```
   let SetUpManFunBuffs[] be
   §SMFB
       MFDefBuff := lv (StructureVec↓StructureLPtr)
       MFAppBuff := NewVec[AppSize]
       PresMFDef := UNKNOWN
       PresMFApp := UNKNOWN
       MFDefPtr := MFDefBuff
       MFAppPtr := MFAppBuff
       MFDefEnd := lv (StructureVec↓StructureNPtr) - 1
       MFAppEnd := MFAppBuff + AppSize
       Store := false
       Link := false
       MFDefBuffFull := false
       MaxAppUsage := 0
   §SMFB


   and CloseManFunBuffs[] be
   §CMFB
       unless (Mode ∧ 1) = 0 do
           ReportS['Def, app sizes = *N, *N', MFDefPtr - MFDefBuff,
               MaxAppUsage]
       ReturnVec[MFAppBuff, AppSize]
   §CMFB


   and ManFunDefinition[Type] be
   §MFD
       let RHop = Hop
       and RSSP = SSP
       and F, G = FindName[Ch], LocalGenNo
       and OldNRD = NonRecDef
       if Type = MANRTDF do UpdateNameType[F, MANRT]
       ClearStack[]
       if NameVal[F] = UNUSED do
       §   StartOutputMode[IGNORE]
           Command[]
           ClearStack[]
           SSP := RSSP
           EndOutputMode[]
           return
       §
       Hop := HopNo[]
       CheckScope[F, LineNo, GetNo]
       SSP := -1
       LocalGenNo := Generation[PresentLevel]
       StartNonRecDef[]
       test MFDefBuffFull
         ifso  §   StartOutputMode[NORMAL]
                   ForHop[Hop]
                   SetManFun[NameNo[F], NameVal[F]]
               §
         ifnot §   StartOutputMode[MFDEF]
                   SetupNewMFDef[F]
               §
       Command[]
       ClearStack[]
       test MFDefBuffFull
         ifso  UpdateNameVal[F, UNSET]
         ifnot
```

```
            §   EndOutputMode[]
                StartOutputMode[NORMAL]
                ForHop[Hop]
                SetManFun[NameNo[F], NameVal[F]]
                OutputMFDef[]
                UpdateNameVal[F, PresMFDef]
                CloseMFDef[]
            §
        NormalOutput[2, ENDMFDEF, NameNo[F]]
        EndNonRecDef[OldNRD]
        LocalGenNo := G
        SSP := RSSP
        ForHopPt[Hop]
        Hop := RHop
        EndOutputMode[]
    $MFD


    and SetupNewMFDef[F] be
    §SNMFD
        let PrevMFDef = PresMFDef
        PresMFDef := MFDefPtr
        if PresMFDef + DEFHEADSIZE ≥ MFDefEnd do
        §   CloseMFDefBuff[F]
            PresMFDef := PrevMFDef
            return
        §
        PresMFDef↓PREV := PrevMFDef
        PresMFDef↓CODETYPE := DIRECT
        PresMFDef↓INITHOPNO := HopNo[]
        PresMFDef↓NOPARAMS := 0
        PresMFDef↓HEADTYPE := NEWMANFUNDEF
        PresMFDef↓NAMEBLOCK := F
        PresMFDef↓PARAMSSAVED := 0
        MFDefPtr := PresMFDef + DEFHEADSIZE
    $SNMFD


    and OutputMFDef[] be
    §OMFD
        let p = PresMFDef + DEFHEADSIZE
        until p ≥ MFDefPtr do
        §   unless CellType[p] = NEWMANFUNDEF do CellOutput[p]
            p := NextCell[p]
        §
    $OMFD


    and CloseMFDef[] be
    §CMFD
        PresMFDef↓CODELENGTH := MFDefPtr - PresMFDef
        PresMFDef↓NOHOPS := HN - PresMFDef↓INITHOPNO + 1
        if PresMFDef↓NOPARAMS = 0 do PresMFDef↓CODETYPE := DIRECT
        IsDefPerfect[]
        PresMFDef := PresMFDef↓PREV
    $CMFD


    and ManFunDefParams[] be
    §MFDP
        let n = SSP + 1
        PresMFDef↓NOPARAMS := n
    $MFDP
```

```
    and OutputtoMFDef[p] be
    §OMFD
        switchon CellType[p] into
130     §s
            case SPsm:
            case LPsm:
                    if CellNo[p] ≤ PresMFDef↓NOPARAMS - 1 do
                        PresMFDef↓CODETYPE := INDIRECT
