files are getting backed up? Need some kind of warning when the database list is incorrect.**
* **TempDescrip (local to ids) is not displaying correctly in the debugger**
* **Redundant code in BackupDBToTextFiles**

**3/12/2024:**

* **Sometimes forget what terminal was selected**
* **Problem with exception handling revealed by T0015A in LB 4.0**
  + **Attempt to access byte 0 of a strong raises an EXEQERR in CSTR (“Invalid string length: 0”)**
  + **In the *Fetch* code, there is an attempt to do “on e:EXEQERR do”. This is where the crash occurs:**
    - ***“FastMM has detected an attempt to call a virtual method on a freed object. An access violation will now be raised in order to abort the current operation.”* The freed object must be “e”.**
  + **Look for references to EXEQERR and see if they are using fErrCode**

**3/13/2024:**

* **Version IV still disappearing when I try to run T0015A-- follows range check error**