```
-- TimeConvert.Mesa  Edited by Sandman on August 5, 1977  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;

MonthTable: ARRAY [0..12] OF CARDINAL =
  [0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366];
MonthNames: STRING = "JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC";


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: CARDINAL] RETURNS [PackedTime] =
  BEGIN
  t: CARDINAL = a.lowbits;
  a.lowbits ← a.lowbits + b;
  IF a.lowbits < t THEN a.highbits ← a.highbits+1;
  RETURN[a]
  END;

CurrentDayTime: PUBLIC PROCEDURE RETURNS [PackedTime] =
  BEGIN
  time: TYPE = MACHINE DEPENDENT RECORD [high,low: CARDINAL];
  clock: POINTER TO time = LOOPHOLE[5728];
  t: time = clock↑;
  RETURN[PackedTime[highbits: t.high, lowbits: t.low]]
  END;

UnpackDT: PUBLIC PROCEDURE [t: PackedTime] RETURNS [unp: UnpackedTime] =
  BEGIN
  u: UP = @unp;
  day4, day, yr4: CARDINAL;
  month: CARDINAL ← 1;

  IF t = DefaultTime THEN t ← CurrentDayTime[];
  [t, u.second] ← DivideTime[t,60];
  [t, u.minute] ← DivideTime[t,60];
  [t, u.hour] ← DivideTime[t,24];
  [t,day4] ← DivideTime[t,DaysInFourYears];
  yr4 ← t.lowbits;
  day4 ← day4 + (SELECT day4 FROM
                 >= 2*365+31+28 => 3,
                 >= 365+31+28 => 2,
                 >= 31+28 => 1,
                 ENDCASE => 0);
  [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;
    u.day ← day - MonthTable[month]+1;
    -- insert daylight computation here
    u.dst ← FALSE;
    RETURN
    END;

  InvalidTime: PUBLIC ERROR = CODE;

  PackDT: PUBLIC PROCEDURE [unp: UnpackedTime] RETURNS [t: PackedTime] =
    BEGIN
    u: UP = @unp;
    year, month, day, day4, hour, minute, second: INTEGER;
    yr3: [0..3];

    IF (year ← u.year-BaseYear) >= 176 OR
       (month ← u.month) >=12 OR
       (day ← u.day) ~IN [0..31] OR
       (hour ← u.hour) >= 24 OR
       (minute ← u.minute) >= 60 OR
       (second ← u.second) >= 60 THEN ERROR InvalidTime;
    yr3 ← year MOD 4;
    IF month = 1 AND day = 29 AND yr3 # 3 THEN ERROR InvalidTime;
    day4 ← yr3*365 + MonthTable[month] + day-1;
    IF yr3 # 3 AND month # 2 THEN day4 ← day4-1;
    IF u.dst THEN
      BEGIN day4 ← day4 - 1; hour ← hour+23 END;
    t.lowbits ← (year/4)*DaysInFourYears+day4;
    t.highbits ← 0;
    t ← AddTime[MultiplyTime[t,24],hour];
    t ← AddTime[MultiplyTime[t,60],minute];
    t ← AddTime[MultiplyTime[t,60],second];
    RETURN
    END;

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

    StringDefs.AppendString[s, " 0-xxx-00  0:00:00"];
    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;

  DayOfWeek: PUBLIC PROCEDURE [t: PackedTime] RETURNS [[0..7)] =
    BEGIN
    u: UnpackedTime = UnpackDT[t];
    day: CARDINAL;
    t ← DivideTime[t,3600].quotient;
    t ← DivideTime[t,24].quotient;
    day ← t.lowbits + StartWeekDay;
    IF u.dst AND u.hour = 0 THEN day ← day+1;
    RETURN[day MOD 7]
    END;

  END...
```