135                 OuttoMFDef[p]
                    endcase

            case SAsm:
            case SGsm:
140         case LGsm:
            case APPLYsm:
                    Store := true
                    OuttoMFDef[p]
                    endcase
145
            case LOADSTATIC:
                    unless p↓1 = RAsm do Store := true
                    OuttoMFDef[p]
                    endcase
150
            default:
                    if Store do
                        unless PresMFDef↓CODETYPE = INDIRECT do
                            PresMFDef↓CODETYPE := SEMIDIRECT
155
            case RTEXITsm:
            case FNEXITsm:
            case RETURNsm:
                    Store := false
160
            case FORHOPPT:
            case BACKHOPPT:
                    OuttoMFDef[p]
        $s
165 $OMFD


    and OuttoMFDef[p] be
    §OMFD
170     test (MFDefPtr + GenCellLength[p] ≤ MFDefEnd) ∧  MFDefBuffFull
          ifso  §   let q = MFDefPtr
                    CopyCell[p, q]
                    MFDefPtr := NextCell[q]
                §
175       ifnot §   CloseMFDefBuff[PresMFDef↓NAMEBLOCK]
                    CellOutput[p]
                §
    $OMFD

180
    and IsDefPerfect[] be
    §IP
        let m, n, i, p = 0, 0, 0, 0
        if (PresMFDef↓CODETYPE = INDIRECT) ∨ (PresMFDef↓NOPARAMS = 0) return
185     p := PresMFDef + (DEFHEADSIZE + CHKJMPSIZE)
        m := CellNo[p]
        n := PresMFDef↓NOPARAMS - 1
        i := m
        §   unless (CellNo[p] = i) ∧ (CellType[p] = RPsm) return
190         i := i + 1
            p := NextCell[p]
```

```
        § repeatuntil i > n
        until (CellType[p] = FNEXITsm) ∨ (CellType[p] = RTEXITsm) do
        §   if (CellType[p] = RPsm) ∧ (m ≤ CellNo[p] ≤ n) return
195         p := NextCell[p]
        §
        PresMFDef↓PARAMSSAVED := n - m + 1
    $IP

200
    and CloseMFDefBuff[F] be
    §CMFDB
        ClearStack[]
        EndOutputMode[]
205     StartOutputMode[NORMAL]
        ForHop[Hop]
        SetManFun[NameNo[F], NameVal[F]]
        OutputMFDef[]
        MFDefBuffFull := true
210     unless (Mode ∧ 4) = 0 do Error[351, SOFT, NameField[F], NullProgram]
        PresMFDef := PresMFDef↓PREV
    $CMFDB


215 and OutputtoMFApp[p] be
    §OMFA
        switchon CellType[p] into
        §s
            case LINKsm:
220                 test Link
                      ifso  §   NewMFApp[]
                                Link := false
                                (PresentOutput = MFAPP → OuttoMFAppBuff,
                                    CellOutput)[p]
225                       §
                      ifnot OuttoMFAppBuff[p]
                    endcase

            case LOADMANFUN:
230                 if UNDEFINED ≠ CellNo2[p] ≠ UNSET do
                    §   if Link do
                        §   EndOutputMode[]
                            ReportNonApp[]
                            CloseMFApp[]
235                         StartOutputMode[MFAPP]
                        §
                        Link := true
                        OuttoMFAppBuff[p]
                        endcase
240                 §

            default:test Link
                      ifso  §   ReportNonApp[]
                                CloseMFApp[]
245                             Link := false
                                CellOutput[p]
                          §
                      ifnot OuttoMFApp[p]
        §s
250 §OMFA


    and OuttoMFApp[p] be
    §OMFA
255     OuttoMFAppBuff[p]
