// ===================================================

/\*    \*\*\* NOTES \*\*\*

REGARDING STRINGS:

I use the String class, because I'm using the 1284p

processor which has plenty of memory available. But for

a library, these should all be changed to character

strings to conserve memory.

REGARDING SERIAL COMMUMNICATIONS:

My 1284p processor, has two serial ports.

Serial.print(n) prints to the screen, while

"serial1.write(s,6)" send to the FN-RM01. The

"SoftwareSerial" library must be used for single-port

processors.

ENDING PLAY:

If set to repeat play, a bug in FN-RM01's call to stop

playing often results in the call stopping the current

play or the next beginning play not both. This allows

play to continue repeating. So be sure to set FN-RM01 to

"single play" BEFORE calling "stop play".  Below, I use

the function, repeatPlay(0) to accomplish this.

\*/

// These are input pins on the Arduino, used for control buttons

const byte up = 3;

const byte select = 4;

const byte dn = 5;

int chk;

unsigned long audioMill = 0;

unsigned long recordingMills = 60000; // maximum recording length

boolean troubleShooting = false;

boolean playing = false;

float volLevel;

byte volNow = 255;

String playsFolder = "GOALS";

void message(String s)

{

  Serial.println(s);

}

void clear1()

{

  while (Serial1.available() > 0)

    Serial1.read();

}

int hex(char \*hex)

{

  byte n = (int)strtol(hex, NULL, 16);

  return n;

}

byte addChk(byte n)

{

  chk += n;

  return n;

}

void repeatPlay(byte x)

{

  //Use "repeatPlay(0)" to make sure audio is set to SINGLE play.

  // Because stopping "auto repeat" near the transition point

  // will NOT stop both the already ending play and the newly

  // starting play. This would allow play to continue repeating.

  byte ss[6];

  clear1();

  chk = 0;

  ss[0] = (hex("7E")); // startCode

  ss[1] = (addChk(hex("04"))); // number

  ss[2] = (addChk(hex("AF"))); // command

  ss[3] = (addChk(x)); // Parameter

  ss[4] = chk; // checkCodere

  ss[5] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,6);

  rCode("repeatPlay");

}

void audioDelay()

{

  while (audioMill > millis()) {};

  audioMill = millis() + 250;

}

byte rCode(char s[20])

{

  audioDelay();

  byte code = 255;

  unsigned long mill = millis() + 5000;

  while ((mill > millis()) && (!Serial1.available())) {};

  if (Serial1.available() > 0)

    code = Serial1.read();

  if (troubleShooting)

    ///Serial.writeln(s);

  return code;

}

void endPlay()

{

  repeatPlay(0);

  byte ss[5];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (hex("03")); // number

  ss[2] = (hex("AB")); // command

  ss[3] = (hex("AE")); // checkCode

  ss[4] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,5);

  rCode("endPlay");  // Get the code returned.

  playing = false;

}

byte v31\_10(byte vol)

// divides the volume settings into 10 levels,

// each roughly double the previous.

{

  switch (vol)

  {

  case 0:

    return 0;

  case 1:

    return 1;

  case 2:

    return 2;

  case 3:

    return 3;

  case 4:

    return 4;

  case 5:

    return 5;

  case 6:

    return 10;

  case 7:

    return 15;

  case 8:

    return 20;

  case 9:

    return 25;

  case 10:

  default:

    return 31;

  }

}

void volume10(byte vol)

{

  volume(v31\_10(vol));

}

void volume(byte vol)

{

  if (vol == volNow)

    return;

  volNow = vol;

  clear1();

  chk = 0;

  byte ss[6];

  ss[0] = hex("7E"); // startCode

  ss[1] = addChk(hex("04")); // number

  ss[2] = addChk(hex("AE")); // command

  ss[3] = addChk(vol); // volume level

  ss[4] = chk; // checkCode

  ss[5] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,6);

}

void pausePlay()

{

  byte ss[6];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (hex("03")); // number

  ss[2] = (hex("AA")); // command

  ss[3] = (hex("AD")); // checkCode

  ss[4] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,5);

  rCode("PausePlay");

  while (touched()) {};

