```
-- TimeConvert.Mesa  Edited by Johnsson on March 21, 1978  8:38 AM

DIRECTORY
  InlineDefs: FROM "inlinedefs",
  StringDefs: FROM "stringdefs",
  TimeDefs: FROM "timedefs";

DEFINITIONS FROM TimeDefs;

TimeConvert: PROGRAM IMPORTS StringDefs EXPORTS TimeDefs SHARES TimeDefs =
BEGIN

--This should be a constant in TimeDefs, but...
DefaultTime: PackedTime = PackedTime[0,0];
UP: TYPE = POINTER TO UnpackedTime;



DivideTime: PROCEDURE [num: PackedTime, den: CARDINAL]
  RETURNS [quotient: PackedTime, remainder: CARDINAL] =
  BEGIN OPEN InlineDefs;
  t: CARDINAL;
  [quotient.highbits, t] ← LDIVMOD[num.highbits,0,den];
  [quotient.lowbits, remainder] ← LDIVMOD[num.lowbits,t,den];
  RETURN
  END;

MultiplyTime: PROCEDURE [multiplicand: PackedTime, multiplier: CARDINAL]
  RETURNS [result: PackedTime] =
  BEGIN OPEN InlineDefs;
  t: CARDINAL;
  result.highbits ← multiplicand.highbits * multiplier;
  [result.lowbits, t] ← LongMult[multiplicand.lowbits, multiplier].product;
  result.highbits ← result.highbits + t;
  RETURN
  END;

AddTime: PROCEDURE [a: PackedTime, b: INTEGER] RETURNS [PackedTime] =
  BEGIN
  t: CARDINAL = a.lowbits;
  a.lowbits ← a.lowbits + b;
  IF b > 0 THEN
    BEGIN IF a.lowbits < t THEN a.highbits ← a.highbits+1 END
  ELSE IF a.lowbits > t THEN a.highbits ← a.highbits-1;
  RETURN[a]
  END;

CurrentDayTime: PUBLIC PROCEDURE RETURNS [PackedTime] =
  BEGIN
  t: HardwareTime ← currentTime↑;
  IF currentParameters.beginDST = 0 THEN -- compatibility
    RETURN[AddTime[[highbits: t.high, lowbits: t.low],8*60*60]]
  ELSE RETURN[[highbits: t.high, lowbits: t.low]]
  END;

TimeParameters: PROCEDURE
  RETURNS [beginDST, endDST: CARDINAL, zone, zoneminutes: INTEGER] =
  BEGIN
  direction: WestEast;
  [beginDST: beginDST, endDST: endDST, zone: zone, zoneminutes: zoneminutes, direction: direction] ← cu
**rrentParameters↑;
  IF beginDST = 0 OR endDST = 0 THEN -- compatibility, assume California
    BEGIN zone ← 8; beginDST ← 121; endDST ← 305 END
  ELSE IF direction # west THEN
    BEGIN zone ← -zone; zoneminutes ← -zoneminutes END;
  RETURN
  END;

UnpackDT: PUBLIC PROCEDURE [p: PackedTime] RETURNS [unp: UnpackedTime] =
  BEGIN
  MonthTable: ARRAY [0..12] OF CARDINAL =
    [0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366];
  u: UP = @unp;
  day4, day, yr4: CARDINAL;
  month: CARDINAL ← 1;
  t: InlineDefs.LongCARDINAL;
```

```
  beginDST, endDST: CARDINAL;
  zone, zoneminutes: INTEGER;

  [beginDST, endDST, zone, zoneminutes] ← TimeParameters[];
  IF p = DefaultTime THEN p ← CurrentDayTime[];
  [p, u.second] ← DivideTime[p,60];
  p ← AddTime[p,-zoneminutes]; -- ignore underflow
  [p, u.minute] ← DivideTime[p,60];
  u.zone ← zone;
  u.dst ← FALSE;
  p ← AddTime[p,-zone]; -- ignore underflow
  DO -- have to repeat if DST
    [t, u.hour] ← DivideTime[p,24];
    u.weekday ← (t.lowbits + StartWeekDay) MOD 7;
    [yr4,day4] ← InlineDefs.DIVMOD[t.lowbits,DaysInFourYears];
    day4 ← (IF day4 >= 2*365+31+28 THEN 3 ELSE (day4+(365-31-28))/365) + day4;
    [day4, day] ← InlineDefs.DIVMOD[day4, 366];
    u.year ← BaseYear + yr4*4 + day4;
    WHILE day >= MonthTable[month] DO month ← month + 1 ENDLOOP;
    u.month ← month ← month-1;
    day ← day + 1;
    u.day ← day - MonthTable[month];