        if CellType[p] ≠ APPLYsm ∨ PresMFApp = UNKNOWN return
```

```
        if PresMFApp↓APPLYNO = CellNo[p] do SubstituteCode[]
    $OMFA

260
    and NewMFApp[] be
    §NMFA
        let PrevMFApp = PresMFApp
        MFCode := rv( MFAppPtr - 1)
265     if MFAppPtr + APPHEADSIZE ≥ MFAppEnd do
        §   DefaultCloseMFAppBuff[]
            return
        §
        PresMFApp := MFAppPtr
270     PresMFApp↓CODELENGTH := APPHEADSIZE
        PresMFApp↓HEADTYPE := NEWMANFUNAPP
        PresMFApp↓CODEPTR := MFCode
        PresMFApp↓PREV := PrevMFApp
        PresMFApp↓CODETYPE := MFCode↓CODETYPE
275     PresMFApp↓APPLYNO := SSP
        MFAppPtr := PresMFApp + APPHEADSIZE
    $NMFA


280 and OuttoMFAppBuff[p] be
    §OMFAB
        test MFAppPtr + GenCellLength[p] ≤ MFAppEnd
          ifso  §   let q = MFAppPtr
                    CopyCell[p, q]
285                 MFAppPtr := NextCell[q]
                §
          ifnot §   DefaultCloseMFAppBuff[]
                    CellOutput[p]
                §
290 $OMFAB


    and SubstituteCode[] be
    §SC
295     MFCode := PresMFApp↓CODEPTR
        NoParams := MFCode↓NOPARAMS
        HopShift := HN - MFCode↓INITHOPNO + 1
        StackNo := PresMFApp↓APPLYNO
        ParamsSaved := MFCode↓PARAMSSAVED
300     test CanSubstituteCode[]
          ifso  §   let PrevApp = PresMFApp↓PREV
                    Params[PresMFApp↓CODETYPE]
                    P1 := MFCode + (DEFHEADSIZE + CHKJMPSIZE) +
                        RPSIZE × MFCode↓PARAMSSAVED
305                 Substitute[]
                    HN := HN + MFCode↓NOHOPS
                    ManApp := NameType[MFCode↓NAMEBLOCK]
                    CloseMFApp[]
                    PresMFApp := PrevApp
310             §
          ifnot DefaultCloseMFAppBuff[]
    $SC


315 and CanSubstituteCode[] = valof
    §CSC
        let CodeLength = MFCode↓CODELENGTH
        and ParamLength = MFAppPtr - 1 - PresMFApp
        let AppUsage = MFAppPtr - MFAppBuff + 4 × (NoParams - ParamsSaved) +
320             (CodeLength ≥ ParamLength → CodeLength, ParamLength)
        if MaxAppUsage < AppUsage do MaxAppUsage := AppUsage
```

```
        resultis (AppUsage < AppSize)
    $CSC

325
    and Params[Type] be
    §P
        let MaxP = NoParams - ParamsSaved
        and ParamEnd = MFAppPtr - APSIZE
330     and i, r = 0, 0
        ParamBlock := MFAppEnd + 1 - 4 × MaxP
        P1 := PresMFApp + (APPHEADSIZE + LINKSIZE)
        P2 := PresMFApp - LMFSIZE
        until P1 ≥ ParamEnd do
335     §u
            test CellType[NextCell[P1]] = ENDPARAM
              ifnot §   ModTransferCell[i - r - 2] repeatuntil
                            CellType[P1] = ENDPARAM ∧ CellNo[P1] = StackNo
                        unless r ≥ MaxP do
340                         ParamBlock↓(4 × r), ParamBlock↓(4 × r + 1) :=
                                RPsm, i + StackNo
                        i := i + 1
                    §
              ifso  test Type = INDIRECT ∨ r ≥ MaxP
345                   ifso  §   ModTransferCell[i - r - 2]
                                unless r ≥ MaxP do
                                §   ParamBlock↓(4 × r) := RPsm
                                    ParamBlock↓(4 × r + 1) := i + StackNo
                                §
350                             i := i + 1
                            §
                      ifnot switchon CellType[P1] into
                            §s
                                case RPsm:
355                                     if CellNo[P1] > StackNo do