}

void usbMode()

{

  byte ss[6];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (hex("04")); // number

  ss[2] = (hex("D2")); // command

  ss[3] = (hex("01")); // command

  ss[4] = (hex("D7")); // checkCode

  ss[5] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,6);

  rCode("usbMode");

}

void fastBack1()

{

  byte ss[6];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (hex("03")); // number

  ss[2] = (hex("D1")); // command

  ss[3] = (hex("D4")); // checkCode

  ss[4] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,5);

  rCode("fastBack");

}

void fastBack()

{

  fastBack1();

  delay(333);

  while (touched()) {

  };

  fastBack1();

}

void fastForward1()

{

  byte ss[6];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (hex("03")); // number

  ss[2] = (hex("D0")); // command

  ss[3] = (hex("D3")); // checkCode

  ss[4] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,5);

  rCode("fastForward");

}

void fastForward()

{

  fastForward1();

  delay(1000);

  while (touched()) {

  };

  fastForward1();

}

byte read1(byte x)

{

  boolean b = false;

  unsigned long mill = millis() + 1000;

  while (mill > millis())

    while (Serial1.available() > 0)

    {

      if (b)

        return Serial1.read();

      if (Serial1.read() == x)

        b = true;

    }

  return 0;

}

byte query()

{ // 01: Playing 02: Stopped 03: Paused 04: Recording 05: Fast forward/backward

  boolean b = false;

  clear1();

  byte ss[6];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (hex("03")); // number

  ss[2] = (hex("C2")); // command

  ss[3] = (hex("C5")); // checkCode

  ss[4] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,5);

  for (byte i=0; i<5; i++)

    if (rCode("194") == 194)

      break; //returns 194, then the result code.

  return rCode("query");

}

byte fExists(char folder[], char name[])

{

  char fName[14];

  strcpy(fName, folder);

  strcat(fName, name);

  clear1();

  chk = 0;

  byte ss[19];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (addChk(hex("10"))); // number

  ss[2] = (addChk(hex("CC"))); // command

  for (byte i=0; i<13; i++)

    ss[i+3] = (addChk(fName[i]));

  ss[16] = (chk); // checkCode

  ss[17] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,18);

  return (rCode("fExists") == 0);

}

void fileNotFound(char\* folder,char\* name)

{

  message("The file:~" + String(name) + "was not found in Audio folder ~" +String(folder));

  wait(0);

  if (String(folder) != "SETUP")

  {

    message("To update your list of available files, put Audio card in Data slot and restart ~DreamMaster.");

    wait(0);

  }

}

void play(char name[9]) // once, no checking

{

  char fName[14];

  strcpy(fName, "SETUP");

  strcat(fName, name);

  chk = 0;

  byte ss[19];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (addChk(hex("10"))); // number

  ss[2] = (addChk(hex("A5"))); // command

  for (byte i=0; i<14; i++)

    ss[i+3] = (addChk(fName[i]));

  ss[16] = (chk); // checkCode

  ss[17] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,18);

  rCode("play");

  playing = true;

}

void doPlay(char folder[], char name[], byte repeat, byte vol)

{

  unsigned long mill = 0;

  volume10(vol);

  byte Q = 1;

  repeatPlay(repeat);

  String s;

  char fName[14];

  strcpy(fName, folder);

  strcat(fName, name);

  clear1();

  chk = 0;

  byte ss[19];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (addChk(hex("10"))); // number

  ss[2] = (addChk(hex("A5"))); // command

  for (byte i=0; i<14; i++)

    ss[i+3] = (addChk(fName[i]));

  ss[16] = (chk); // checkCode

  ss[17] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,18);

  if (rCode("doPlay"))

    fileNotFound(folder,name);

  else

  {

    playing = true;

    while ((repeat == 1) || ((Q == 1) || (Q == 3) || (Q == 5)))

    {

      backgroundFunctions();

      Q = query();

      if (digitalRead(up))

      {

        if (Q == 3)

          fastBack();

        else

        {

          if (vol > 0)

            vol--;

          volume10(vol);

