```
/*develop cold start device
 *the "prime function" (to blip the throttle) has been removed!
 *increase rpm = D5 limit = D6 (in)
 *decrease rpm = D4 limit = D7 (out)
 *auto - manual = D10
 *A0 spare
 *A1 spare
 *A2 air temp
 *A3 engine temp
 *A4 spare setup param
 *A5 rate of idle reduction set
 *D3, D11 H-Bridge enables
 *D12, D13 H-Bridge current direction
 *D8, D9 spare
 *there are approx 750 steps between the limit switches
 *21.1.12
 */
 #include <avr/pgmspace.h>

PROGMEM const byte idleTable[] = {
0,0,0,5,10,20,40,60,80,90,100,0,0,5,10,20,40,60,80,90,100,5,5,10,15,20,40,60,80,
90,100,10,10,15,20,25,45,65,80,90,100,20,20,25,30,35,50,70,80,90,100,20,25,30,40
,45,55,75,80,90,100,30,35,40,45,55,60,75,85,90,100,30,40,45,50,60,65,75,85,90,10
0,40,50,55,60,65,70,75,85,90,100,50,60,60,70,70,80,80,90,90,100
};


int engineTemp;
int airTemp;
int index;  // index to idle table obtained from engine and air temp
int idleValue;
int intcel;  // celsius turned to integer
int concel;  // integer celsius constrained
float celsius;
float tValue;
int count;
int idleReduction;  // number of steps in each of 4 phases of idle reduction

 int calibtime;  // time between each phase of idle reduction
 int d = 2;  // stepper motor speed parameter
 int val4 = digitalRead(4);
 int val5 = digitalRead(5);
 int val6 = digitalRead(6);
 int val7 = digitalRead(7);
 int val10 =digitalRead(10);
 int value2;  // raw analogue value
 int value3;  // raw analogue value

 //--------------------------------------------------------------------------

void setup() {

  Serial.begin(9600);

  pinMode(4,INPUT); // manual decrease rpm (out)
  pinMode(5,INPUT); // manual increse rpm (in)
  pinMode(6,INPUT); // limit switch (in)
  pinMode(7,INPUT); // limit switch (out)
  pinMode(10,INPUT); // auto/manual switch
  pinMode(3,OUTPUT); // H bridge enable
  digitalWrite(3,LOW);
  pinMode(11,OUTPUT); // H bridge enable
  digitalWrite(11,LOW);
  pinMode(12,OUTPUT); // current direction through coil A
  pinMode(13,OUTPUT); // current direction through coil B
```

```
 // the following outputs could control the choke flap when fitted
 pinMode(14,OUTPUT);
 pinMode(15,OUTPUT);
 pinMode(8,OUTPUT);
 pinMode(9,OUTPUT);

 // get engine and air temperature and process values
 // engine first
 tValue = 0;
 for(count = 0; count < 10; count++){
 value3 = analogRead(3);
 tValue = tValue + value3;
 }
 tValue = tValue / 10;
 celsius = (tValue / 1024.0) * 500;

 intcel = floor(celsius + 0.5);
 concel = constrain( intcel, 10, 70);
 engineTemp = map(concel, 10, 70, 1, 10);

 Serial.print("engine temp = ");
 Serial.println(celsius);
 Serial.print("mappped = ");
 Serial.println(engineTemp);
 Serial.println("++++++++++++++++++++++++++++++++");

 // next air
 tValue = 0;
 for(count = 0; count < 10; count++){
 value2 = analogRead(2);
 tValue = tValue + value2;
 }
 tValue = tValue / 10;
 celsius = (tValue / 1024.0) * 500;

 intcel = floor(celsius + 0.5);
 concel = constrain( intcel, 3, 30);
 airTemp = map(concel, 3, 30, 1, 10);

 Serial.print("air temp = ");
 Serial.println(celsius);
 Serial.print("mappped = ");
 Serial.println(airTemp);
 Serial.println("++++++++++++++++++++++++++++++++");

 index = engineTemp + 10*airTemp - 10;

 Serial.print("index = ");
 Serial.println(index);