                                            UpdateCellNo[P1, i - r - 2]
                                case Nsm:
                                case LOADMANFUN:
                                case LGsm:
360                                     CopyCell[P1, ParamBlock + 4 × r]
                                        P1 := NextCell[P1]
                                        endcase

                                case RAsm:
365                             case RGsm:
                                case LOADSTATIC:
                                case LOADSTRING:
                                        if Type = DIRECT do
                                        §   CopyCell[P1, ParamBlock + 4 × r]
370                                         P1 := NextCell[P1]
                                            endcase
                                        §

                                default:ModTransferCell[i - r - 2]
375                                     ParamBlock↓(4 × r) := RPsm
                                        ParamBlock↓(4×r + 1) := i + StackNo
                                        i := i + 1
                            §s
            P1 := NextCell[P1]
380         r := r + 1
        §u
        ParamsSaved := ParamsSaved + r - i
        unless r = NoParams do
        §   Error[350, SOFT, r - NoParams, NullProgram]
385         OutputtoMFAppBuff[Ssm, NoParams - r + i + StackNo - 1]
            for s = r to MaxP - 1 do
```

```
            ParamBlock↓(4 × s), ParamBlock↓(4 × s + 1) := Nsm, 0
        $
    $P
390

    and ModTransferCell[n] be
        switchon CellType[P1] into
        §s
395         case APPLYsm:
            case Ssm:
            case JUMPPTsm:
            case RPsm:
            case LPsm:
400         case SPsm:
                if CellNo[P1] > StackNo do UpdateCellNo[P1, n]

            default:CopyCell[P1, P2]
                P2 := NextCell[P2]
405
            case ENDPARAM:
                P1 := NextCell[P1]
        $s

410
    and Substitute[] be
    §S
        let Returning, ReturnSS = false, false
        switchon CellType[P1] into
415     §s
            case RPsm:
                test 0 ≤ CellNo[P1] < NoParams
                  ifso  TransferParam[CellNo[P1]]
                  ifnot EditCell[StackShift[CellNo[P1] + 1]]
420               endcase

            case Ssm:
            case JUMPPTsm:
            case APPLYsm:
425             EditCell[StackNo - ParamsSaved]
                endcase

            case LPsm:
            case SPsm:
430             EditCell[StackShift[CellNo[P1] + 1]]
                endcase

            case COERCETOVAL:
                if CellType[NextCell[P1]] = FNEXITsm do
435             §   FHN := CellNo[P1] + HopShift
                    SSP := SSP - 1
                    LastRelExt := (CellNo2[P1] ≠ UNSET)
                    P1 := NextCell[P1]
                    endcase
440             §
                EditCVCell[HopShift, StackNo - ParamsSaved]
                endcase

            case FORHOPPT:
445         case BACKHOPPT:
            case FORHOP:
            case BACKHOP:
            case CONDFORHOP:
                EditCell[HopShift]
450             endcase
```

```
            case SWITCH:
                    TransferSwitch[]
                    endcase
455
            default:TransferCell[]
                    endcase

            case RETURNsm:
460               § let n = MFCode↓INITHOPNO + HopShift
                    OutputtoMFAppBuff[FORHOP, n]