    IF u.dst OR ~CheckDateGE[u, day, beginDST, 2]
      OR CheckDateGE[u, day, endDST, 1] THEN EXIT;
    p ← AddTime[p,1];
    u.dst ← TRUE;
    ENDLOOP;
  RETURN
  END;

InvalidTime: PUBLIC ERROR = CODE;

PackDT: PUBLIC PROCEDURE [unp: UnpackedTime, computeDST: BOOLEAN] RETURNS [t: PackedTime] =
  BEGIN
  MonthTable: ARRAY [0..12] OF CARDINAL =
    [0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366];
  u: UP = @unp;
  year, month, day, day1, hour, minute, second: CARDINAL;
  yr3: [0..3];
  beginDST, endDST: CARDINAL;
  zone, zoneminutes: INTEGER;

  IF (year ← u.year-BaseYear) >= 136 OR
     (month ← u.month) >=12 OR
     (day ← u.day) ~IN [1..31] OR
     (hour ← u.hour) >= 24 OR
     (minute ← u.minute) >= 60 OR
     (second ← u.second) >= 60 THEN ERROR InvalidTime;
  yr3 ← year MOD 4;
  IF day > LOOPHOLE[MonthTable[month+1] - MonthTable[month], CARDINAL] OR
    (month = 1 AND day = 29 AND yr3 # 3) THEN ERROR InvalidTime;

  [beginDST, endDST, zone, zoneminutes] ← TimeParameters[];

  -- compute days this year in day1
  day1 ← MonthTable[month] + day;
  IF yr3 # 3 AND month >= 2 THEN day1 ← day1-1;

  t.highbits ← 0;
  t.lowbits ← (year/4)*DaysInFourYears + yr3*365 + day1 - 1;
  u.weekday ← t.lowbits MOD 7;
  t ← AddTime[MultiplyTime[t,24],hour];
  IF computeDST THEN
    BEGIN
    IF CheckDateGE[u,day1,beginDST,2]
      AND ~CheckDateGE[u,day1,endDST,2] THEN zone ← zone - 1
    END
  ELSE IF u.dst THEN zone ← zone - 1;

  t ← AddTime[MultiplyTime[
      AddTime[MultiplyTime[
        AddTime[t,zone],60],minute+zoneminutes],60],second];
  RETURN
  END;

AppendDayTime: PUBLIC PROCEDURE [s: STRING, unp: UnpackedTime] =
```

```
  BEGIN
  MonthNames: PACKED ARRAY [0..36) OF CHARACTER =
    ['J,'a,'n,'F,'e,'b,'M,'a,'r,'A,'p,'r,'M,'a,'y,'J,'u,'n,
     'J,'u,'l,'A,'u,'g,'S,'e,'p,'O,'c,'t,'N,'o,'v,'D,'e,'c];
  template: PACKED ARRAY [0..18) OF CHARACTER =
    [' ,'0,'-,'x,'x,'x,'-,'0,'0,' ,' ,'0,':,'0,'0,':,'0,'0];
  u: UP = @unp;
  p: CARDINAL ← s.length;
  m: CARDINAL;
  w2d: PROCEDURE [v: INTEGER] =
    BEGIN
    CharZero: CARDINAL = LOOPHOLE['0];
    d1, d2: INTEGER;
    [d1, d2] ← InlineDefs.DIVMOD[v,10];
    IF d1 # 0 THEN s[p] ← LOOPHOLE[d1 + CharZero, CHARACTER];
    s[p+1] ← LOOPHOLE[d2 + CharZero, CHARACTER];
    p ← p + 3;
    END;

  FOR m IN [0..LENGTH[template]) DO
    StringDefs.AppendChar[s, template[m]];
    ENDLOOP;
  w2d[u.day];
  m ← u.month*3;
  THROUGH [0..2] DO
    s[p] ← MonthNames[m]; p ← p + 1; m ← m + 1 ENDLOOP;
  p ← p + 1;
  w2d[u.year MOD 100];
  w2d[u.hour];
  w2d[u.minute];
  w2d[u.second];
  RETURN
  END;

AppendFullDayTime: PUBLIC PROCEDURE [s: STRING, unp: UnpackedTime] =
  BEGIN
  KnownZones: TYPE = [4..10];
  zones: PACKED ARRAY KnownZones OF CHARACTER ← ['A,'E,'C,'M,'P,'Y,'H];
  AppendDayTime[s,unp];
  IF unp.zone IN KnownZones THEN
    BEGIN OPEN StringDefs;
    AppendChar[s,' ];
    AppendChar[s,zones[unp.zone]];
    AppendChar[s,IF unp.dst THEN 'D ELSE 'S];
    AppendChar[s,'T];
    END;
  END;

CheckDateGE: PROCEDURE [u: UP, days, dstDay, dstHour: INTEGER] RETURNS [BOOLEAN] =
  BEGIN
  weekday: INTEGER ← u.weekday;
  RETURN[
    IF days < dstDay-6 THEN FALSE
    ELSE IF days > dstDay THEN TRUE
    ELSE IF weekday = 6 THEN u.hour >= dstHour
    ELSE days-weekday > dstDay-6]
  END;

END...
```