idleValue = pgm_read_byte(&idleTable[index]);
idleValue = idleValue * 7;
idleReduction = (750 - idleValue)/4;


 Serial.print("idle value = ");
 Serial.println(idleValue);
 Serial.print("Idlereduction = ");
 Serial.println(idleReduction);
 Serial.println("++++++++++++++++++++++++++++++++");

 // get setup parameters
 int aval4 = analogRead(4);
 // aval4 is spare
 int aval5 = analogRead(5);
```

```
  //map calibtime between 3 and 249

  calibtime = map(aval5,0,1023,3,240);

  Serial.print("idle time param (calibtime) = ");
  Serial.println(calibtime);
  Serial.print("spare raw val A4 = ");
  Serial.println(aval4);
  Serial.println("++++++++++++++++++++++++++++++++");

 }
//-------------------------------------------------------------

void loop() {
 // initialise position - max idle
  d = 2;
  do {
   val6 = digitalRead(6);
   inaStep();
  } while(val6 == HIGH);

  // if manual swich set, adjust to medium idle then go to manual
   val10 = digitalRead(10);
  if(val10 == HIGH) {
    for( int c = 0; c < idleReduction; c++) {
    val7 = digitalRead(7);
    if(val7 == HIGH) {
    outaStep();
  }
  }
    goto manual;
  }
  // if engine is hot goto manual
  //if(aval3 > 50) {
    //goto manual;
    //}


  // set to fast idle value from temperatures and conversion table

  for( int c = 0; c < idleValue; c++) {
    val7 = digitalRead(7);
    if(val7 == HIGH) {
    outaStep();
  }
  }

  // gradually reduce idle speed in 4 stages

  d = 10;

  for ( int x = 0; x < 4; x++) {
  for( int i = 0; i < calibtime; i++) {
   delay(1000);
   val4 = digitalRead(4);
   val5 = digitalRead(5);
   val10 = digitalRead(10);
   if(val5 == LOW or val4 == LOW or val10 == HIGH) {
     goto manual;
   }
  }
  for( int c = 0; c < 200; c++) {
    val7 = digitalRead(7);
    if(val7 == HIGH) {
```

```
    outaStep();
   }
   }
   }

   //manual operation
   manual:
   d = 2;

   while (true != false) {
   val4 = digitalRead(4);
   val5 = digitalRead(5);
   val6 = digitalRead(6);
   val7 = digitalRead(7);
   if(val4 == LOW and val7 == HIGH) {
      outaStep();
   }
    if(val5 == LOW and val6 == HIGH) {
      inaStep();
    }
   }
}

//--------------------------------------------------------------------------

// functions to turn stepper motor one step in either direction

void outaStep() {
    digitalWrite(3, HIGH);
    digitalWrite(11, HIGH);

    digitalWrite(12, HIGH);
    digitalWrite(13, LOW);
    delay(d);
    digitalWrite(12, LOW);
    digitalWrite(13, LOW);
    delay(d);
    digitalWrite(12, LOW);
    digitalWrite(13, HIGH);
    delay(d);
    digitalWrite(12, HIGH);
    digitalWrite(13, HIGH);
    delay(d);
    digitalWrite(3, LOW);
    digitalWrite(11, LOW);
  }

  //--------------------------------------------------------------------------

  void inaStep() {
    digitalWrite(3, HIGH);
    digitalWrite(11, HIGH);

    digitalWrite(12, HIGH);
    digitalWrite(13, HIGH);
    delay(d);
    digitalWrite(12, LOW);
    digitalWrite(13, HIGH);
    delay(d);
    digitalWrite(12, LOW);
    digitalWrite(13, LOW);
    delay(d);
    digitalWrite(12, HIGH);
    digitalWrite(13, LOW);
    delay(d);
```

```
 digitalWrite(3, LOW);
 digitalWrite(11, LOW);
}

//--------------------------------------------------------------------------------
```