                    Returning := true
                    ReturnSS := (CellNo[P1] + 1 ≠ ParamsSaved)
                    P1 := NextCell[P1]
465                 endcase
                  §

            case FNEXITsm:
                    if ParamsSaved ≠ CellNo[P1] do
470                     test FHN = UNSET
                          ifso  §   OutputtoMFAppBuff[SPsm, StackNo]
                                    OutputtoMFAppBuff[Ssm, StackNo]
                                §
                          ifnot §   OutputtoMFAppBuff[Ssm, StackNo - 1]
475                               LastRelExt := true
                                §
                    MFAppPtr := P2
                    return

480         case RTEXITsm:
                    if Returning do
                        OutputtoMFAppBuff[FORHOPPT,
                            MFCode↓INITHOPNO + HopShift]
                    if ParamsSaved ≠ CellNo[P1] + 1 ∨ ReturnSS do
485                     OutputtoMFAppBuff[Ssm, StackNo - 1]
                    MFAppPtr := P2
                    return

            case NEWMANFUNDEF:
490                 P1 := NextCell[P1]
                    endcase
        $s repeat
    $S

495
    and StackShift[n] = n ≤ NoParams → StackNo, StackNo - ParamsSaved


    and OutputtoMFAppBuff[a, b, c, d] be
500 §OMAB
        CopyCell[lv a, P2]
        P2 := NextCell[P2]
    §OMAB

505
    and CloseMFApp[] be
    §CMFA
        ClearStack[]
        EndOutputMode[]
510     if PresentOutput ≠ MFAPP do
        §   let p = PresMFApp = UNKNOWN → MFAppBuff, PresMFApp - LMFSIZE
            and q = MFAppPtr
            and RSSP = SSP
            MFAppPtr := p
515         until p ≥ q do
            §   StackOutput[p]
```

```
                p := NextCell[p]
            §
            SSP := RSSP
520     §
    §CMFA


    and DefaultCloseMFAppBuff[] be
525 §DCMFAB
        ClearStack[]
        unless (Mode ∧ 4) = 0 do
            Error[352, SOFT, NameField[MFCode↓NAMEBLOCK], NullProgram]
        until PresentOutput ≠ MFAPP do
530     §   EndOutputMode[]
            unless PresMFApp = UNKNOWN do PresMFApp := PresMFApp↓PREV
        §
      § let p = PresMFApp = UNKNOWN → MFAppBuff, PresMFApp - LMFSIZE
        and q = MFAppPtr
535     MFAppPtr := p
        until p ≥ q do
        §   switchon CellType[p] into
            §s
                case LOADMANFUN:
540                     p↓2 := UNSET

                default:CellOutput[p]

                case NEWMANFUNAPP:
545             case ENDPARAM:
            §s
            p := NextCell[p]
        §
    §DCMFAB
550


    and TransferCell[] be
    §TC
        CopyCell[P1, P2]
555     P1 := NextCell[P1]
        P2 := NextCell[P2]
    §TC


560 and EditCell[n] be
    §EC
        CopyCell[P1, P2]
        UpdateCellNo[P2, n]
        P1 := NextCell[P1]
565     P2 := NextCell[P2]
    §EC


    and EditCVCell[h, s] be
570 §ECC
        CopyCell[P1, P2]
        UpdateCellNo[P2, h]
        unless CellNo2[P2] = UNSET do UpdateCellNo2[P2, s]
        P1 := NextCell[P1]
575     P2 := NextCell[P2]
    §ECC


    and TransferParam[n] be
580 §TP
        CopyCell[ParamBlock + 4 × n, P2]
```

```
        P1 := NextCell[P1]
        P2 := NextCell[P2]
    §TP
585

    and TransferSwitch[] be
    §TS
        CopyCell[P1, P2]
590     P2↓3, P2↓4 := P2↓3 + HopShift, P2↓4 + HopShift
        P1 := NextCell[P1]
        P2 := NextCell[P2]
    §TS

595
    and CopyCell[p, q] be
        for i = 0 to GenCellLength[p] do q↓i := p↓i

****
```