        }

        while (touched()) {

        };

      }

      if (digitalRead(dn))

      {

        if (Q == 3)

          fastForward();

        else

        {

          if (vol < 10)

            vol++;

          volume10(vol);

          while (touched()) {

          };

        }

      }

      if (digitalRead(select))

      {

        mill = millis() + 1000;

        while (digitalRead(select))

          if (mill < millis())

          {

            endPlay();

            return;

          }

        pausePlay();

      }

    }

    endPlay();

  }

}

//void doPlay(char folder[], char name[], byte pMen, byte pItm, byte repeat, boolean show)

boolean touched() // if anything high

{

  return ((digitalRead(up) || digitalRead(dn) || digitalRead(select)));

}

void backgroundFunctions()

{

  // Run other functions from here as needed

}

void wait(byte x) // Stop program until a button is pressed.

{

  if (x > 0)

  {

    Serial.print("wait: ");

    Serial.println(x);

  }

  while (touched())

  {

    backgroundFunctions();

  }

  while (!touched())

  {

    backgroundFunctions();

  }

  while (touched())

  {

    backgroundFunctions();

  }

}

String byteToStr(byte x)

{

  char chars[4];

  sprintf(chars, "%d", x);

  return String(chars);

}

void rSearch(char name[9])

{

  clear1();

  chk = 0;

  byte ss[19];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (addChk(hex("0B"))); // number

  ss[2] = (addChk(hex("CB"))); // command

  for (byte i=0; i<8; i++)

    ss[i+3] = (addChk(name[i]));

  ss[11] = (chk); // checkCode

  ss[12] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,13);

  rCode("rSearch");

}

void doRecord(char folder[6],char name[9],boolean wheel)

{// "recording(of a file)by indexed sequence in the root directory"

  int i,j;

  char fName[14];

  char s[9];

  strcpy(s,name);

  strcpy(fName, folder);

  strcat(fName, s);

  clear1();

  chk = 0;

  byte ss[19];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (addChk(hex("10"))); // number

  ss[2] = (addChk(hex("D8"))); // command

  for (byte i=0; i<14; i++)

    ss[i+3] = (addChk(fName[i]));

  ss[16] = (chk); // checkCode

  ss[17] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,18);

  byte theCode = rCode("doRecord");

  switch (theCode)

  {

  case 0:

    while (true)

    {

      if (touched())

      {

        endRecord();

        break;

      }

      if (wheel && (recordingMills < millis()))

      {

        endRecord();

        play("STOPTALK");

        return;

      }

    }

    break;

  case 1:

    message("Audio Card is full or missing.");

    return;

  case 2:

    message("Unable to record.");

    return;

  default:

     message("Cannot record: ~" + String(s) +

      " in folder: ~" +String(folder)+ " Error code: " + byteToStr(theCode));

  }

}

void endRecord()

{// "Stop recording"

  byte ss[6];

  ss[0] = (hex("7E")); // startCode

  ss[1] = (hex("03")); // number

  ss[2] = (hex("D9")); // command

  ss[3] = (hex("DC")); // checkCode

  ss[4] = (hex("7E")); // endCode

  audioDelay();

  Serial1.write(ss,5);

}

void playNext()

{

  byte ss[5];

  ss[0] = (hex("7E"));

  ss[1] = (hex("03"));

  ss[2] = (hex("AC"));

  ss[3] = (hex("AF"));

  ss[4] = (hex("7E"));

  audioDelay();

  Serial1.write(ss,5);

  rCode("C");

  getFNumber();

}

void getFNumber()

{

  byte ss[5];

  ss[0] = (hex("7E"));

  ss[1] = (hex("03"));

  ss[2] = (hex("C9"));

  ss[3] = (hex("CC"));

  ss[4] = (hex("7E"));

  audioDelay();

  Serial1.write(ss,5);

  rCode("D");

  unsigned long mill = millis() + 2000;

  while (mill > millis())

    while (Serial1.available() > 0)

      Serial.print(Serial1.read());

}

void setup()

{

  Serial1.begin(9600);

}

void loop()

{

}